13. Distributed Computing – GRID Computing

Andreas Mauthe (Lancaster University)
Oliver Heckmann (Darmstadt University of Technology)

13.1 Introduction

The idea of GRID computing originated in the scientific community and was initially motivated by processing power and storage intensive applications [213]. 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 example, bandwidth, storage, processing capacity, and data [304, 429]. The resources pertain to organizations and institutions across the world; they can belong to a single enterprize or be in an external resource-sharing and service provider relationship. On the GRID, they form distributed, heterogeneous, dynamic *virtual organizations* [221]. The GRID provides a resource abstractions in which the resources are represented by services. Through the strong service-orientation the GRID effectively becomes a networked infrastructure of interoperating services. The driving vision behind this is the idea of "service-oriented science" [215].

The GRID builds on the results of distributed systems research and implements them on a wider scale. Through the proliferation of the Internet and the development of the Web (together with emerging distributed middleware platforms), large-scale distributed applications that span a wide geographical and organizational area becoming possible. This has been taken advantage of within the Peer-to-Peer and the GRID communities more or less at the same time. The GRID has been driven from within 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 [225].

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 [227]. This is one of the defining differences between Peer-to-Peer (P2P) and GRID computing. Although both

R. Steinmetz and K. Wehrle (Eds.): P2P Systems and Applications, LNCS 3485, pp. 193-206, 2005.

are concerned with the pooling and co-ordinated use of resources, the GRID's objective is to provide a platform for the integration of various applications, whereas initially Peer-to-Peer applications were vertically integrated [217].

Many current GRID implementations are based on the Globus Toolkit^{TM}, an open source toolkit [203, 219]. 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) developed by the Global GRID Forum (GGF) - inspired by the Globus project - develops the GRID idea further and is also concerned with issues that have not been the focus of the Globus project such as architectural and standardization matters. OGSA combines GRID technology with Web services. This has led to a strong service orientation, where services on the GRID are generally autonomous and not subject of centralized control.

Besides Globus and OSGA there is the European driven development around Unicorn. It has especially influenced information GRIDs in the corporate world but is also still very much in the development phase. In this chapter we concentrate on the developments within Globus and GGF. Unfortunately, a comprehensive discussion is not possible within the space of this chapter. It rather provides an overview of the ideas and concepts behind the GRID, but also shows how they have evolved from an infrastructure driven approach towards a service-oriented architecture. This is particularly important since the GRID is still a very active and fast developing area. More information can be found in [225] and on the GGF Web site (www.ggf.org).

In this chapter, the main initiatives and concepts driving GRID developments are introduced. This includes an outline of the architectural concepts behind GRID but also a discussion of the Globus Project and the developments around OGSA. Subsequently, the relationship between Peer-to-Peer and GRID is outlined.

13.2 The GRID Architecture

The GRID idea has evolved over the years and there is no prescribed or standardized GRID architecture. The current understanding of the GRID has been influenced by a number of initiatives and individuals who have been driving the development. Therefore, the GRID architecture presented in the following represents a generally accept view and not a standardized reference framework. It should be regarded as abstraction in which the various GRID tools and services can be located according to their functionality.

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*" [225]. 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* [222]. The concept underlying the GRID facilitates collaboration across institutional boundaries. To achieve this a protocol architecture is proposed and standard components are specified that can be used by the different parties entering into a sharing relationship [227].

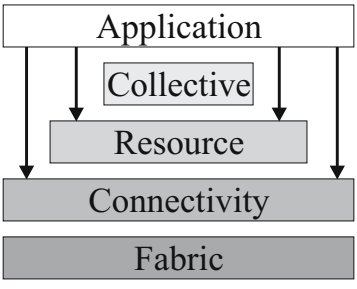

Fig. 13.1: GRID Layered Architecture

The GRID architecture can be described by a layered model where the different components are placed within a layer depending on their functionality and capabilities. An hourglass model (comparable to the Internet), in which a small group of core protocols and components forms the link between the high-level mechanisms and a number of lower level base technologies, has been adopted [67]. The components of the architecture (as depicted in Figure 13.1) are the *Fabric Layer*, the *Connectivity Layer*, the *Resource Layer* and the *Collective Layer* [227]. 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 makes the resources that are provided by the different nodes of the GRID available for common usage, i.e., it provides common access to resources that are shared within a virtual organization. Resources are divided into computing resources, storage, network resources, code storage, and directories [227]. The Fabric Layer implements the resource specific operations and offers a unified interface to the upper layers.

The Connectivity Layer hosts the most important communication and authentication protocols required for GRID specific communication. It enables 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 [223]. Above 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 [144], scheduling and brokering services, monitoring and diagnostic services, data replication services, workload management, etc. [227].

Finally, the Application Layer comprises the user applications used to achieve the virtual organization. Applications are utilizing the services offered by the underlying layers. They can directly access these services.

13.3 The Globus Project

The Globus project started in 1996 and 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 was one of the first and most visible activities in this area. 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) [248]. The project is centered on four main activity areas:

1. *Building of large-scale GRID applications* such as distributed supercomputing, smart instruments, desktop supercomputing tele-immersion.
2. Support for planning and *building of large-scale testbeds* for GRID research, as well as for functional GRID systems
3. *Research into GRID related issues* such as resource management, security, information services, fault detection and data management.
4. *Building of software tools* for a variety of platforms (the so-called Globus Toolkit^{TM}), which are, however, considered research prototypes only.

The Globus Toolkit^{TM} supplies the building blocks for a GRID infrastructure, 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 [221]. The services are programs that interact with each other to exchange information or co-ordinate the processing of tasks. They can be used independently or together to form a supporting platform for GRID applications.

A Globus service encapsulates a certain functionality and provides an abstraction for resources. For instance, a number of 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. The *Resource Selection Service* (RSS) provides a generic resource selection framework for all kinds of GRID applications. For a specific case it identifies a suitable set of resources by taking into account application characteristics and system status [392].

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 [226][533]. Dynamic feedback is used among resource managers to coordinate resource management decisions [226]. 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 [145]. The resource management tools build on existing languages, protocols, and infrastructure. Their capabilities depend on the functionality and capacity of the hosting environments in which they operate, i.e., they can only guarantee a certain level of QoS if this is supported by the underlying networks and operating systems.

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-2 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; namely 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) [144].

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. [23, 24, 248].

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. The 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.

13.4 Defining the GRID: The Global GRID Forum Initiative

The *Global GRID Forum* (GGF) subsumes the major activities in the GRID domain. It is a community of users, developers, and vendors that is leading the global standardization effort for GRID computing. GGF promotes the adoption of GRID computing in research and industry by defining GRID specifications and building an international community for the exchange of ideas, experiences, requirements and best practices.

Within GGF the *Open GRID Services Architecture* (OGSA) has been defined. OGSA integrates key GRID technologies (including the Globus Toolkit^{TM}) and combines them with Web services to build an open architecture for GRID services [222]. The goal is to provide a set of well-defined basic interfaces and an architecture that is extensible, vendor neutral and adheres (and in some cases actively contributes) to open standards. The *Open GRID Services Infrastructure* (OGSI) supports the definition of services that compose OGSA by extending WSDL (Web Services Description Language) and XML schema. It builds on the GRID and Web services technologies and defines mechanisms for creating, managing, and exchanging information among GRID services [599]. One of the major issues in this context is how to express the relationship between stateful resources and Web services. The *WS-Resource Framework* has been defined to model stateful resources and to formalize interaction with state [216]. This section introduces OGSA and the basic concepts behind it. Further, some of the current issues in the development of GRID services are discussed. It is important to note, however, that the development within the GRID community and GGF has not stopped. Thus, this provides a snapshot of the current state.

13.4.1 The Open GRID Services Architecture (OGSA)

Within OGSA, everything is regarded as a service - including applications that become Web services. This implies that all components in this environment are virtual objects[1]. OGSA enables the development of virtual infrastructures, which 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) [221].

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

[1] Note, OGSA does not use the term "object" since it is regarded as overused. Instead, it defines services.

best use of the advantages of both technologies. Web services define techniques for describing software components and accessing them. Further, Web service 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.

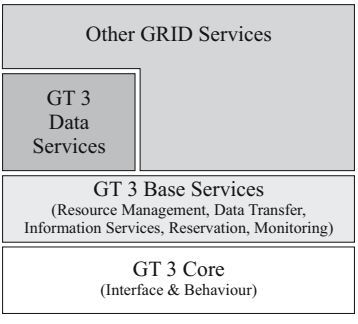

Fig. 13.2: Open GRID Services Infrastructure: GT3

In the context of OGSA, a new version of the Globus toolkit was developed (Globus Toolkit Version 3, GT3). From GT3 onwards the Globus architecture was defined together with OGSA. Currently (i.e. in 2005) GT4 is the most recent version. It is a refined version compared to GT3 where the main concepts are still valid. The Core as shown in Figure 13.2 implements service interfaces and behaviors as specified in the GRID Services Specification [599]. The Core and Base Service Layer are part of the OGSI system framework.

A number of (standard) high-level services that address requirements of eBusiness and eScience applications are being discussed within GGF. 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 [222].

13.4.2 GRID Services: Building Blocks for the GRID

With the strong service orientation adopted by OGSA the GRID can be regarded as a network of services. Web service mechanisms provide support for describing, discovering, accessing and securing interaction (see chapter 14) for a more detailed discussion on Web services). More formally OGSA defines a GRID service as *a network-enabled entity that represents computational and storage resources, networks, programs, and databases, inter alia*. Within the virtual organization formed by these networked services, clear service definitions *and* a set of protocols are required to invoke these services. Note, the protocols are independent of the actual service definitions and vice versa. They specify a delivery semantic and address 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. Multiple protocol bindings for a single interface are possible because WDSL is used for the service definition [222]. However, the protocol definition itself is outside the scope of OGSA.

To ensure openness virtual service definitions are used according to which multiple (ideally interworking) implementations can be produced. Thus, a client invoking a service does 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. 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. Though, they have to be addressed within a complete OGSA implementation.

The OGSA services are also concerned with transient service instances within the GRID infrastructure because services are not necessarily static and persistent (i.e., a service can be created and destroyed dynamically). Furthermore, OGSA conventions allow identification of service changes such as service upgrades. The information documenting these changes also states 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 portable not only in terms of their design, but also as far as code and the hosting environment are concerned. OGSA defines the basic behavior of a service but does not prescribe how a service should be executed. It is the hosting environment, which defines how a GRID service implementation realizes the GRID service semantics [221]. 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 are 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.

13.4.3 Stateful Web Services: OGSI and the WS-Resource Framework

As mentioned before, GRID services are Web services conforming to conventions regarding their interfaces and behavior. By using Web services the service paradigm has been firmly adopted for OGSA. However, a major concept lacking in WSDL is *state*.

The Open GRID Services Infrastructure (OGSI) defines the mechanisms for creating, naming, managing the lifetime of instances of services, and exchanging information between GRID services; it forms the basis for OGSA [599]. Further, it is concerned with declaring and inspecting service data state, asynchronous notification of service state change, representing and managing collections of service instances (so-called *ServiceGroups*), and for common handling of service invocation faults [142]. Therefore, WSDL is extended to make it suitable for the definition of these conventions. In particular, these extensions deal with state (i.e., they allow to create and manipulate stateful Web services) and service typed instances. Furthermore, global naming and addressing is an issue addressed by OGSI. GRID Service Handles in conjunction GRID Service References and interface extensions that introduce portTypes are used to address these issues.

Though, OGSI is regarded as too heavy-weight and not fully compatible with the Web services view [142]. To overcome this the *WS-Resource Framework* (WS-RF) [143] is specified that defines the means by which a Web service and stateful resources are composed. WS-RF is primarily concerned with the creation, addressing, inspection, and lifetime management of stateful resources. Its advantage is that it better exploits existing XML and emerging Web service standards (e.g., WS-Addressing). In short, WS-RF defines a Web service resource as a composition of a Web service and a stateful resource described by an XML document associated with the Web service's port type and addressed using WS-Addressing [312].

13.5 GRID and Peer-to-Peer Computing

There is an ongoing argument about GRID and Peer-to-Peer computing, their merits, differences, and commonalities. It is difficult to delineate the two concepts since there is a lot of overlap in their motivation, though they have a different background and take a different approach. According to Foster's three point GRID checklist [214], for example, a GRID is a system that:

1. coordinates resources that are *not* subject to *centralized* control ...
2. ...using standard, open general-purpose protocols and interfaces...
3. ... to deliver non trivial quality of service

From this list especially the first two points apply to many Peer-to-Peer systems as well. Even on a system level the distinction between Peer-to-Peer and GRID is not clear cut. For instance, the PlanetLab initiative originally conceived as testbed for the Peer-to-Peer research community is also used as example of a GRID infrastructure [215].

A comparison of the two areas at a conceptual level is even more difficult since there is first of all no universally accepted definition of Peer-to-Peer. Moreover, the idea of the GRID is also continuously further developed. Originally it was defined as a "hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities" [225]. Later, the emphasis has shifted towards virtual organizations and the importance of standard protocols in enabling interoperability to build the common infrastructure [214]. Today the focus is on "service-oriented science" and the GRID as infrastructure that enables scientific research by distributed networks of interoperating services [215].

A number of publications are combining both concepts or are describing GRID systems that employ Peer-to-Peer mechanisms [9, 218, 314]. To compare both concepts and assess how much they have in common, it is necessary to characterize Peer-to-Peer in this context in more detail.

In Chapter 2, a Peer-to-Peer 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 for the purpose of resource sharing. Here, the emphasis is on the system aspect that allows joint resource utilization. Another characterization stresses that Peer-to-Peer is a class of applications that takes advantage of resources that are available at the edge of the network [217]. This latter definition gives a much more concrete but also restricted view of the nature of Peer-to-Peer computing in that it describes it as "class of application", not systems, components or platform. Therefore, it is important to distinguish between the *Peer-to-Peer paradigm*, which encompasses decentralization, self-organization, and autonomous collaboration between independent entities in a system context, and *Peer-to-Peer applications*, which are mostly vertically integrated applications [217] used for the sharing of specific resources (e.g., file and information sharing [352]). Further, there are also emerging *Peer-to-Peer platforms* that provide

an operating system independent middleware layer, which allows sharing of resources in a Peer-to-Peer fashion [404].

13.5.1 Comparing GRID and Peer-to-Peer: Commonalities and Differences

The original motivation behind GRID and Peer-to-Peer applications has been similar; both are concerned with the pooling and organization of distributed resources that are shared between (virtual) communities connected via an ubiquitous network (such as 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 levels. The applications supported through the GRID are mainly scientific applications that are used in a professional context. The number of entities is still rather moderate in size, and the participating institutions are usually known. Current Peer-to-Peer *applications*, in contrast, provide open access for a large, fluctuating number of unknown participants with highly variable behavior. Therefore, Peer-to-Peer has to deal with scalability and failure issues much more than GRID applications. Peer-to-Peer applications are still largely concerned with file and information sharing. In addition, they usually provide access to simple resources (e.g. 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.) [217]. Peer-to-Peer *applications* usually are vertically integrated, i.e. the application itself realizes many of the conceptual and basic functionalities that should be part of and architecture or the infrastructure. An example are overlay structures as part of the application. In contrast, the GRID is essentially a multipurpose infrastructure where the core functionality is provided by a set of services that are part of the architecture. The resources are represented by services that can be used by different applications.

In recent years, a number of Peer-to-Peer middleware platforms have been developed that provide generic Peer-to-Peer support. The functionality they support comprises, for example, naming, discovery, communication, security, and resource aggregation. One example is JXTA [330], 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. Another, emerging Peer-to-Peer platform is Microsoft's Windows Peer-to-Peer Networking (MSP2P) [119], which provides simple access to networked resources. There is also ongoing research in this area. For instance, in the EU funded project on Market Managed Peer-to-Peer Services (MMAPPS) a mid-

dleware platform has been created that incorporates market mechanisms (in particular, accounting, pricing and trust mechanisms) [578]. On top of this platform, a number of applications (i.e., a file sharing application, a medical application, and a WLAN roaming application) have been implemented to show how such a generic platform can be used.

13.5.2 GRID and Peer-to-Peer: Converging Concepts?

The GRID has been successfully in operation within the scientific community for a number of years. However, the potential of the GRID goes beyond scientific applications and can for instance also be applied to the government domain, health-care, industry and the eCommerce sector [62]. 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 Peer-to-Peer application and platform domain and the Peer-to-Peer paradigm in general could be adopted more widely by the GRID. This would result in a more dynamic, scalable, and robust infrastructure without changing the nature or fundamental concepts. Though, this will only happen in the context of the service-oriented architecture. Thus, the developments between Peer-to-Peer and Web services as described in Chapter 14 and between Peer-to-Peer and GRID are actually running in parallel.

Peer-to-Peer applications are also developing into more complex systems that provide more sophisticated services. A platform approach has been proposed by some vendors and research initiatives to provide more generic support for sophisticated Peer-to-Peer applications. It is expected that developers of Peer-to-Peer systems are going to become increasingly interested in such platforms, standard tools for service description, discovery and access, etc. [217]. Such a Peer-to-Peer infrastructure would then have a lot in common with the GRID infrastructure. However, the goal behind the GRID (i.e., providing access to computational resources encapsulated as services) is not necessarily 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 Peer-to-Peer paradigm of providing more flexibility, dynamicity, robustness, dependability and scalability for large scale distributed systems. If this is successful and additional quality of service features (such as performance and efficiency) can also be ensured, Peer-to-Peer mechanisms can become central to the GRID. Peer-to-Peer 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, continually influencing each other.

13.6 Summary

The idea for the GRID was conceived within the science community and 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 GRID provides an abstraction for the different resources in form of services.

The architectural view of the GRID can be compared to the Internet hourglass model where a small group of core protocols and components build the link between the high-level mechanisms and a number of lower level base technologies [67]. The various services in this architecture can be located at one of the different layers, namely the Fabric, Connectivity, Resource, and Collective Layer. The Globus Toolkit^{TM} provided the first tools for a GRID infrastructure. These tools exploit the capabilities of the platforms and hosting environments they run on, but do not add any functionality on the system level. Using this pragmatic approach, some remarkable systems have been realized by the Globus project, or with the help of the Globus Toolkit. The OGSA initiative within the Global GRID Forum (conceived within the Globus Project) is developing the original ideas further. It takes a more systematic approach and defines a universal service architecture in which the advantages of GRID technology and Web services are combined. It is strictly service-oriented; i.e. everything is regarded as a service characterized by well-specified platform and protocol independent interfaces. This universal service idea combined with openness and platform independence, allows building very large and functional complex systems. Applying these concepts could provide a way to deal with management issues that have so far restricted the size of distributed systems.

The relationship between Peer-to-Peer and GRID is still a controversial topic. Since GRID is defined as infrastructure formed out of services representing resources, its scope and extent are more well defined than that of Peer-to-Peer. The term Peer-to-Peer 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 [573]. Recently, middleware platforms have been developed that provide generic support for Peer-to-Peer applications, implementing the Peer-to-Peer 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 resources similar to the access to electricity provided by the power grid). The most recent development within the GRID community go towards a strong service orientation. Within GGF the ideas developed in the service-oriented architecture and Web service domain are being adopted. Hence, a convergence between GRID and Peer-to-Peer would actually run in parallel or be predated by a convergence of Peer-to-Peer and Web Services. Though, it has been recognisee that an adoption of Peer-to-Peer principles could be beneficial in terms of scalability, dependability, and robustness. The pooling and sharing of resources is also a common theme in Peer-to-Peer applications. This could be supported by Peer-to-Peer middleware platforms in the future. However, this does not mean global access to computational resources (represented by services) anywhere, anytime. The question of how, indeed if, the two concepts converge is still open.