[MH05] *Andreas Mauthe, Oliver Heckmann:* **Peer-to-Peer and GRID Computing;** The 2005 Networking and Electronic Commerce Research Conference (NAEC 2005), Lake Garda, Italien, Oktober 2005, S.

Peer-to-Peer and GRID Computing

Andreas Mauthe[‡], Oliver Heckmann[†]

[‡] Computing Department
Lancaster University
Lancaster, LA1 4WA
UK
andreas@comp.lancs.ac.uk

[†] KOM – Multimedia Communications Lab
TU Darmstadt
64283 Darmstadt
Germany
heckmann@kom.tu-darmstadt.de

Abstract

Peer-to-Peer (P2P) and GRID computing are two concepts that have recently emerged within the distributed systems domain. Both have been successfully applied in a number of areas and have received considerable attention. Frequently the two concepts are linked, though a straight forward comparison is not possible. In this paper the GRID background is described and GRID and P2P are compared to establish where commonalities and differences are.

1. Introduction

Peer-to-Peer and GRID computing are two concepts that are frequently linked and compared. However, since they are basically addressing different areas and come from a different background, this is not a straight forward comparison of features and functionalities. The idea of GRID computing originated in the scientific community and has been motivated by processing power and storage intensive applications [19]. The basic objective of GRID computing is to support resource sharing among individuals and institutions (organizational units), or resource entities within a networked infrastructure. Resources that can be shared are for instance bandwidth, storage and processing capacity, but also data [16, 17]. The resources pertain to organizations and institutions across the world; they can belong to a single enterprise or be in an external resource-sharing and service provider relationship. On the GRID, they form distributed, heterogeneous, dynamic virtual organizations [1].

The GRID is not a completely new concept; it builds (as P2P does) on results of distributed systems research. With the proliferation of the Internet and the development of the Web (together with emerging distributed middleware platforms), it is now possible to build large-scale distributed applications that can span a wide geographical and organizational area. This has been taken advantage of within the P2P and the GRID community more or less at the same time. The GRID has been driven by the science community, which first saw the potential of such systems and implemented them on a wider scale. Application areas here are distributed supercomputing (e.g. physical process simulations), high-throughput computing (to utilize unused processor cycles), on-demand computing (for short-term demands and load balancing), data intensive computing (synthesizing information from data that is maintained in geographically distributed repositories), and collaborative computing [2].

It is important to note that the prime objective of GRID computing is to provide access to common, very large pools of different resources that enable innovative applications to utilize them [13]. This is one of the defining differences between Peer-to-Peer (P2P) and GRID computing. Although both are concerned with the pooling and coordinated use of resources, the GRID's objective is to provide a platform for the integration of various applications, whereas initially P2P applications have been vertically integrated [15].

Many current GRID implementations are based on the Globus ToolkitTM, an open source toolkit [20, 21]. Within the Globus project, a pragmatic approach has been taken in implementing services needed to support a computational GRID. The Open GRID Services Architecture (OGSA) initiative - inspired by the Globus project - develops the GRID idea further and is also concerned with issues that have not been in the focus of the Globus project such as architectural and standardization matters.

In this paper, the main initiatives and concepts driving GRID are introduced. Subsequently, the relationship of P2P and GRID are discussed in more detail. The different approaches are being compared considering and it is discussed if and how these concepts converge.

Section 2 discusses GRID architectural issues. Section 3 introduces the Globus project whereas section for concentrates on Open GRID Services Architecture (OGSA) issues. In section 5 GRID and P2P are discussed in context, and finally section 6 concludes the paper.

2. The GRID Architecture

A computational GRID is more formally defined as "a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities" [2]. Different applications can be

implemented on top of this infrastructure to utilize the shared resources. If different institutions or individuals participate in such a sharing relationship, they form a virtual organization [11]. The architectural concept underlying the GRID facilitates the collaboration across institutional boundaries by proposing a protocol architecture and using standard components that can be used by the different parties entering into a sharing relationship [13].
The GRID architecture can be described by a layered model as shown in Figure 1. The different components are placed within a layered structure depending on their functionality and capabilities.

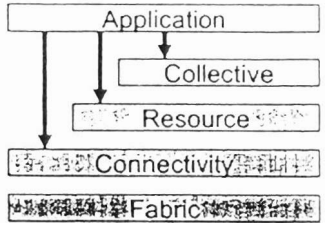

Figure 1: GRID: A layered View

An hourglass model (comparable to the Internet) is used to visualize the different concepts. In this view a small group of core protocols and components build the link between high-level mechanisms and a number of lower level base technologies [22]. The components of the architecture (as depicted in Figure 1) are the Fabric Layer, the Connectivity Layer, the Resource Layer and the Collective Layer [13]. The applications that reside on top of this infrastructure can use the components of the Collective, Resource and Connectivity Layers directly, depending on their requirements.
The Fabric Layer within the architecture makes the resources available that are provided by the different nodes of the GRID for common usage, i.e. it provides common access to resources that are shared within a virtual organization. Resources can be classified as computing resources, storage, network resources, code storage, and directories. [13]. The Fabric Layer implements the resource specific operations for particular resources and offers a unified interface to the upper layers.
The Connectivity Layer hosts the most important communication and authentication protocols that are required for GRID specific communication. It allows the data exchange between the resources located at the Fabric Layer. The communication protocols employed in this context are predominantly from the TCP/IP protocol suite. Security is provided by a public key infrastructure based on special GRID security protocols [23]. On top of the Connectivity Layer, the Resource Layer is responsible for resource management operations such as resource negotiation, resource reservation, resource access and management, QoS control, accounting, etc. The actual resources that are managed, however, are the resources under the control of the Fabric Layer.
The Collective Layer is concerned with the overall co-ordination of different resource groups. It hosts components such as directory services [8], scheduling and brokering services, monitoring and diagnostic services, data replication services, workload management, etc. [13].
Finally, the Application Layer comprises the user applications that are used to realize the virtual organization. Applications are utilizing the services offered by the underlying layers. They can directly access these services.

3 The Globus Project

The Globus project started in 1996; it is hosted by Argonne National Laboratory's Mathematics and Computer Science Division, the University of Southern California's Information Sciences Institute, and the University of Chicago's Distributed Systems Laboratory. It is supported by a number of institutional (e.g. National Computational Science Alliance (USA), NASA, Universities of Chicago and Wisconsin) and industry partners (e.g. IBM and Microsoft) [3]. The project is centered on four main activity areas:
• Building of large-scale GRID applications such as distributed supercomputing, smart instruments, desktop supercomputing teleimmersion.
• Support for planning and building of large-scale testbeds for GRID research but also for productively used GRID systems
• Research into GRID related issues such as resource management, security, information services, fault detection and data management.
• Building of software tools for a variety of platforms (the so call Globus ToolkitTM). These software tools are considered research prototypes.
The Globus ToolkitTM is one of the major results generated by the Globus project. It supplies the building blocks of the GRID, i.e. it provides services and modules required to support GRID applications and programming tools. It is a community based, open architecture, open source set of services and software libraries [1]. The Globus services can be used independently or together to form a supporting platform for GRID applications. The application area is mainly concerned with distributed science applications. The services are programs that interact with each other to exchange information or co-ordinate the processing of tasks.
A number of Globus services deal with resource selection, allocation, and management. Resource in this context is a generic term for everything required to process a task. This includes system resources such as CPU, network

bandwidth and storage capacity. In order to do this, a Resource Selection Service (RSS) is required. It provides a generic resource selection framework for all kinds of GRID applications. Such a service identifies a suitable set of resources by taking into account application characteristics and system status [4].

The Globus Resource Allocation Manager (GRAM) is part of the lowest level of the resource management architecture. It provides resource allocation, process creation, monitoring, and management services. The GRAM service maps requests expressed in the Resource Specification Language (RSL) into commands to local schedulers and computers [5]. An end-to-end management of QoS for different resource types such as bandwidth, CPU, and storage is provided by the General-Purpose Architecture for Reservation and Allocation (GARA) system [18][6]. Dynamic feedback is used among resource managers to coordinate the resource management decisions [7]. The resource management tools build on existing languages, protocols, and infrastructure. Their capabilities depend on the functionality and capacity of the hosting environments they operate in.

A central service within the Globus Toolkit is the Monitoring and Discovery Service (MDS-2). This generic service provides a framework for service and data discovery. The MDS service supplies information concerning system configuration and status information to other services. This includes server configuration, location of data replica, network status, etc. Two protocols are used for accessing and exchanging information in this context; viz. GRIP - the GRID Information Protocol (used to access information about entities) and GRRP - the GRID Registration Protocol (used to notify directory services of the availability of certain information) [8].

A number of other services are available within the Globus Toolkit to deal with issues such as:

- security, authentication, integrity and confidentiality (provided by the GRID Security Service, GSI),
- the management of data movement and access strategies (provided by Global Access to Secondary Storage, GASS),
- data transfer in and replication management (for instance provided by GridFTP),
- and the monitoring of the system state (provided by Heartbeat Monitor, HBM), cf. [3, 9, 10].

Each Globus service has an API (written in C). In addition Java classes are available for important services. The Globus services have been implemented in a joint effort by the participating project partners. These services support GRID applications that run on existing hardware platforms and hosting environments. Thus, the implementations make extensive use of existing technologies, platforms, languages (e.g. CORBA, Java, MPI Python) and services (such as the LDAP, SLP, DNS, UDDI) whenever deemed appropriate.

4 The Open GRID Services Architecture (OGSA)

The Open GRID Services Architecture (OGSA) combines GRID technology and Web services to build an open architecture for GRID services [11]. With this move the GRID has started to adopt a strong service orientation. The goal is to provide a set of well-defined basic interfaces and an architecture that is extensible, vendor neutral and adheres and contributes to open standards. Within OGSA, everything is regarded as a service - including applications that become Web services. OGSA enables the development of virtual infrastructures that form part of virtual organizations. These virtual organizations can be of different sizes, lifetimes, spanning multiple (physical) organizations and run on heterogeneous infrastructures (i.e. provide a consistent functionality across various platforms and hosting environments) [1].

OGSA defines the mechanisms required for sophisticated distributed systems (including change and lifetime management, and notification). This is done using the Web Services Description Language (WSDL) and associated conventions. The combination of GRID technology and Web services makes best use of the advantages of both technologies. Web services define techniques for describing software components and accessing them. Further, Web services discovery methods allow the identification of relevant service providers within the system regardless of platform or hosting environment specific features. Web services are open in that they are programming language, programming model, and system software neutral.

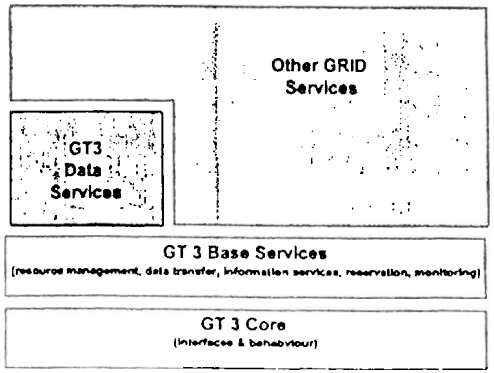

Figure 2: OGS Infrastructur: GT3

With the emergence of OGSA a new version of the Globus Toolkit was developed (Globus Toolkit Version 3, GT3). Since then the Open GRID Services Infrastructure and the Globus architecture are defined together. The latest version is GT4. The Open GRID Services Infrastructure is defined as a minimal layer that enables GRID services to invoke each other as well as baseline operations. This corresponds to the GT3 Core and GT3 Base Services Layer as shown in Figure 2. The GT3 Core will implement the service interfaces and behaviors described in the GRID Services Specification [12]. Existing Globus toolkit components might be reused within the GT3 Core and GT3 Base Services Layer.

A number of (standard) high-level services that address requirements of eBusiness and eScience applications are being discussed within the OGSA initiative. Such services include:

- distributed data management services (e.g. for database access, data translation, replica management and location, and transactions),
- workflow services (for coordinating different tasks on multiple Grid resources),
- auditing services (for recording usage data),
- instrumentation and monitoring services (for measuring and reporting system state information),
- and security protocol mapping services (for enabling distributed security protocols to be transparently mapped onto native platform security services).

These services can be implemented and composed in various different ways replacing some of the current Globus toolkit services for instance dealing with resource management and data transfer [11].

4.1 GRID Services Characteristics

OGSA defines a GRID service as a network-enabled entity that represent computational and storage resources, networks, programs, and databases, inter alia. Within the virtual organization formed by these services, clear service definitions and a set of protocols are required to invoke these services. The protocol is independent of the actual service definition and vice versa. It specifies a delivery semantic and addresses issues such as reliability and authentication. A protocol that guarantees that a message is reliably received exactly once can for instance be used to achieve reliability, if required. Since for the service definition WDSL is used, multiple protocol bindings for a single interface are possible [11]. However, the protocol definition itself is outside the scope of OGSA.

Openness within OGSA is ensured by virtual service definitions that are used to produce multiple (ideally interworking) implementations. A client invoking a service should not have to consider the platform a service instantiation is running on, or have to know anything about the implementation details. The interaction between services happens via well-defined, published service interfaces that are implementation independent. The interfaces address discovery, dynamic service creation, lifetime management, notification, and manageability. At the same time, they take into account upgradeability and naming conventions. In order to increase the generality of the service definition, authentication and reliable service invocation are viewed as service protocol binding issues that are external to the core service definition but which have to be addressed within a complete OGSA implementation.

Since within the GRID infrastructure services are not necessarily static and persistent (i.e. a service can be created and destroyed dynamically), the OGSA services are also concerned with transient service instances. Furthermore, OGSA conventions allow identifying service changes such as service upgrades. The information about these changes also state whether the service is backward compatible regarding interface and semantics.

Since GRID services have to run on multiple platforms in a distributed heterogeneous environment, service implementations should be ideally portable not only in terms of their design but also as far as code and the hosting environment is concerned.

4.2 GRID Service Specification

A GRID service interface specifies a set of operations that can be invoked by exchanging a defined sequence of messages. The GRID service interfaces correspond to WDSL portTypes that are defined by serviceTypes. These serviceTypes are WDSL extensibility elements defined by OGSA that contain the portType definition and additional information relating to versioning. Hence, the portType defines the GRID service's interface and the serviceType the GRID service. For its lifetime, a GRID service can maintain a state. This state distinguishes one service instance from others. In the context of GRID services, a particular instantiation of a service is referred to as GRID service instance.

Services are dynamic; interfaces are defined for managing their lifetime and state. In order to distinguish services, each service is assigned a globally unique name called GRID Service Handle (GSH). This name is unique, i.e. it is created to name one service instance and will never be (re-)used in connection with any other GRID service. In the GSH no reference is made to any protocol or instance specific information such as network addresses and supported protocol bindings. For long-lived and persistent services, this information is kept with the GRID Service Reference (GSR) - together with all other vital instance

specific information. In contrast to the GSH, the GSR may change over a service's lifetime and can become invalid.
Information about a GRID service is referred to as service data. There are two types of service data pertaining to a service instance:
• metadata (i.e. information about a service instance) and
• state data (specifying runtime properties of a service instance).
So-called Service Data Elements (SDE) represent the service data of a particular service instance. SDE correspond to containers holding XML encoded metadata and state data elements. The SDE can appear as extensible element in the portType and serviceType description. The characteristics of an SDE is defined by the Service Data Description (SDD) that specifies properties such as the SDE name, the XML type of the service data value elements, how many times such elements might occur, if they change during the lifetime of an instance, etc. The SDD must be unique within the SDD elements namespace [12].
The method Notification is used to exchange information between GRID service instances. This method is a one-way asynchronous delivery from a notification source to a notification sink. It corresponds to a push model for data delivery. The semantics of notification are a property of the protocol binding used to deliver the message. A service can pull information pull is possible by using FindServiceData. This can for instance be used for service discovery (via a basic GRID service called registry) [11].
A set of basic OGSA interfaces (i.e. WSDL portTypes) for manipulating service model abstractions have been defined. The interfaces can be combined in different ways to from various GRID services. Service interfaces defined so far are for instance [11]:
• FindServiceData to query information about a GRID service instance such as basic introspection information (handle, reference, primary key, home handleMap, etc.) and service specific information (e.g. service instances known to a registry). It provides extensible support for various query languages.
PortType: GridService
SetTerminationTime to set and query the termination time of a GRID service instance.
PortType: GridService
• Destroy to terminate a GRID service instance
PortType: GridService
• SubscribeToNotificationTopic to subscribe to notifications of service-related events (based on message types and interest statement; allows the delivery via third party messaging services).
PortType: NotificationSource
• DeliverNotification to asynchronously deliver notification messages.
PortType: NotificationSink
• RegisterService to conduct soft-state registration of GSH.
ProtType: Registry
• UnregisterService to deregister a GSH.
PortType: Registry
• CreateService to create a new GRID service instance.
PortType: Factory
• FindByHandle to return the GSR currently associated with the supplied GSH.
PortType: HandleResolver
Different service interfaces are associated with different PortTypes. The Factory PortType for instance refers to a programming model related abstract concept or pattern. It is used by a client to create an instance of a GRID service. When a client invokes a CreateService on a Factory it receives as response a GSR for the newly created service. The HandleRelsolver PortType is a GRID Service for resolving a GSH to a GSR. It has not yet been decided what the role of the resolver protocol and the HandleResolver in this context should be [12].

4.3 Hosting Environments

OGSA defines the basic behavior of a service but does not prescribe how a service is executed. It does not address issues such as implementation programming model, programming language, implementation tools and debugging tools, or execution environments. This is determined by the hosting environment that also defines how a GRID service implementation realizes the GRID service semantics [1]. The hosting environment is provided for most GRID applications by an operating system. The service can be implemented in a number of programming languages on such a hosting environment (e.g. in C, C++, Java, Fortran, Python). The use of standard programming libraries to implement services can facilitate the development. However, these have to be regarded as specific to the hosting environment and are not a general representation of the respective GRID service. Although the goal is to have generic, platform independent GRID services, it has to be considered that the hosting environment to some extend also determines the kind service semantics that can be offered.
Apart from the traditional OS based implementations, GRID services can also be built on top of new hosting environments such as J2EE, Web-sphere, .NET, JXTA, or Sun ONE. These hosting environments tend to offer better programmability and manageability. They are usually also more flexible and provide a degree of safety.
Despite OGSA not being concerned with implementation details, the definition of baseline characteristics can facilitate the service implementation. Issues that

have to be addressed in the context of hosting environments should the mapping of GRID wide names and service handles into programming language specific pointers or references, the dispatch of invocations into actions such as events and procedure calls, protocol processing and data formatting for network transmission, lifetime management, and inter-service authentication.
Hosting environments can be constituted by complex structures that span over a number of simple hosting environments creating a virtual hosting environment. The resources accessed in such a complex hosting environment are accessed in the same way than of those GRID services implemented on simple hosting environments.

4.4 Protocol Bindings

There are various protocol bindings possible for a service, for instance SOAP/HTTP with TLS (for security). Although the protocol bindings allow a certain degree of flexibility there are certain requirements protocol binding should meet. The following four primary requirements have been proposed in [11]:

- Reliable transport; this might be required for some service invocation. It can already be supported by certain protocol bindings such as HTTP-R.
- Authentication and delegation to communicate proxy credentials to remote side. This can be supported within the network protocol binding (e.g. by TLS extended with proxy credential support).
- Ubiquity is required to enable any arbitrary pair of service instances to interact.
- GSR format that considers binding specific formats, e.g. WSDL document or CORBA IOR.

Although OGSA is not chiefly concerned with the specification of communication protocols, it might be of advantage to define a set of protocols that specify a small number of protocol bindings for example for service discovery and invocation to allow any two services to communicate. It has been acknowledged that this kind of InterGRID protocol could be useful but it is an open issue if such a protocol will be defined and subsequently accepted.

5 GRID and P2P Computing

There is an ongoing argument about GRID and P2P computing, their merits, differences, and commonalities. The comparison usually concentrates on certain aspects in order to highlight specific issues. However, the situation is more complex since both terms are used in different contexts and sometimes can have various connotations. In the following we try to capture both as comprehensively as possible and strive to give an overview of the related issues.

Although P2P is mainly used in connection with file sharing systems, other areas (such as collaboration support and distributed computing) also use P2P mechanisms [24]. The term GRID is also not solely used for Globus or OGSA related activities. More and more it is also becoming part of product descriptions (e.g. of IBM and Oracle).
A number of publications are combining both concepts or are describing GRID systems that employ P2P mechanisms [25, 26, 27]. In order to compare both concepts and assess how much they have in common, it is required to characterize P2P in this context in more detail.
In [28] a P2P system is defined as a self-organizing system of equal, autonomous entities (i.e. peers) that operates preferably without using any central services based on a communication network with the purpose of resource sharing. Here, the emphasis is on the system aspect that allows joint resource utilization. Another characterization [15] stresses that P2P is a class of applications that takes advantage of resources that are available at the edge of the network. This latter definition gives a much more concrete view on the nature of P2P computing in that it describes it as "class of application", not systems, components or platform. Therefore, it is important to distinguish between the *P2P paradigm* that encompasses decentralization, self-organization, and autonomous collaboration between independent entities in a system context, and *P2P applications* that are mostly vertically integrated applications used for the sharing of specific resources (e.g. file and information sharing [29]). Further, there are also emerging P2P platforms that provide an operating system independent middleware layer that allows sharing of resources in a peer-to-peer fashion between participating entities [24].

5.1 Comparing GRID and P2P: Commonalities and Differences

The initial motivation behind GRID and P2P applications has been similar, both are concerned with the pooling and organization of distributed resources that are shared between (virtual) communities connected via the Internet. The resources and services they provide can be located anywhere in the system and are made transparently available to the users on request. Both also take a similar structural approach by using overlay structures on top of the underlying communication (sub-)system.
However, there are also substantial differences on the application, functional and structural level. The applications supported through the GRID are mainly scientific applications that are used in a professional context. The number of entities is rather moderate in size, the participating institutions are usually known. Current P2P applications, in contrast, provide open access for a large, fluctuating number of unknown participants with highly variable behavior.

Therefore, P2P has to deal with scalability and failure issues much more than GRID applications. P2P applications are largely concerned with file and information sharing. In addition, they usually provide access to simple resources (mostly files, sometimes processing power), whereas the GRID infrastructure provides access to a resource pool (e.g. computing clusters, storage systems, databases but also scientific instruments, sensors, etc.) [15]. P2P applications usually are vertically integrated using overly structures as part of the application. The GRID is essentially a multipurpose infrastructure where the core functionality is provided by a set of tools and services that are part of the architecture, the resources can be used by different applications.

In recent years a number of P2P middleware platforms have been developed that provide generic P2P support: The functionality they support comprises for instance naming, discovery, communication, security, and resource aggregation. One example is JXTA [32], an open platform designed for peer-to-peer computing. Its goal is to develop basic building blocks and services to enable innovative applications for peer groups. JXTA provides a common set of open protocols and an open source reference implementation for developing peer-to-peer applications. The JXTA protocols are designed to be independent of programming languages, and independent of transport protocols. Another, commercially driven P2P platform is Microsoft's Windows Peer-to-Peer Networking (MSP2P) [33]. It provides simple access to networked resources. There are three central components within MSP2P providing generic P2P communication and interaction support, i.e. the Peer Name Revolver Protocol (PNRP), Graphing API (for the organization of peers in a virtual network graph), and Grouping API (to form closed groups). The P2P functionality is closely coupled with the Windows Sockets 2 architecture. There is also ongoing research in this area. For instance in the EU funded project on Market Managed Peer-to-Peer Services (MMAPPS) a middleware platform has been created that incorporates market mechanisms (in particular accounting, pricing and trust mechanisms) [34]. On top of this platform, a number of applications (i.e. a file sharing application, a medical application, and a WLAN rooming application) have been implemented to show how such a generic platform can be used. However, the emphasis of these platforms is predominantly on the middleware aspect and not (as with the GRID initiative) in providing access to a universal computing resource infrastructure comparable to the power grid.

The P2P paradigm has become an underlying theme of the P2P platform development as well as of most P2P applications. Although there are a number of applications widely regarded as P2P (e.g. Napster [31], SETI@home [30]) that use central entities. The reason why they are sometimes considered being P2P is that the participating peers are still autonomous in there behavior and decisions.

In contrast, within the GRID infrastructure centralized components and client/server structures are often used because they are considered best suited for their specific purposes. However, this is not inherent in the idea or concepts behind GRID. Rather it can be attributed to the nature and scale of the current implementation, and the requirements placed onto the infrastructure by applications and the kind of integrated components.

5.2 GRID and P2P: Converging Concepts?

The GRID has been successful in operation within the scientific community for a number of years. However, the potential of the GRID goes beyond scientific applications and can be for instance also applied to the government domain, healthcare, industry and eCommerce sector [14]. Many of the basic concepts and methods could remain unchanged when applied to these new domains. Other issues not within the scope of the current GRID initiative will have to be addressed in the context of these application areas (e.g. commercial accounting and IPR issues). Further, with a more widespread adoption of the GRID, there is a greater need for scalability, dependability and trust mechanisms, fault-tolerance, self-organization, self-configuration, and self-healing functionality. This indicates that mechanisms from the P2P application and platform domain and the P2P paradigm in general are going to be adopted more widely by the GRID. This would result in more dynamic, scalable, and robust infrastructures without changing the nature or fundamental concepts behind it. It is just a further development of the original idea.

P2P applications on the other hand are also developing into more complex systems that provide services that are more sophisticated. A platform approach has been proposed by some vendors and research initiative to provide more generic support for sophisticated P2P applications. It is being expected that developers of P2P systems are going to become increasingly interested in such platforms, standard tools for service description, discovery and access, etc. [15]. Such a P2P infrastructure would then have a lot in common with the GRID infrastructure. However, the ambition behind the GRID (i.e. providing access to computational resources comparable to the power grid) is not shared by these middleware platforms. They are built for better and more flexible application support.

Essentially, it is a matter of substantiating the claims represented by the P2P paradigm of providing more flexibility, dynamicity, robustness, dependability and scalability for large scale distributed systems. If this is successful and additional quality features (such as performance and efficiency) can also be

ensured, P2P mechanisms can become core to the GRID. P2P applications, on the other hand, will have to adopt a more platform-based development to provide sufficient flexibility in a very dynamic environment. It remains to be seen if this means a convergence of the two areas or if they will co-exist mutually influencing each other.

6 Summary

The idea for the GRID was conceived within the science community inspired by the success of the Internet and results produced by distributed systems research. The main target application areas are resource sharing, distributed supercomputing, data intensive computing, and data sharing and collaborative computing. The idea of the GRID is now being extended into other areas such as eLearning and eBusiness.

The relationship between peer-to-peer and GRID is a topic still argued about. Since GRID is defined as infrastructure, its scope and extent is better defined than that of P2P. The term P2P is on the one hand being used for a group of distributed applications such as the well known file sharing applications. On the other hand it also refers to a paradigm encompassing the concepts of decentralization, self-organization and resource sharing within a system context [28]. Recently, middleware platforms have been developed that provide generic support for P2P applications, implementing the P2P paradigm in an operating system independent fashion. The objective of the GRID is to provide an infrastructure that pools and coordinates the use of large sets of distributed resources (i.e. to provide access to computational resource similar to the access to electricity provided by the power grid). So far, GRID tools and services are mainly developed using "traditional" distributed systems concepts (such as client/ server). However, it has been recognized that an adoption of P2P principles could be beneficial in terms of scalability, dependability, and robustness. The pooling and sharing of resources is also a common theme in P2P applications. This could be supported by P2P middleware platforms in the future. Though, this does not mean global access to computational resources anywhere, anytime. The question if and how both concepts converge is still open.

References

[1] I. Foster, C. Kessleman, J. Nick, S. Tuecke: "Grid Services for Distributed System Integration", IEEE Computer, No. 36, June 2002.

[2] I. Foster, C. Kessleman,: "Computational GRIDS" in The GRID: Blueprint for a Future Computing Infrastructure, I. Foster, C. Kessleman (Eds.), Morgan Kaufmann Publishers, 1999.

[3] "The Globus Project", Web site: www.globus.org, 2002

[4] C. Liu, L. Yang, I. Foster, D. Angulo:" Design and Evalutation of a Resource Selection Framework for GRID Applications" GLOBUS Technical Report, www.globus.org/research/papers.html, 2002.

[5] K. Czajkoskiw, I. Foster, N. Karonis, C. Kesselman, S. Martin, W. Smith, S. Tuecke: "A Resource Management Architecture for Metacomputing Systems" Proc. IPPS/SPDP '98 Workshop on Job Scheduling Strategies for Parallel Processing, 1998.

[6] V. Sander, W. Adamson, I. Foster, A. Roy: "End-to-End Provision of Policy Information for Network QoS" Proc. of 10th IEEE Symposium on High Performance Distributed Computing, 2001.

[7] I. Foster, A. Roy, V. Sander: "A Quality of Service Architecture that Combines Resource Reservation and Application Adaptation" Proc. of 8th IWQoS, 2000.

[8] K. Czajkowski, S. Fitzgerald, I. Foster, K. Kesselman: "GRID Information Services for Distributed Resource Sharing" Proc. of 10th IEEE Symposium on High Performance Distributed Computing, 2001.

[9] B. Allcock, J. Bester, J. Bresnahan, A. Chervenak, I. Foster, K. Kesselman, S. Meder, V. Nefedova, D. Quesnel, S.Teuck: "Data Management and Transfer in High-Performance Computational GRID Environments" in Parallel Computing, 2001.

[10] W. Allcock, I. Foster, S. Tueck, A. Chervenak, K. Kesselman: "Protocols and Services for Distributed Data-Intensive Science" Proc. of ACAT2000, 2000.

[11] I. Foster, C. Kesselman, J. Nick, S. Tuecke: "The Physiology of the GRID" Globus Technical Reoport, www.globus.org/research/papers.html, 2002.

[12] S. Tuecke, K. Czajkowski, I. Foster, J. Frey. S. Graham, C. Kesselman: "GRID Service Specification " Globus Technical Reoport, www.globus.org/research/papers.html, 2002.

[13] I. Foster, C. Kesselman, S. Tuecke: "Die Antomie des GRID" in Peer-to-Peer, D. Schoder, K. Fischbach, R. Teichmann, (Hrsg.), Springer- Verlag Berlin, Heidelberg, 2002.

[14] T. Barth, M. Grauer: "GRID Computing – Ansätze für verteiltes virtuelles Prototyping" in Peer-to-Peer, D. Schoder, K. Fischbach, R. Teichmann, (Hrsg.), Springer- Verlag Berlin, Heidelberg, 2002.

[15] I. Foster, A. Iamnitchi: "On Death, Taxes, and the Convergence of Peer-to-Peer and Grid Computing, in Proceedings of 2nd International Workshop on P2P Systems (IPTPS'03), 2003.

[16] W. Hoschek, J. Jaen-Martinez, A. Samar, H. Stockinger, K. Stockinger: "Data Management in an International Data Grid Project", in Proceedings of the 1st IEEE/ACM International Workshop on Grid Computing, 2000

[17] R. Moor, C. Baru, R. Marciano, A. Rajasekar, M. Wan: "Data-Intensive Computing" in The GRID: Blueprint for a Future Computing Infrastructure, I. Foster, C. Kessleman (Eds.), Morgan Kaufmann Publishers, 1999.

[18] I. Foster, A. Roy, V. Sander: "A Qualtiy of Service Architecture that Combines Resource Reservation and Application Adaptation", in Proceedings of the 8th International Workshop on Quality of Service, 2000.

[19] I. Foster: "The Grid: A new Infrastructure for 21st Century Science", in Physics Today, No. 55 (2), 2002.

[20] G. Fedak, C. Germain, V. Neri, F. Cappello: "XtremWeb: A Generic Global Computing System", in Proceedings of Workshop on Global Computing on Personal Devices (CCGRID2001), 2001.

[21] I. Foster, C. Kessleman,: "Globus: A Toolkit-Based Grid Architecture" in The GRID: Blueprint for a Future Computing Infrastructure, I. Foster, C. Kessleman (Eds.), Morgan Kaufmann Publishers, 1999.

[22] W. Benger, I. Foster, J. Novonty, E. Seidel, J. Shalf, W. Smith, P. Walker: „Numerical Relativity in a Distributed Environment", in Proceedings of the 9th SIAM Conference on Parallel Processing for Scientific Computing, 1999.

[23] I. Foster, C. Kessleman, G. Tsudik, S. Tuecke: "A Security Architecture for Computational Grids", in Proceedings of ACM Conference on Computers and Security, 1998.

[24] A. Mauthe, D. Hutchison: "Peer-to-Peer Computing: Systems, Concepts and Characteristics". Praxis in der Informationsverarbeitung & Kommunikation (PIK), K. G. Sauer Verlag, Special Issue on Peer-to-Peer, 2003.

[25] A. Iamnitchi, I. Foster: "A Peer-to-Peer Approach to Resource Location in the Grid Environments", in Grid Resource Management, J. Weglarz, J. Nabrzyski, J. Schopf, and M. Stroinski, Eds. Kluwer Publishing, 2003.

[26] A. Iamnitchi, I. Foster, D. Nurmi: "A Peer-to-Peer Approach to Resource Discovery in Grid Environments", Report TR-2002-06, University of Chicago, 2002.

[27] T. Ackermann, R. Gold, C. Mascolo, W. Emmerich: "Incentives in Peer-to-Peer and Grid Networking", UCL-CS. Research Note 02/24, 2002.

[28] R. Steinmetz, K. Wehrle: "Peer-to-Peer-Networking and -Computing", Informatik Spektrum, Vol.: 27(1), February 2004.

[29] F. Kileng: "Peer-to-Peer File Sharing Technologies: Napster, Gnutella and Beyond", Technical Report, 18/ 2001 Telenor, http://www.telenor.no/fou/publisering/Rapp01/R18_2001.PDF, 2002.

[30] SETI@home Web Site: "SETI@home - The Search for Extraterrestrial Intelligence" University of Berkeley, http://setiathome.ssl.berkeley.edu/, 2002.

[31] C. Shirky: "Listing to Napster", in Peer-to-Peer – Harnessing the Power of Disruptive Technologies, A. Oram (Edt.), O'Reilly, 2001.

[32] Sun Microsystems: "JXTA v2.0 Protocol Specifications", http://www.jxta.org, 2004.

[33] E. Chtcherbina, B. Freisleben, T. Friese: "Peer-to-Peer Computing: Microsoft P2P versus Sun JXTA", in JavaSPEKTRUM, vol. 5, 2004.

[34] Ben Strulo, "Middleware to Motivate Co-operation in Peer-to-Peer Systems (A Project Discussion)", in P2P Journal, March, 2004.

Authors

Andreas Mauthe is a Senior lecturer at the Computing Department of Lancaster University. His research areas are content management and content distribution architectures and large-scale distributed systems using the P2P paradigm and autonomous mechanisms for co-ordination.

Oliver Heckmann is a postdoc researcher and research group leader at TU Darmstadt. His "IT Architectures" research group investigates service oriented architectures to support business processes with support for quality of service. In the "Peer-to-Peer Networking" group, the peer-to-peer communications paradigm and the quality of service achievable with peer-to-peer communication is investigated.

Acknowledgement

This paper was written in collaboration between KOM, TU Darmstadt and the Computing Department, Lancaster University supported by the European Network of Excellence E-Next/ FP6-506869.