# ACM SIGACT News Distributed Computing Column 8

Sergio Rajsbaum*

July 30, 2002

**Abstract**

The Distributed Computing Column covers the theory of systems that are composed of a number of interacting computing elements. These include problems of communication and networking, databases, distributed shared memory, multiprocessor architectures, operating systems, verification, internet, and the web.

This issue consists of the paper "Distributed Computing Research Issues in Grid Computing" by Henri Casanova. Many thanks to Henri for contributing to this issue.

**Request for Collaborations:** Please send me any suggestions for material I should be including in this column, including news and communications, open problems, and authors willing to write a guest column or to review an event related to theory of distributed computing.

## Distributed Computing Research Issues in Grid Computing

**Henri Casanova**
**Department of Computer Science and Engineering**
**University of California at San Diego**
**La Jolla, CA 92093-0114**
casanova@cs.ucsd.edu

**Abstract**

Ensembles of distributed, heterogeneous resources, or *Computational Grids*, have emerged as popular platforms for deploying large-scale and resource-intensive applications. Large collaborative efforts are currently underway to provide the necessary software infrastructure. *Grid computing* raises challenging issues in many areas of computer science, and especially in the area of *distributed computing*, as Computational Grids cover increasingly large networks and span many organizations. In this paper we briefly motivate Grid computing and introduce its basic concepts. We then highlight a number of distributed computing research questions, and discuss both the relevance and the shortcomings of previous research results when applied to Grid computing. We choose to focus on issues concerning the dissemination and retrieval of information and data on

*HP Cambridge Research Laboratory, One Cambridge Center, Cambridge, MA 02142-1612. Sergio.Rajsbaum@hp.com, rajsbaum@math.unam.mx. On leave from Instituto de Matemáticas, UNAM.

Computational Grid platforms. We feel that these issues are particularly critical at this time, and as we can point to preliminary ideas, work, and results in the Grid community and the distributed computing community. This paper is of interest to distributing computing researchers because Grid computing provides new challenges that need to be addressed, as well as actual platforms for experimentation and research.

## 1 Introduction

As computation, storage, and communication technologies steadily improve, increasingly large, complex, and resource-intensive applications are being developed both in research institutions and in industry. It is a common observation that computational resources are failing to meet the demand of those applications. The power of network, storage, and computing resources is projected to double every 9, 12, and 18 months, respectively. As noted in [28], those three constants have important implications. Anticipating the trends in storage capacities (and price), application developers and users are planning increasingly large runs that will operate on and generate petabytes of data. Although microprocessors are reaching impressive speeds, in the long run they are falling behind storage. As a result, it is becoming increasingly difficult to gather enough computational resources for running applications at a single location. Fortunately, improvements in wide-area networking make it possible to aggregate distributed resources in various collaborating institutions and to form what have come to be known as *Computational Grids* (or *Grids*). To date, most Grid applications have been in the area of scientific computing as scientists world-wide are resorting to numerical simulations and data analysis techniques to investigate increasingly large and complex problems. Recently, Grid computing has been identified as a critical technology by industry for enterprise computing and business-to-business computing [31].

The term *Grid* was coined in the late 90s [29] to describe a set of resources distributed over wide-area networks that can support large-scale distributed applications. The analogy likens the Grid to the electrical power grid: access to computation and data should be as easy, pervasive, and standard as plugging in an appliance into an outlet. This analogy is appealing and was made as early as 1965 [28]. The term *Grid computing* has been widely adopted (e.g. see articles in the New York Times on August 9th and 12th, 2001). In fact, the term has been used in so many contexts that it has become difficult to get a clear picture of what *Grid computing* really is.

In the foundational paper "The Anatomy of the Grid" [31], Foster, Kesselman, and Tuecke attempt to address this problem by (re-)defining the *Grid problem* as *coordinated resource sharing and problem solving in dynamic, multi-institutional, virtual organizations.* This concept of a *virtual organization* (VO) is central to Grid computing. A simplified view is that a VO is a set of participants with various relationships that wish to share resources to perform some task. In that paper, Foster et al. argue that the Grid problem is thus central not only to "e-science", but also to industry, where the coordination of distributed resources both within and across organizations is increasingly important.

Grid computing has been the focus of a tremendous amount of research and development effort, both in research institutions and in industry. Even though the technology is in its early development stages and is still evolving rapidly, Grid systems are being deployed and used worldwide. This situation creates a great opportunity for computer science researchers in several areas for two reasons. First, many crucial computer science research questions need to be answered in order to deploy and operate Grids effectively. Second, now that basic infrastructure elements are in place [36], the Grid has become a viable platform for such research. Indeed, as communities of users start running applications, large amounts of empirical trace data are becoming available for determining relevant characteristics of Grid platforms; examples include usage patterns, resource availability, and resource contention. These data make it possible to build increasingly realistic models that are critical for conducting Grid computing research.

It is well-known that lack of communication between research communities is an impediment to scientific and technological progress. For instance, lack of communication between the networking and the distributed computing communities was noted during PODC'00 [53]. Grid computing is no exception. The area really evolved from High Performance Computing (HPC), which explains the large amount of Grid-related activities at traditionally HPC conferences (e.g. SC [64], HPDC [40]). Connections with other research communities have become necessary because of the drastic changes in platform scale and usage model when moving from traditional HPC to Grid computing. For instance, current work in areas such as Internetworking are extremely relevant as, after all, Grid computing takes place on wide-area networks. However, although connections and collaborations exist, they are difficult to establish as communities tend to publish in their own conferences, employ different vocabularies, and value different types of contributions.

An area that is clearly critical for Grid computing is *distributed computing*. Traditional assumptions that are more or less valid in traditional HPC settings break down on the Grid. In HPC settings, one often assumes a "well-behaved" system: no faults or failures, minimal security requirements, consistency of state among application components, availability of global information, and simple resource sharing policies. While those assumptions are arguably valid in tightly coupled systems, they break down as systems become more distributed. Fortunately, the distributed computing community has long investigated algorithmic solutions for distributed computing when the aforementioned assumptions do not hold. Results obtained in that arena provide a number of algorithmic solutions and system design principles that can be leveraged for managing large VOs. Therefore, Grid computing provides an exciting and high-impact area in which distributed computing results are extremely relevant. However, many new challenges need to be addressed for those results to be applied and deployed successfully.

Many observers have noted that many activities in the field of Grid computing over the last 5 years have been closer to advanced development than to pure research. What has been termed as "Grid computing research" is really combined research and engineering with the concrete goal of producing a widely-used artifact: a software infrastructure. Originally, progress was made by developing prototype software for concrete applications that could be deployed on early testbeds. The software employed simple engineering solutions that were often sub-optimal (e.g. they do not ensure good scalability for large VOs). Results in the distributed computing community have shown that more sophisticated solutions in the form of evolved algorithms and protocols can provide significant improvements. Such solutions have not yet been implemented by Grid developers. This is due both to the sheer magnitude of the initial implementation effort, and also to challenges that are inherent to Grid computing. Now that a basic infrastructure is in place, it is a critical time for distributed computing researchers to understand the concepts and goals of Grid computing and to make contributions in the form of algorithms, protocols, and theoretical models.

The main goal of this paper is to highlight distributed computing research issues that are (or soon will be) critical for the success of Grid computing. Grid computing is broad and diverse in its domain of application, and the associated research questions span many areas of distributed computing and of computer science in general. Rather than providing a cursory discussion of many areas, we choose to focus on two that we feel are particularly relevant and for which we can point to ideas, work, and preliminary results in both the Grid community and the distributed computing community: namely, **dissemination of and access to data and information**.

We describe the state of Grid computing in Section 2 and discuss the concept of a Virtual

Organization in Section 3. We discuss relevant distributed computing research issues pertaining to data and information in Sections 4 and Section 5. In Section 6, we take a broader focus by surveying other research questions and areas that are associated with Grid computing, and by discussing next steps.

## 2 Grid Computing

In the previous section we have motivated the need for Grid computing. We give here a brief introduction to basic concepts, to set the stage for the later sections of this paper. This material borrows heavily from the two foundational papers *The Anatomy of the Grid* [31] and *The Physiology of the Grid* [30] which make fundamental contributions by defining the field and providing a common vocabulary.

Let us state again the Grid problem as presented in [31]: *coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations.* This definition expresses several distinct dimensions of the Grid problem:

- The word "resource" is to be taken in a broad sense, to include data, computers, scientific instruments, software, etc.
- Sharing must be "coordinated" in that resources together with their providers/consumers are clearly defined, and in that multiple resources may need to be organized in an integrated fashion to achieve various qualities of service. Achieving this coordination involves the establishment and enforcement of sharing agreements [22].
- The ability to negotiate resource sharing agreements in a dynamic and flexible fashion enables a wide variety of problem solving methodologies, ranging from collaborative engineering to distributed data mining.
- Membership in a VO is "dynamic" as participants may join or leave at any time. A *Virtual Organization* (VO) is then the set of individuals and institutions defined by this sharing. Many examples of existing or envisioned VOs show that the concept spans a broad spectrum of purpose, size, structure, and duration.

"The Anatomy of the Grid" [31] identifies and defines a set of common requirements. One critical observation is that a simple *client-server* model is not sufficiently flexible for enabling most VOs. Instead, a spectrum of architectures ranging from client-server to general *peer-to-peer* are necessary as participants are alternately resource providers or consumers. Another observation is that currently available distributed computing technologies are not appropriate to enable VOs. Either those technologies do not support the wide variety of required services and resources, or they suffer from lack of flexibility and control needed for enabling the type of resource sharing necessary. As a result, there is a need for defining a new technology for supporting VOs: *a Grid software infrastructure.*

The software infrastructure presented in [31] places a large emphasis on interoperability as it is fundamental to ensure that VO participants can share resources dynamically and across different platforms, programming environments, and languages. To achieve interoperability, it is necessary to specify *Grid protocols.* These protocols can then be implemented as part of Application Programming Interfaces (APIs) and Software Development Kits (SDKs) to provide layered programming abstractions. We review Grid architecture layers in the following section.

## 2.1 Grid Architecture Layers

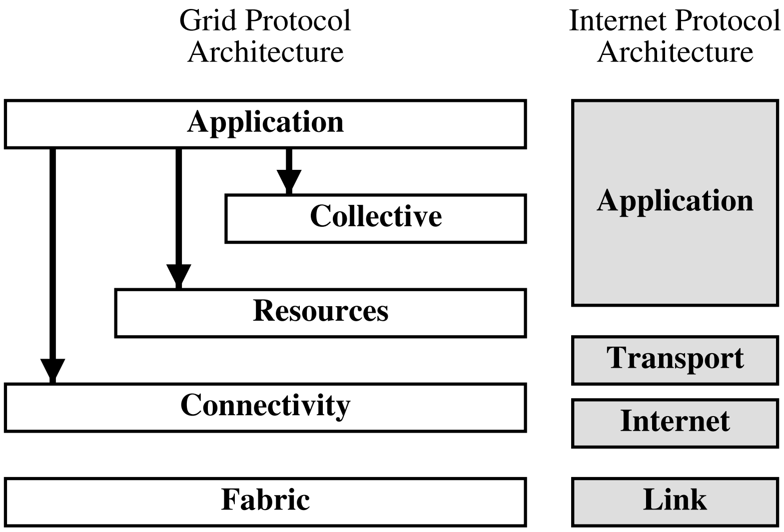


Figure 1: The layered Grid architecture and its relationship to the Internet protocol architecture - reproduced from [31].

The Grid *fabric* provides the lowest level of access to actual resources (e.g. computer, disk, file system, cluster of computers) and implements the mechanisms that allow those resources to be utilized. More specifically, those mechanisms must at least include *state enquiry* and *resource management* mechanisms, each of which must be implemented for a large number of native systems. The Grid *connectivity* layer defines communication, security, and authentication protocols required for network transactions between resources. The Grid *resource* layer builds on the connectivity layer to implement protocols that enable the use and sharing of individual resources. More specifically, two fundamental components are (i) information protocols for querying the state of a resource; and (ii) management protocols to negotiate access to a resource. The Grid *collective* layer focuses on the coordination of multiple resources. Example of functionalities include resource discovery, co-allocation, scheduling. This is the layer which is of greatest interest for the purpose of this paper as it is where many challenging distributed computing questions must be answered. Finally, the *application* layer is where VO applications are implemented and may use several of the previous layers. The resulting architecture is depicted in Figure 1.

The description of this architecture in [31] uses components of the Globus toolkit [36] as a concrete example of Grid layer implementations. Globus is a large, open-source, community-based effort at the Argonne National Laboratory and the Information Science Institute; Globus provides a software infrastructure that is currently leveraged by most Grid efforts. As of late 2001, over 12 companies had announced endorsement for the Globus Toolkit. Beyond the Globus toolkit, the Global Grid Forum [33] is a consortium of over 1000 academic researchers and industrial partners whose goal is to make recommendations for Grid infrastructure development.

## 2.2 Current Developments and Limitations

In this section, we describe components of the Grid infrastructure that address the management of application data and resource information, as they are the focus of our discussion in the rest of this paper.

The infrastructure that focuses on management of distributed application data is commonly labeled a *Data Grid* [18]. An increasing number of scientific disciplines manage large data collections generated by measurements and derivation of measurement data. As a result, many Data Grids are currently being deployed [25, 4, 52, 50, 5]. Infrastructure targeting resource information is often referred to as a *Grid Information Service* [21]. A number of research groups have designed and prototyped components for collecting, indexing, and publishing Grid information. The problems of indexing, discovering, and accessing such "Grid information services" is in some respects quite similar to those encountered when indexing, discovery, and accessing other data sources. However, we will see in the rest of this paper that both infrastructures raise a number of distinct research questions from a distributed computing perspective.

For both infrastructures, appropriate *data schemas* must be defined so that information can be encoded, stored, and searched in an efficient manner. A number of recent developments have made contributions in that area. In the Data Grid context, the Chimera system [32] targets a data schema that can be used to establish a *virtual data catalog* that describes all ways in which data in the catalog has been derived. This is a generic solution that should be applicable to many different VOs and has been demonstrated for high-energy physics and astronomy applications. In the context of Grid Information Services, schemas are being developed for various Grid resource types as part of the GGF [33] activities in the Grid Information Services working group [34]. Commonalities with Common Information Model (CIM) [19] are also being explored.

The definition of schemas is an important, but in some sense mundane, issue. More challenging is the design and implementation of a distributed system that implements mechanisms to publish information, disseminate information, notify participant of information changes, locate information, and retrieve information. Initial Grid infrastructure efforts have engineered software solutions for those mechanisms (e.g. [27]). Those mechanisms have made it possible to take the first steps in Grid computing and have been key to making the Grid a plausible platform. However, a large part of those efforts were focused on "getting it to work," without directly addressing issues of scalability, reliability, and information quality.

Now that we are facing VOs that contain thousands of individuals in hundreds of institutions world-wide, issues such as scalability and usability are becoming a near-term concern. These issues are being increasingly recognized by the Grid computing community and recent work explores avenues of research that are strongly connected to distributed systems and distributed computing research questions. In that sense, Grid computing presents a key opportunity for distributed systems and distributed computing researchers. Grid developers are implementing large scale infrastructures such as GriPhyn as this paper is being written, and those infrastructures provide a great "playground" to explore research issues in a concrete setting that will have a major impact on disciplinary science. Furthermore, information dissemination techniques developed in the distributed systems community (e.g. wide-area group communications) have shortcomings that must be addressed for Grid computing.

In the next section we detail the concepts defining VOs and, inspired by current work, give an example VO scenario for a hypothetical scientific community. That scenario will be used later in the paper to motivate and illustrate research challenges in the area of distributed computing.

## 3 Virtual Organizations

An important factor that has driven the evolution from HPC systems and applications to Grid computing is the widespread deployment of high-speed wide-area networks. The dramatic increase in network connectivity makes it feasible to consider deploying applications that tightly couple geographically distributed resources, data, and users.

This distribution has major implications when designing a software infrastructure to support problem solving activities. In a tightly coupled systems, such as a Massively Parallel Processor (MPP), it is possible to obtain an accurate picture of the global state of the system and to control its components in a centralized fashion. In a VO, distributed ownership and high-latency networks render the HPC approach to system design infeasible. In order to illustrate the challenges the Grid community is facing, we present a hypothetical VO and the activities it supports.

Consider a community of thousands of users that span hundreds of research institutions worldwide and who all focus on overlapping portions of a common scientific problem. Those users and institutions form a VO. Even though the software infrastructure to support a VO of that magnitude is not fully deployed at the time of writing, a number of large multi-institution projects are underway and are making rapid progress in that direction. A notable such project is GriPhyn [4], and our discussion is inspired by GriPhyn accomplishments and current developments.

The Grid available to members of our example VO consist of several types of resources: (i) a number of scientific instruments that generate raw experimental measurement data (e.g. particle collider, radio telescope); (ii) compute resources ranging from desktop workstations to clusters and MPPs, that are used to perform derivations and analysis of the measurement data as well as simulations; and (iii) storage resources on which measurement and derived data can be stored. All resources are interconnected via wide-area networks and are prone to downtime, either due to failures or to maintenance tasks.

Members of the VO, or software agents acting on their behalf, wish to perform a variety of tasks. For example:

- Publish new measurement data,
- Locate data items matching some criteria,
- Retrieve particular data items efficiently,
- Locate appropriate compute resources to run a simulation or data analysis task,
- Be constantly informed when new "relevant" data is produced,
- Publish new derived data of potential interest to other members of the VO,
- Be constantly informed of the load on a selected number of resources during the next 2 hours.

In the GriPhyn project, the overall goals of those activities is to allow VO members to construct a large collection of data and derived data. These goals must be achieved in a collaborative fashion so that little or no redundant work is performed. In other words, if some user has already performed a derivation on some portion of the measurement data and has stored it on a VO storage resources, other users should be able to discover and retrieve that data easily, rather than wasting (perhaps large amounts of) time performing redundant computation.

Information about application data is only one part of the picture. In order to achieve performance, it is critical to gain information about Grid hardware resources. For instance, when a user wishes to launch a computation, a software agent could decide which compute resource is most appropriate. An example of an agent is an *application-level scheduler* [10], which attempts to select resources that optimize a particular user's achieved performance. More specifically, an agent is a software component that understands the user's application requirements (e.g. "must use data from database X") and the user's objective (e.g. "reduce execution time"). An agent can then select appropriate resources, enact data movements and job launches, and potentially modify the way the application is running in order to adapt to changing resource conditions. Those decisions must take into account the location of the input data for that computation, as well as the basic capabilities of candidate resources (e.g. CPU speed). In addition, dynamic resource information such as compute resource loads (e.g. CPU load, batch queue length) or network speed (e.g. bandwidth between a storage resource and a compute resource) must also be taken into account. Timely access to this information is critical to achieve good performance as Grid resources are shared and exhibit dynamic performance characteristics.

In our VO example, it is clear that VO participants need to access both static and dynamic information and data. This access needs to be real-time, either via notifications or on a per-query basis. In the rest of the paper we highlight areas of distributed computing and distributed systems research that could be leveraged (and extended) to enable those two modes of operation.

# 4 Grid Information Dissemination

In this section we discuss issues of scalable delivery of dynamic information about the state of a Virtual Organization.

## 4.1 Publisher/Subscriber Systems

The example scenario in the previous section shows that an important question in Grid computing is the scalable and timely propagation of *dynamic information* concerning both application data and Grid resources. This information is critical for enabling large-scale VOs, and as a result new ways of achieving collaborative science. Because of its HPC heritage, Grid computing started from the perspective of tightly coupled and centrally controlled resources. As Grids grow in scale, the trend is to assemble loosely-coupled autonomous components interconnected over wide-area networks. It is known that *event-based* distributed systems are scalable ways of managing such ensembles. The asynchronous, heterogeneous, and collaborative aspects of the scenario described in the previous section suggest that event-based interaction is a natural abstraction for enabling VOs. Implementing a scalable event-based system is difficult, not only due to the sheer number of clients in the system, but also due to networking limitations. Assumptions of low latency, abundant bandwidth, reliable connectivity, and centralized control, which are valid within a local-area network, do not hold on the wide-area.

A class of event-based systems that are ideally suited for the dissemination of Grid information is that of *publisher/subscriber* systems (or *pub/sub* for short): systems that interconnect information providers to information consumers in distributed environments. A broad pub/sub paradigm is that of subject-based routing (and its subtle variations: "group-based", "channel-based", and "topic-based"). Publishers label each event with a subject name, and subscribers receive all events with desired subjects. A number of subject-based systems have been successfully implemented [38, 69, 48, 20, 63]. A more recent type of pub/sub systems are "content-based" ones. In those systems, each event follows a schema that defines the event's content as a set of attributes with different types. Subscriptions are then defined as predicates over event attributes. A taxonomy of several existing systems is presented in [12] and it shows the following trend: Moving away from centralized design, pub/sub systems are increasingly designed with an overlay network of *information brokers* that route events between publishers and subscribers.

For instance, imagine a Grid resource sensor (e.g. as provided by [75] or [70]) generating events about the load of compute resources. An application-level scheduler, acting as an agent for a user, could then register for events that match the following predicate `(hostname == bh.sdsc.edu)` and `(load < 2.50)` (assuming that the load of a compute resource is represented as a floating point number). By registering for several such events for several resources, the scheduler can then make real-time decisions and select appropriate resources for the application.

Similarly, in a system such as the GriPhyn VO, users wish to be notified of new measurement data available from scientific instruments as is produced, or from new derived data as it is computed by a member of the VO. In these cases, publishers would be components of the Grid storage infrastructure: the Data Grid. They could send events about new data items becoming available. This could be very useful for critical data that is periodically produced, rather than VO members doing periodic queries which can be too frequent or too infrequent, leading either to prohibitive network load or to users not becoming aware of new data items in a timely fashion. Such considerations are the usual motivations for using event-based systems. Events need to be appropriately routed to users and to user agents so that new computations, data analyses, and data derivations can be triggered.

In the early days of Grid computing, these scenarios were implemented with lookups of centralized (or sometimes replicated) databases. The scalability limitations of this approach were known and became a major impediment as Grid computing gained in popularity. Recent advances in MDS-2 [21] and developments of the Open Grid Software Architecture (OGSA) [30] allow for more scalable and powerful architectures that allow for extensive caching and replication of information to support high query rates. Also, interesting first steps have been taken to implement Grid event distribution [46] and there is a "Grid Notification Framework" research group as part of the Global Grid Forum [35]. Therefore, it is clear that the Grid community has acknowledge the importance of notification and event distribution and the required software architecture is being developed. We argue that (i) several results from pub/sub research should be leveraged by that architecture; (ii) Grid computing poses new challenging questions in the area of pub/sub systems. We discuss those points in the next section.

### 4.2 Publisher/Subscriber and the Grid

Subject-based pub/sub systems present two major advantages. The matching of events to subscriptions is a simple lookup in a table, and scalability is achieve by straightforward multicasting, where a multicast group is created for each subject. The power of the content-based paradigm comes with the loss of those two convenient characteristics. Event matching is more involved as it requires evaluations of predicates over event content. A simple multicast technique where a multicast group is created for each matching set of events produces a prohibitive number of groups [51]. This is the main challenge that motivates most of the recent research work on content-based systems. Our goal is to see what results can be leveraged for Grid computing, and what additional questions are raised.

A number of content-based pub/sub systems have been implemented and evaluated [65, 76, 66, 39]. One of the main design issues is the choice of the topology of the overlay network formed by the event brokers. For instance, the work on the SIENA [66] system studied three possible topologies: client/server, acyclic peer-to-peer, and general peer-to-peer. Each topology has its advantages and drawbacks [13, 14]. The authors make the observation that different topologies can be combined to form *hybrid* topologies where the overlay network is partitioned in possibly hierarchical subnets with different internal topologies. The question is then: **What overlay topology of event brokers is appropriate for VOs?** The choice depends on the size of the VO and the usage patterns of VO members. In addition, VOs will undoubtedly exhibit characteristics that are not present in the domains previously targeted by pub/sub systems. In fact, it is most likely that no single topology is best in all cases. However, it is worthwhile to examine particular important domains, i.e. scientific communities, and discover which overlay network topologies are the most appropriate. For instance, this could be done for the scientific communities that are targeted by GriPhyn [4] and, in spite of not providing a general solution for Grid computing, would have a tremendous impact on disciplinary science.

Another critical issue for providing a pub/sub system is the specification of the *subscription predicates*, that is the ways in which subscribers can define the events in which they are interested. The trade-off is that richer predicates (expressiveness) are more difficult to deploy (scalability). One extreme is the purely subject-based approach, and another is a content-based approach that allows any boolean expressions over event content, including past events (i.e. taking into account event history). A common model for predicates has been that of a conjunction of simple tests over event attributes [1]. Researchers have focused on defining rich languages to describe subscription predicates [49]. It is then the choice of the pub/sub system designers to choose which subset of those languages to use. For instance, the work in [14] restricts predicates to *filters* and *patterns*, which enables several optimizations of notification selection within the event service. Another notable example is the work in [7] where the authors define an *Information Flow Graph* model that extends the simple conjunction of simple tests over event attributes. The question is: How to determine the appropriate trade-offs for enabling VOs on the Grid? The first step is to analyze the data schemas being developed within the Grid computing community, e.g. for Grid information services. One must understand the ways in which this data is to be used. This can be achieved by collecting logs from current VO efforts and analyzing them for trends and patterns. Building on the aforementioned pub/sub results, one can then make an informed decision for proposing predicate languages for Grid information that make appropriate expressiveness/scalability trade-offs. It is likely that different components of Grid information will require different levels of expressiveness (e.g. resource information vs. application data information). It would then be interesting to answer the question: **What are the benefits and challenges of simultaneously supporting different classes of predicate specifications within a Grid pub/sub infrastructure?**

Previous work on pub/sub systems and routing algorithms makes several assumptions about the underlying network of subscribers and about patterns of subscription matches. In what follows we review common assumptions and give insight regarding how appropriate they are to Grid computing.

### 4.2.1 Selectivity and Regionalism

Two related assumptions that make it possible to devise efficient event routing algorithms are *selectivity* and *regionalism*, using the terminology in [51]. High selectivity means that subscriptions are selective enough that the probability of a match is low. High regionalism means that matches for an event are non-uniformly distributed over the entire subscriber network. In other words, sets of subscribers that are geographically close tend to be interested in similar events. For instance, the results presented in [6] simulate "locality of interest" by mapping different subnets of a wide-area network to distinct distributions of interest values. The authors in [51] make the observation that if selectivity is low **and** regionalism is low, then most events are of interest to most subscribers, and an event broker in the overlay network should route most events. In this case, one should use a simple *flooding* algorithm, although this solution does not scale if subscribers are to receive events reliably. The question is: **What levels of selectivity and regionalism are to be expected for VOs on the Grid**? Let us consider the two types of events that we identified in Section 3: Grid resource information events and Data Grid events.

It is rather difficult to foresee the levels of selectivity and regionalism for events related to Grid resource information. For instance, suppose most software packages used for data derivations are portable and installed on most resources. Then, all VO members who wish to perform data derivations may be interested in most information concerning all available compute resources within the VO. This leads to a low selectivity / low regionalism scenario which does not scale. A great research opportunity is then to understand the specific needs for Grid resource information within VOs and make recommendation both for event routing protocols and for resource usage policies.

Data Grid events, i.e. events about new data becoming available, will exhibit a variety of regionalism levels in most VOs. For instance, it is reasonable to assume that scientists working on the same components of a specific problem are likely to be clustered. For instance, different research teams in different participating sites in the GriPhyn project are probably interested in the same type of data being produced. Events related to that type of data then exhibit high regionalism and high selectivity. However, as collaboration among universities increase thanks to the establishment of VOs, the regionalism could potentially become lower as "virtual" research teams are established. Another reasonable assumption is that certain data is probably relevant to most members of the VOs, e.g. measurement data coming from a unique scientific instrument. Events regarding such data will therefore exhibit low regionalism and selectivity. Once again, the diversity of Grid applications leads to a mixed population of events. The way to proceed is then to: (i) examine a large VO; (ii) gather information about the behavior and requirements of VO members; (iii) instantiate a realistic model for the mix of events; (iv) quantify values for regionalism and selectivity.

#### 4.2.2 Dynamic Subscriptions

Another assumption commonly made in previous work is that the set of subscriptions is relatively static. This means that events are published at a rate orders of magnitudes higher than the rate at which subscriptions enter and leave the system. For instance, the work presented in [6] describes an efficient event matching and routing scheme that uses a Parallel Search Tree (PST) data structure. The PST encodes all subscriptions in the system and is replicated on all event brokers in the overlay network. Each time a new subscription is added (or removed) from the system, there is potentially a need for a global update of the replicated PST structure. In [51] it is noted that "many systems are likely to experience a flux of subscriptions". This is true for classes of subscriptions in Grid VOs. For instance, when a user wishes to perform a data derivation, he will probably start an agent that will subscribe to Grid resource events in order to select appropriate resources. Once the derivation is complete, the user agent will un-subscribe and shut down. Other subscriptions will have longer range, e.g. subscriptions that track new data being produced by an on-line scientific instrument. Dynamic subscriptions often require that multicast groups used for event routing be reconstructed periodically. [51] contains a discussion of which routing algorithms are more resilient or sensitive to dynamic subscriptions. The (expected) conclusion is that a "hybrid" solution will be the best solution. This is a rather general statement, and the question is: **What hybrid routing algorithm and topology will be resilient to the dynamics of VO subscriptions**?

This discussion has revealed several concerns relating to the practical applicability of existing pub/sub results to the Grid. We conclude that none of the assumptions discussed above hold for all Grid events, but that some hold for classes of events. In addition, it is difficult at this time to precisely quantify requirements for a Grid event system in a "typical" VO. However, it is possible to gather information today on cutting-edge efforts such as GriPhyn [4]. In the next section, we provide perspectives on what we believe is a crucial point to investigate in order to build on existing pub/sub results: *quality of service for Grid event delivery.*

### 4.3 QoS for Event Delivery

Similar to the trade-off between expressiveness and scalability described at the beginning of the previous section, there is a clear trade-off between Quality of Service (QoS) for event delivery and scalability. QoS for event delivery can come in various forms and be specified by several strong or weak guarantees. Examples include statements such as: "every event is eventually delivered to all interested subscribers" or "every event is delivered to most interested subscribers in under 1 second". Existing event-based pub/sub systems specify such QoS requirements. For instance, SIENA [66] provides a *best effort* service, meaning that race conditions induced by network latencies and out-of-order messages are not prevented. The two most common QoS specifications in existing systems are whether events are delivered reliably or unreliably, and whether events are delivered in order or if they may be out of order. We claim that the diversity of activities in a Grid VO will require more flexible QoS semantics.

We believe that supporting different degrees of QoS requirements for different classes of events makes it possible to overcome the difficult issues that we have identified in Section 4.2. For instance, consider Data Grid events. Those events should be reliably delivered so that users, or user agents, do not "miss" interesting new pieces of data. However, those messages may not have tight timeliness requirements: a piece of data that has just been generated will probably not go away. Therefore, those messages can tolerate delayed and out-of-order delivery. On the other hand, events related to Grid resource information can implement a simple "best-effort" paradigm with various levels of lossiness. Consider a user agent (e.g. an application-level scheduler) that makes decisions for resource selection. Arguably, better decisions can be made with up-to-date information about resources. However, if information is missing or out-of-date, there are several strategies that can be used for making decisions. For instance, one could decide to not use any resource on which one does not have sufficiently up-to-date information. Alternatively, depending on the Grid information values and their staleness, one can decide to select resources for running a computation according to ad-hoc heuristics. In fact, current Grid implementation efforts typically make provisions for such deficiencies in Grid information. The event system could then afford to "drop" a fraction of certain events without a devastating impact on the VO. Therefore, an interesting question is: **What are event routing schemes that can exploit a variety of QoS requirements for different classes of events to ensure scalability in VOs?**

This question has already been identified and partially addressed by previous work on pub/sub systems. Indeed, in [78] it is noted that pub/sub systems typically offer "limited and low-level options for quality of service". The authors propose an *event stream interpretation* model so that every subscriber can specify a spectrum of delivery QoS semantics. The rationale is that the pub/sub system can then implement efficient and scalable protocols that exploit weaker QoS requirements on some event streams to do message routing. We view this avenue of research as very promising for Grid event systems.

# 5 Retrieving Data and Information

In the previous section we have discussed systems that deliver events to subscribers. This is a flexible way to allow components to interact in large-scale, wide-area environments such as the ones that will be spanned by VOs. However, this does not imply that all interoperation in the Grid can (or should) be done via such systems. In fact, there is a clear need for enabling *queries*. First, some Grid resource information is static. Second, users or user agents may want to perform queries to identify all (or most) resources that fit some criteria. For instance, one may want to find all compute resources on which some specific software is currently installed and that provide at least 1GB of RAM. Also, a user needs to issue queries to discover relevant application data that is available in Grid storage devices. As we have seen in Section 4.2, events can be generated for periodic data creations. However, in a realistic VO, we also expect users and user agents to generate queries to the Data Grid to discover and retrieve archived data. The goals for Grid computing are no different from other areas: to make *discovery* and *retrieval* efficient and scalable.

A number of relevant protocols and mechanisms have been explored in many related contexts such as distributed databases, Web caching, content distribution networks, and distributed file systems. Key concepts are shared among those efforts. However, it is difficult to compare protocols from different domains and understand all the trade-offs that are relevant to a particular problem, such as Grid computing. Consider for instance the concept of data replication, which is commonly used to increase data availability and reduce data retrieval latency. As seen in [73], truly understanding the connection with replication protocols used in distributed databases and those used in distributed systems is a non-trivial task. If one can unify protocols from different communities, the potential pay-off is better and more robust protocols. For instance, many researchers have successfully explored database replication protocols that utilize concepts from distributed systems (e.g. using group communication [67]).

A number of research works in the Grid community have recently recognized and explored fundamental connections between Grid computing and work in the distributed computing and the distributed systems areas. Further work will be needed to develop adequate distributed algorithms and protocols for enabling discovery and retrieval of data and information within VOs. In the next section we review key areas for such developments.

## 5.1 Discovery

One of the challenges for VOs is that they must implement robust and fast data/information discovery. The problem is the following. A number of participants store information (e.g. the clock frequency of a host, or the location of a particular data item). Other VO participants submit possibly complex queries over that information, and should experience low response time for those queries. In the early years of Grid computing, this was achieved via centralized services that contained all stored information. An example is the Condor matchmaker [58]. While efficient for a local-area network, this system breaks down for large VOs as it is both a performance bottleneck and a single point of failure. Similarly, the Globus Toolkit's Monitoring and Discovery Service (MDS) [27] was initially designed as a centralized way to obtain Grid information (via an LDAP server). Later designs in MDS-2 have moved to a decentralized approach where Grid information is stored and indexed by index servers that communicate via a registration protocol [21]. Users can then query directory servers. At the moment, the assignment of content to servers and the overlay topology of those servers is done in an ad-hoc fashion. Nevertheless, these recent developments make it possible to address distributed computing questions in a practical and concrete context.

Based on recent advances in the area of distributed systems, Grid researchers are investigating how scalable discovery mechanisms can be implemented using a *peer-to-peer* architecture [41]. Every participant (organization or individual) in a VO must have full control of which information is published about its local resources. We therefore assume that every participant in a VO maintains one or more servers, or *peers*, that provide access to local resource information. Those peers may join or leave the system at any time. One may expect certain VOs to be more or less dynamic, but it is currently too early to make any statement about what could be "typical". In this discussion we ignore issues about the construction and maintenance of the overlay network as we surmise they are probably no different for Grid computing than for other peer-to-peer systems.

A question is then whether previous work and results on discovery in dynamic, self-organizing peer-to-peer networks can be utilized and extended. As in our discussion of pub/sub systems, we make the distinction between the discovery of data locations (e.g. find the file 'foo') and of Grid resources (e.g. find a CPU with some desired clock-rate).

### 5.1.1 Discovering data

A number of recent efforts such as CAN [60], Chord [68], Pastry [62], and Tapestry [77] provide powerful mechanisms for locating data in peer-to-peer networks. The goal is to locate a particular data item given a key, or name, which is used for indexing. In the context of the Data Grid, this is often stated as finding a physical data file given a logical file name. The aforementioned systems use clever routing and indexing schemes to reduce the latency of the search process (e.g. [56]). Those systems can conceivably be utilized to discover data (and data replicas) efficiently in a Data Grid. In fact, it may be that a system like Oceanstore, which uses Tapestry, could provide a good solution. Our goal here is not to argue which one of those systems is the most appropriate, but rather to point to interesting research questions. Such a question is: **are there any characteristics of Grid VOs that can be exploited for optimizing data discovery?** In the scenario we presented in Section 3, the data is not produced and consumed by largely unrelated individuals (e.g. such as the Web). Rather, the data access patterns result from specific scientific activities. One may then wonder if there exists data sharing patterns that could be exploited for reducing the latency of data discovery. If so, then there is a great opportunity for extending currently available data discovery systems and specializing them for scientific data on the Grid.

Even though there are only a few VOs currently in existence, work in [43] makes a very interesting hypothesis: scientific communities sharing data tend to behave like *small-world* networks [72]. Small-world networks have arisen in contexts as diverse as social networks, the World Wide Web, and neuro-networks. They exhibit two fundamental characteristics: (i) a small average path length; (ii) a large clustering coefficient that is independent of the network size. The clustering coefficient quantifies how many of a node's neighbors are connected to each other. Intuitively, a small-world network consists of a loosely connected network of almost fully connected sub-networks. This hypothesis is substantiated by examining an actual VO that involves physicists in over 18 countries [23]. Using logs from the data access system over different time-periods, it was possible to construct the following graph. Each VO participant is a node in the graph. There is an edge in between two nodes if the two participants shared at least one file during the given time period. It was found that the resulting graphs indeed exhibit many characteristics of small-world networks. Note that the clusters in the graph do not reflect any geographical clustering of the participants, but rather commonalities of interest.

A key idea is then that having some sort of structure on the network should make it easier to develop algorithms and protocols for efficient data management. In this context, [43] asks one of many relevant questions: **What data discovery scheme can take advantage of the**

**small-world characteristics**? Assuming that the small-world topology is known, they propose a data discovery strategy that uses the following principle: “Data location information is propagated aggressively within clusters, while inter-cluster search uses request-forwarding techniques”. The intra-cluster propagation is done via a standard gossiping mechanism that is used to maintain Bloom Filters. Eventually, all peers within a cluster know the location of “most” data items relevant to other peers in the cluster.

A crucial question raised by the work in [43] (but not answered) is: **What protocols will lead to a self-organizing overlay network that reflects the small-world properties of a VO**? In other words, how does the overlay network automatically discover the topology of the small-world network and self-organize to lower the number of necessary hops per requests? [43] presents an interesting discussion of this issue, which opens up an exciting avenue for future research.

Early data replica location services used a centralized approach [2, 8] which of course causes scalability problems. The new generation of data replica location services for Data Grids is currently being developed, implemented and deployed [17, 61]. The new system provides a flexible way to arrange index brokers in various overlay topologies. This is a critical time for developing appropriate distributed algorithms and protocols that will scale and be efficient in VOs.

### 5.1.2 Discovering Grid resources

The data location systems mentioned in the previous section use *names* as their sole search criteria. It is assumed that every data item is assigned a unique identifier that is used for indexing and routing. This is not amenable to *Grid resource discovery*. Indeed, Grid resource discovery services need to answer requests that specify desired sets of attribute values. One could envision a system by which attributes and attribute values are mapped to logical names that can be used for clever routing by the aforementioned systems. This is of course possible for locating data that is entirely described by a single attribute, e.g. a logical file name (see previous section). However, the mapping would be highly sensitive to dynamic attributes such as those describing compute resources. Besides, this would still limit the type of queries that could be performed. Therefore, in general a request-routing scheme for Grid resource discovery cannot utilize the same strategies as in [60, 68, 62, 77].

Among the first works in the Grid computing community to address these questions is [41]. In that paper, Iamnitchi and Foster evaluate 4 simple request-forwarding strategies. Among these, 3 allow peers to forward requests based on past experience. Their evaluation makes a number of assumptions concerning the distribution of resources among peers (how many resources are managed by a peer), the resource frequency (how common is a resource), and the distribution of queries. Nevertheless, that paper sets the stage for answering the following question: **What request-forwarding scheme is appropriate for resource discovery on the Grid?**

Another strongly related question is: **What is the appropriate data model, and associated query language, for enabling Grid resource discovery?** Grid Information Services really implement a distributed database. One approach for describing the data is to use a *hierarchical* model. This is the approach which is currently in place as Grid Information Services have been built on top of directory services. An active debate in the Grid community is whether those systems and the hierarchical model will provide sufficient performance and expressiveness. An alternate solution is to use a *relational* data model, which arguably is more difficult to implement and scale, but allows for more expressiveness with a relational query language. In that respect, the design of Grid Information Services could face certain of the challenges encountered in the area of distributed databases. The question of consistency is probably not critical as most Grid resource information follows a one-writer many-readers model.

Lastly, let us note that this last issue overlaps with our discussion of pub/sub systems in Section 4. For instance, the subscription predicate language for Grid events should probably be a subset of the Grid resource discovery query language. This leads to the notion of an integrated Grid Information Service that supports both queries and subscriptions. Many Grid researchers are working on designs for such an integrated system. All the research questions that we have identified so far will be of relevance in that context.

### 5.2 Replication

Data replication is a well-known technique used in various distributed storage systems for improving performance and availability. In this section we focus on the problem of storing and replicating application data in a Data Grid that is used by participants in a VO. Once again, an important focus in Grid computing today is on distributed scientific communities who wish to perform large amount of data analyses on increasingly large datasets. Current efforts [4, 52, 25] are already building Data Grids that are expected to process and store petabytes of data each year. The Data Grid that is being deployed as part of GriPhyn is hierarchical and organized in tiers [59]. This hierarchical structure is reminiscent of Web caches. Grid researchers are also looking at peer-to-peer architectures for data storage, as a number of such experimental systems have been proposed and are being developed (e.g. Oceanstore [47]).

Data replication has been the focus of an impressive number of research works in many areas [73, 54, 44, 55, 57, 71, 16]. A typical research question, which is addressed in most of those works, is "how does one ensure consistency among diverging replicas in a scalable way?" [37, 45, 26]. However, current work on Data Grids does not place a large emphasis on that problem. This is due to the specific nature of many scientific communities. In many scientific collaborations, data is either generated from instruments or derived from measured data. It is then annotated with domain-specific metadata and published to the community. From that point on, the file is almost never modified (even though it can be replaced by a new version). Therefore a model that assumes that files are *read-only* and that they can be uniquely identified by their version numbers is currently considered to be realistic for data in Grid VOs [17]. This is certainly not true for all foreseeable applications, and is probably not true of metadata associated with application data. Therefore, it is expected that data consistency issues will require increasing attention, as envisioned for instance in [24].

The simplifying "read-only assumption" makes it possible to initially address the following question: **What are efficient algorithms for creating and disseminating replicas in a Data Grid?** That question has been asked and partially answered in [59]. In that paper, 5 caching/replication strategies, some of them inspired by the Web caching literature, have been proposed and evaluated. The advantage of dynamic caching and replication is that it automatically creates (and deletes) replicas according to changes in data access patterns. A wide spectrum of results from distributed computing could be potentially utilized and evaluated for automatic replica creation. Another related issue, which we have mentioned in Section 5.1.1, is to better understand the relationship of files in scientific collaborations so that data access patterns can be exploited by distributed protocols and algorithms. This is becoming increasingly feasible as VOs are deployed and real trace data is collected.

## 6 Conclusion

Grid computing is broad in its domain of application and raises research questions that span many areas of distributed computing, and of computer science in general. In this paper we have opted for providing detailed descriptions of a few issues that we feel are particularly interesting, can largely benefit from cross-fertilization with the distributed computing research community, and have already been the object of preliminary work in Grid computing. Namely, we discussed issues pertaining to the discovery and dissemination of information and data, both static or dynamic, within Virtual Organizations. We focused on two different models: subscription-based and query-based.

Other authors would likely have chosen to discuss other issues. We provide here a brief and non-exhaustive glimpse of what those issues might be.

A fundamental concern for Grid computing is *security*. The current Grid Security Infrastructure (GSI) [11] supports single sign-on, delegation, and user-based trust relationships, each of which raises a number of challenging questions. Issues relating to policy have only being touched upon briefly to date. Another issue is that of *scheduling* for distributed applications. Application scheduling has been an active field of research for decades and the Grid poses a number of new challenges that have been identified and partially addressed [9]. A critical issue is to schedule applications that combine intensive computation with the use of data stored (and replicated) in emerging Data Grids. For instance, one question is to determine in which scenarios coupled [15] and decoupled [42] scheduling approaches are appropriate. Another issues is to ensure that scheduling agents acting of behalf of their users cooperate in order to avoid "herd behavior". Although this problem has been identified in the context of the Grid early on, it has not been addressed. This ties into the notion of a *Grid economy*. Several authors argue that policies for Grid usage should be derived from economical models that are based on a commodity market. Early work in the Grid community has already evaluated a few hypotheses for defining viable *Grid economy* models [74].

The question of *co-scheduling* is that of ensuring that an application can reserve and utilize several resources simultaneously at a given time. This is a difficult problem which is important for many Grid users and applications. More generally, ways for VO participants to achieve *agreement* are needed. Indeed, *fault-tolerance* becomes a critical issue with increases in scale: large VOs mean that the probability of failure of individual components becomes significant. Agreement protocols have not been adopted by the Grid community so far. They have not yet been important for application writers, because few applications require much in the way of fault-tolerance; simply restarting a failed computation is sufficient. Also, even with an increase in VO scale, there are more scalable methods of fault-tolerance (such as rollback recovery) that would make more engineering sense to use than agreement for most applications. However, the services comprising the Grid software infrastructure itself are long-running and are key for sustaining VO activities. Although inexpensive approaches like eventual update combined with randomized scheduling have proven sufficient so far (e.g. see [3]), we believe that a number of Grid services are likely candidates for agreement protocols.

The Grid is currently being build as a concerted effort among many institutions and is already supporting leading scientific applications. In this paper we have identified several differences between the Grid and other distributed computing models and systems. We argued that those differences motivate new research questions. As more and more VOs are deployed, it will be possible to gather very large amounts of trace data concerning the social and technical interactions among VO participants. Mining that data will undoubtedly reveal crucial features of the nature of scientific collaborations that can be exploited to design appropriate distributed protocols and algorithms. Furthermore, it will be possible to construct increasingly realistic models that can be used for the evaluation of those protocols and algorithms. Finally, the Grid will provide several concrete platforms for the validation of research results in real-world scenarios. We hope that this paper provides evidence that Grid computing is an exciting area, and that it provides many opportunities for researchers in distributed computing and distributed systems to tackle new problems and to evaluate their solutions.

## Acknowledgements

The author is extremely grateful to Ian Foster and Keith Marzullo for their insightful comments and suggestions for improvements.

## References


[1] M. Aguilera, R. Strom, D. Sturman, M. Astley, and T. Chandra. Matching Events in a Content-based Subscription System. In *Proceedings of the 18th Annual ACM Symposium on Principles of Distributed Computing (PODC 1999)*, pages 53–61, Atlanta, Georgia, May 1999.

[2] B. Allcock, J. Bester, J. Bresnahn, A. Chervenak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnel, and S. Tuecke. Data Management and Transfer in High-Performance Computational Grid Environments. *Parallel Computing*, 2002. to appear.

[3] A. Amoroso, K. Marzullo, and A. Ricciardi. Wide-Area Nile: A Case Study of a Wide-Area Data-Parallel Application. In *Proceedings of the 18th International Conference on Distributed Computing Systems (ICDCS), Amsterdam, Netherlands*, pages 506–515, May 1998.

[4] P. Avery and I. Foster. The GriPhyN Project: Towards Petascale Virtual Data Grids. `http://www.griphyn.org`, 2001.

[5] P. Avery, I. Foster, R. Gardner, H. Newman, and A. Szalay. An International Virtual-Data Grid Laboratory for Data Intensive Science. `http://www.griphyn.org`, 2001.

[6] G. Banavar, T. Chandra, B. Mukherjee, J. Nagarajarao, R. Strom, and D. Sturman. An Efficient Multicast Protocol for Content-Based Publish-Subscribe Systems. In *Proceedings of the 19th IEEE International Conference on Distributed Computing Systems (ICDCS)*, 1998.

[7] G. Banavar, M. Kaplan, K. Shaw, R. Strom, D. Sturman, and W. Tao. Information Flow Based Event Distribution Middleware. In *Proceedings of the 19th IEEE International Conference on Distributed Computing Systems, Workshops on Electronic Commerce and Web-based Applications*, 1999.

[8] C. Baru, R. Moore, and M. Rajasekar, A. Wan. The SDSC Storage Resource Broker. In *Proceedings of CASCON'98, Toronto, Canada*, Nov. 1998.

[9] F. Berman. *The Grid, Blueprint for a New computing Infrastructure*, chapter 12. Morgan Kaufmann Publishers, Inc., 1998. Edited by Ian Foster and Carl Kesselman.

[10] F. Berman, R. Wolski, S. Figueira, J. Schopf, and G. Shao. Application Level Scheduling on Distributed Heterogeneous Networks. In *Proceedings of Supercomputing'96*, November 1996.

[11] R. Butler, D. Engert, I. Foster, C. Kesselman, and S. Tuecke. Design and Deployment of a National-Scale Authentication Infrastructure. *IEEE Computers*, 33(12):60–66, 2000.

[12] A. Carzaniga, D. Rosenblum, and A. Wolf. Challenges for Distributed Event Services: Scalability vs. Expressiveness. In *Proceedings of the ICSE'99 Workshop on Engineering Distributed Objects (EDO'99)*, 1999.

[13] A. Carzaniga, D. Rosenblum, and A. Wolf. Interfaces and Algorithms for a Wide-Area Event Notification Service. Technical Report CU-CS-888-99, Department of Computer Science, University of Colorado, Oct. 1999.

[14] A. Carzaniga, D. Rosenblum, and A. Wolf. Achieving Scalability and Expressiveness in an Internet-Scale Event Notification Service. In *Proceedings of the 19th Annual Symposium on Principles of Distributed Computing (PODC 2000)*, pages 219–227, Portland, Oregon, Jul. 2000.

[15] H. Casanova, A. Legrand, D. Zagorodnov, and F. Berman. Heuristics for Scheduling Parameter Sweep Applications in Grid Environments. In *Proceedings of the 9th Heterogeneous Computing Workshop (HCW'00)*, pages 349–363, May 2000.
[16] Y. Chen, R. Katz, and J. Kubiatowicz. Dynamic Replica Placement for Scalable Content Delivery. In *Proceedings of the First International Workshop on Peer-to-Peer Systems (IPTPS 2002)*, March 2002.
[17] A. Chervenak, E. Deelman, I. Foster, A. Iamnitchi, C. Kesselman, W. Hoschek, P. Kunszt, M. Ripeanu, B. Schwartzkopf, H. Stockinger, K. Stockinger, and B. Tierney. Giggle: A Framework for Constructing Scalable Replica Location Service. In *Proceedings of Supercomputing'02*, Nov 2002.
[18] A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, and S. Tuecke. The Data Grid: Towards and Architecture for the Distributed Management and Analysis of Large Scientific Data Sets. *Journal of Network and Computer Applications*, 23(3):187–200, 2000.
[19] Common Information Model, Distributed Management Task Force, Inc. `http://www.dmtf.org/standards/standard_cim.php`.
[20] G. Cugola, E. Di Nitto, and A. Fuggetta. Exploiting an Event-Based Infrastructure to Develop Complex Distributed Systems. In *Proceedings of the 20th International Conference on Software Engineering (ICSE'98)*, Apr. 1998.
[21] 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.
[22] K. Czajkowski, I. Foster, C. Kesselman, V. Sanger, and S. Tuecke. SNAP: A Protocol for Negociating Service Level Agreements and Coordinating Resource Management in Distributed Systems. In *Proceedings of the 8th Workshop on Job scheduling Strategies for Parallel Processing*, July 2002.
[23] The D0 Experiment. `http://www-d0.fnal.gov`.
[24] D. Düllman, W. Hoschek, J. Jean-Martinez, A. Samar, B. Segal, H. Stockinger, and K. Stockinger. Models for Replica Synchronisation and Consistency in a Data Grid. In *Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing (HPDC-10)*, August 2001.
[25] European Datagrid Webpage. `http://eu-datagrid.web.cern.ch`.
[26] Z. Fei. A Novel Approach to Managing Consistency in Content Distribution Networks. In *Proceedings of Web Caching and Content Distribution Workshop (WCW'01), Boston, MA*, June 2001.
[27] S. Fitzgerald, I. Foster, C. Kesselman, G. von Laszewski, W. Smith, and S. Tuecke. A Directory Service for Configuring High–Performance Distributed Computations. In *Proceedings of the 6th IEEE International Symposium on High Performance Distributed Computing (HPDC-6)*, August 1997.
[28] I. Foster. The Grid: A New Infrastructure for 21st Century Science. *Physics Today*, 55(2):42, February 2002.
[29] I. Foster and C. Kesselman, editors. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers, Inc., San Francisco, USA, 1999.
[30] I. Foster, C. Kesselman, J. Nick, and S. Tuecke. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. Available at `http://www.globus.org`, 2002.
[31] I. Foster, C. Kesselman, and S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International Journal of High Performance Computing Applications*, 15(3), 2001.
[32] I. Foster, J. Vöckler, M. Wilde, and Y. Zhao. Chimera: A Virtual Data system for Representing, Querying, and Automating Data Derivation. In *Proceedings of the 14th International Conference on Scientific and Statistical Database Management, Edinburgh*, July 2002.
[33] Global Grid Forum. `http://www.gridforum.org/`.
[34] Working Group on Grid Information Services at the Global Grid Forum. `http://www.gridforum.org/1_GIS/GIS.htm`.
[35] Research Group on Grid Notification at the Global Grid Forum. `http://www.gridforum.org/1_GIS/GNF.htm`.
[36] Globus Project. `http://www.globus.org`.
[37] J. Gray, P. Helland, O. O'Neil, and D. Shasha. The Dangers of Replication and a Solution. In *Proceedings of ACM SIGMOD*, pages 173–182, 1996.
[38] O. M. Group. CORBA Services: Common Object Service Specification. Technical report, Object Management Group, July 1998.
[39] The Gryphon Project. `http://www.research.ibm.com/gryphon`.

[40] International Symposium on High Performance Distributed Computing (HPDC). `http://www.hpdc.org`.
[41] A. Iamnitchi and I. Foster. On Fully decentralized Resource Discovery in Grid Environments. In *Proceedings of the International Workshop on Grid Computing, Denver, Colorado*, November 2001.
[42] A. Iamnitchi and I. Foster. Decoupling Computation and Data Scheduling in Distributed Data-Intensive Applications. In *Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing (HPDC-11)*, July 2002.
[43] A. Iamnitchi, M. Ripeanu, and I. Foster. Locating Data in (Small-World?) Peer-to-Peer Scientific Collaborations. In *Proceedings of the First International Workshop on Peer-to-Peer Systems, Cambridge, Massachusetts*, March 2002.
[44] J. Kangasharju, J. Roberts, and K. Ross. Object Replication Strategies in Content Distribution Networks. In *Proceedings of Web Caching and Content Distribution Workshop (WCW'01), Boston, MA*, June 2001.
[45] A.-M. Kermarrec, A. Rowston, M. Shapiro, and P. Druschel. The IceCube approach to the reconciliation of divergent replicas. In *Proceedings of the 20th Annual ACM Symposium on Principles of Distributed Computing (PODC 2001)*, August 2001.
[46] C. Krintz and R. Wolski. NWSAlarm: A Tool for Accurately Detecting Degradation in Expected Performance of Grid Resources. In *Proceedings of CCGrid'01*, May 2001.
[47] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao. OceanStore: An Architecture for Global-Scale Persistent Storage. In *Proceedings of the Ninth international Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2000)*, 2000.
[48] S. Maffeis. iBus: The Java Intranet Software Bus. Technical report, SoftWired AG, Zuric, Switzerland, Feb. 1997.
[49] M. Mansouri-Samani and M. Sloman. GEM: A Generalized Event Monitoring Language for Distributed Systems. *IEE/IOP/BCS Distributed Systems Engineering Journal*, 4(2):96–108, June 1997.
[50] Network for Earthquake Engineering Simulations. `http://www.eng.nsf.gov/nees`.
[51] L. Opyrchal, M. Astley, J. Auerbach, G. Banavar, R. Strom, and D. Sturman. Exploiting IP Multicast in Content-Based Publish-Subscribe Systems. In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC 2001)*, pages 219–228, 2001.
[52] Particle Physics Data Grid. `http://www.ppdg.net`.
[53] C. Partridge. Data Communications vs. Distributed Computing, 2000. Invited talk at PODC 2000.
[54] K. Petersen, J. Spreitzer, D. Terry, M. Theimer, and A. Demers. Flexible Update Propagation for Weakly Consistent Replication. In *Proceedings on the 16th ACM Symposium on Operating Systems Principles (SOSP-16), Saint Malo, France*, 1997.
[55] G. Pierre, I. Kuz, M. van Steen, and A. Tanenbaum. Differentiated Strategies for Replicating Web Documents. *Computer Communications*, 24(2):232–240, 2000.
[56] C. Plaxton, R. Rajaraman, and A. Richa. Accessing Nearby Copies of Replicated Objects in a Distributed System. In *Proceedings of the Symposium of Parallel Algorithms and Architectures (SPAA'97)*, pages 311–320, June 1997.
[57] P. Radoslavov, R. Govindan, and D. Estrin. Topology-Informed Internet Replica Placement. In *Proceedings of the Web Caching and Content Distribution Workshop (WCW'01), Boston, MA*, June 2001.
[58] R. Raman, M. Livny, and M. Solomon. Matchmaking: Distributed Resource Management for High Throughput Computing. In *7th IEEE International Symposium on High Performance Distributed Computing (HPDC-7)*, July 1998.
[59] K. Ranganathan and I. Foster. Identifying Dynamic Replication Strategies for a High Performance Data Grid. In *Proceedings of the International Workshop on Grid Computing, Denver, Colorado*, November 2001.
[60] S. Ratsanamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A Scalable Content Addressable Network. In *Proceedings of SIGCOMM 2001*, 2001.
[61] M. Ripeanu and I. Foster. A Decentralized, Adaptive Replica Location Mechanism. In *Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing (HPDC-11)*, July 2002.

[62] A. Rowstron and P. Druschel. Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems. In *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), Heidelberg, Germany*, pages 329–350, Nov. 2001.

[63] A. Rowstron, A.-M. Kermarrec, M. Castro, and P. Druschel. SCRIBE: The Design of a Large-Scale Event Notification Infrastructure. In *Proceedings of the Third International Workshop on Networked Group Communication*, pages 30–43, 2001.

[64] The International Conference for High Performance Computing and Communications (SC). `http://www.supercomp.org`.

[65] B. Segall and D. Arnold. Elvin has left the building: A publish/subscribe notification service with quenching. In *Proceedings of AUUG'97, Brisbane, Australia*, Sept. 1997.

[66] The SIENA Project. `http://www.cs.colorado.edu/users/carzanig/siena/`.

[67] I. Stanoi, D. Agrawal, and A. Abbadi. Using Broadcast Primitives in Replicated Databases. In *Proceedings of the International Conference on Distributed Computing Systems (ICSDS'98)*, pages 148–155, Amsterdam, The Netherlands, May 1998.

[68] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In *Proceedings of SIGCOMM 2001*, 2001.

[69] I. Sun Microsystems. Java Distributed Event Specification. Technical report, Sun Microsystems, Inc., Mountain View, CA, U.S.A., Nov. 1998.

[70] B. Tierney, W. Johnston, B. Crowley, H. Hoo, C. Brooks, and D. Gunter. The NetLogger Methodology for High Performance Distributed Systems Performance Analysis. In *Proceedings of 7th IEEE International Symposium on High Performance Distributed Computing (HPDC-7)*, July 1998.

[71] A. Venkataramani, P. Weidmann, and M. Dahlin. Bandwidth Constrained Placement in a WAN. In *Proceedings of the 19th Annual ACM Symposium on Principles of Distributed Computing (PODC 2000)*, pages 53–61, 2000.

[72] D. Watts. *Small Worlds. The Dynamics of Networks between Order and Randomness.* Princeton University Press, Princeton, New Jersey, U.S.A., 1999.

[73] M. Wiesmann, F. Pedone, A. Schiper, B. Kemme, and G. Alonso. Understanding Replication in Databases and Distributed Systems. In *Proceedings of the 20th International Conference on Distributed Computing Systems (ICDCS 2000)*, 2000.

[74] R. Wolski, J. Plank, J. Bervik, and T. Bryan. Analyzing Market-based Resource Allocation Strategies for the Computational Grid. *International Journal of High-performance Computing Applications*, 15(3), 2001.

[75] R. Wolski, N. Spring, and J. Hayes. The Network Weather Service: A Distributed Resource Performance Forecasting Service for Metacomputing. *Journal of Future Generation Computing Systems*, 15(5-6):757–768, 1999.

[76] M. Wray and R. Hawkes. Distributed virtual environments and VRML: an event-based architecture. In *Proceedings of the Seventh International WWW Conference (WWW7), Brisbane, Australia*, 1998.

[77] B. Zhao, J. Kubiatowicz, and A. Joseph. Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing. Technical Report UCB/CSD-01-1141, University of California, Berkeley, 2001.

[78] Y. Zhao and R. Strom. Exploiting Event Stream Interpretation in Publish-Subscribe Systems. In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC 2001)*, 2001.