The Research Monograph Series in Computing, Electrical & Communication Networks
Vol. 1 (01), January 2023, pp. 8–42

BASICS OF MOBILE GRID COMPUTING

It describes the basics of Grid Computing, Mobile Computing and Mobile Grid Computing.

GRID COMPUTING OVERVIEW

The Grid is formed by a collaboration of a potentially unlimited number of ubiquitous computing devices geographically distributed and owned by different organizations in solving large-scale computational and data intensive problems. Grid enables sharing, selection and aggregation of a wide variety of resources, including supercomputers, storage systems, data sources, and specialized devices. The Grid computing is employed in the field of science, engineering, Medicine, commerce,financial and Government organizations. Figure 2.1 shows a Grid Computing environment.

Some of the merits of Grid computing are that it handles large numbers of hardware and software systems to perform computations and functions on the enormous volume of data. It provides highly scalable, highly secure, and extremely high-performance mechanisms for accessing remote computing resources in a seamless manner. The Grid provides transparent access and uniform look and feel for a wide range of high-end resources. A Grid allows location independence [65]. The Grid allows both the set of users and the total set of resources to vary dynamically and continuously.

Grid computing can be used in a variety of ways to address various kinds of application requirements. As the Grid applications are concerned Grid Computing can be classified into a number of types such as computational grid, data grid, science grid, access grid, knowledge grid, cluster grid, terra grid, and commodity grid. The following paragraphs describe different Classifications of Grid concisely.

CLASSIFICATIONS OF GRIDS

Computational Grid

A computational Grid aggregates the processing power of individual or numerous hardware and software-enabled devices such as High-end

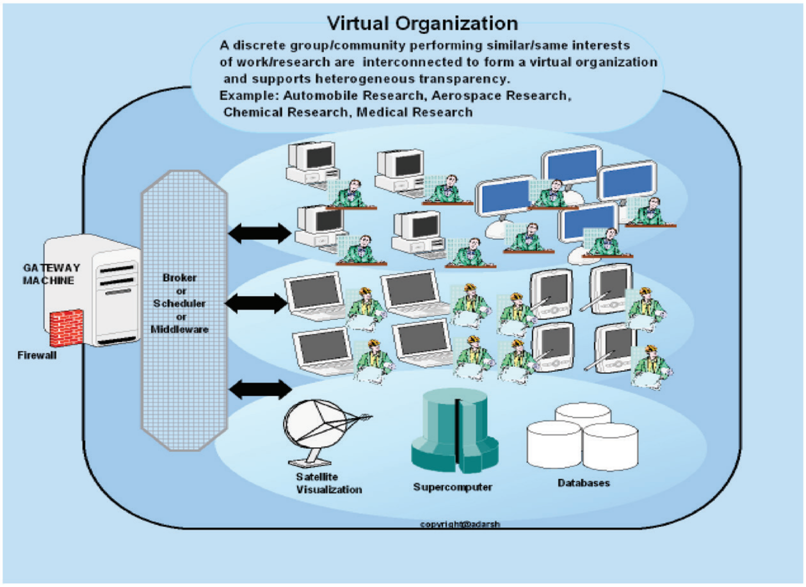

Virtual Organization
A discrete group/community performing similar/same interests of work/research are interconnected to form a virtual organization and supports heterogeneous transparency.
Example: Automobile Research, Aerospace Research, Chemical Research, Medical Research
GATEWAY MACHINE
Firewall
Broker or Scheduler or Middleware
Satellite Visualization
Supercomputer
Databases
copyright@adarsh

Figure 2.1. Grid Computing Environment.

computing systems, online instruments, data archive, Clusters, Intranet and Internet. It runs very large applications such as weather forecasting, stock market management, etc. The computational grid can be realized in a variety of ways. The distributed supercomputing executes the job in parallel at various resources, reducing job completion time. Under high throughput computing overall job execution time is minimized. The On-demand strategy provides a way for business to acquire large scale computing capacity (processor, storage and networking) on demand rather than buying server systems of their own that run their applications. The data intensive computing processes and manipulates large amounts of data and algorithms that scale to high volume of data sets. The Collaborative computing provides an environment in which people can share their information without the constraints of space and time [69].

DATA GRID

A Data grid is a set of services that gives individuals or groups of users the ability to access, modify and transfer extremely large amounts of data distributed geographically. The data in a Data grid can be located at a single site or multiple sites where each site can be governed by its own administrative domain with security restrictions. When a user request for a data the middleware applications and services pull the data from multiple administrative domains and present to the user. Similarly, when large data is to be stored, the middleware moves the data to a single or multiple locations and make an account of it [69].

Science Grid

A Science grid is a new system of interconnected computers in various universities and science labs all over the world. The science grid provides the massive combination of computing power and storage capacity to solve large data-intensive challenges in the field of science. The middleware and tools play a major role in enabling the science grid to attain a high degree of scalability in scientific computing.

Access Grid

The technology was invented at Argonne National Laboratory, Chicago. Access Grid is a collection of resources and technologies that enables large format audio and video based collaboration between groups of people in different locations. It includes multimedia large-format displays, presentation and interactive environments, interfaces with grid computing middleware and visualization environments. In simple terms, it is an advanced video conferencing using big displays and with multiple simultaneous camera feeds at each node (site).

Knowledge Grid

The Knowledge Grid is an intelligent and sustainable interconnection environment that enables people and machines to effectively capture, publish, share and manage knowledge resources. It provides appropriate on-demand services to support scientific research, technological innovation, cooperative teamwork, problem solving, and decision making. A semantic grid is an approach to describe the knowledge grid. The semantic grid applies epistemology and ontology to describe the computing resources and services.

Tera Grid

The Tera grid includes computing systems capable of managing and storing data in terms of teraflops. These components are tightly integrated and connected through a network that operates at 40 gigabits per second. This is a fastest research network. This includes creation of a high-speed network, grid services that provide data sharing, computing power [67].

Commodity Grid

Researchers have created a platform for trading computing resources that allows the selling and buying of standardized computing resources. In the process, they formulate computing a utility like electricity.

In all the aforementioned classifications, Grid Computing applications have the common needs such as:

i. Job partitioning that involves breaking of problems into subtasks
ii. Discovery of suitable Resources for job execution
iii. Scheduling of tasks and workflow
iv. Provisioning and distributing application codes to specific system nodes
v. Results management
vi. Autonomic features such as self-configuration, self-optimization, self-recovery and self-management
vii. Security

The high level primary components of a grid environment that satisfy the common desires are: schedulers and resource brokers, load balancing services, QOS and security services.

GRID COMPONENTS

The following paragraphs explore the Grid Components and their usage patterns in brief and explicit.

Resource Broker

The resource broker provides pairing and brokering services between the service requester and the service provider. This pairing and brokering enables the selection of the best available resources having the capabilities such as resource availability, usage models, and resource constraints and pricing information for execution of a specific task. The resource broker on selecting a suitable resource collaborates with the scheduler and executes the task(s). Figure 2.2 illustrates the use of a resource broker service [67].

Schedulers

Schedulers form the core component of the grids. Schedulers are most responsible entity for managing jobs such as allocating resources for a specific job execution, partitioning jobs into subtasks, parallel

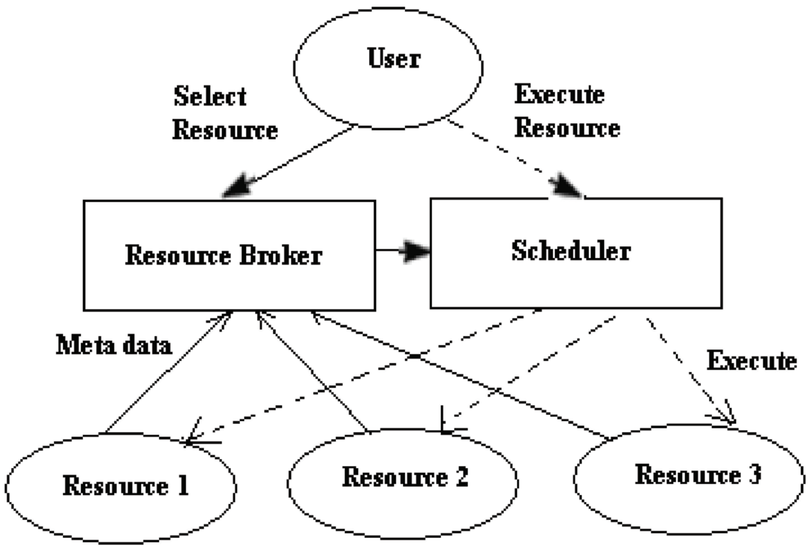


Figure 2.2. Resource Broker.

execution of tasks, data management, correlating events and service-level management capabilities. Figure 2.3 shows a grid scheduling hierarchy. In the hierarchical scheduling system, meta-scheduler forms the root and other lower level schedulers forms the leaves. These schedulers may be formed by local scheduler implementation or other meta-scheduler or a cluster scheduler for parallel executions [67].

The Grid Computing schedulers, with the help of resource brokers identify the resources suitable for user's service-level requirements. The schedulers then submit the jobs to the respective resources for execution. On the completion of job execution the grid scheduler grasps the responsibility of returning back the results to the users. An effective scheduler further provides capabilities such as:

i. Advanced resource reservation
ii. Service-level agreement, validation and enforcement
iii. Job and resource policy management, enforcement for best turnaround times within the allowable budget constraints
iv. Monitoring job executions and status
v. Rescheduling and corrective actions of partial failover situations

Load Balancing

According to Grid Computing, load-balancing is concerned with the distribution of workload among the resources. The workload can be

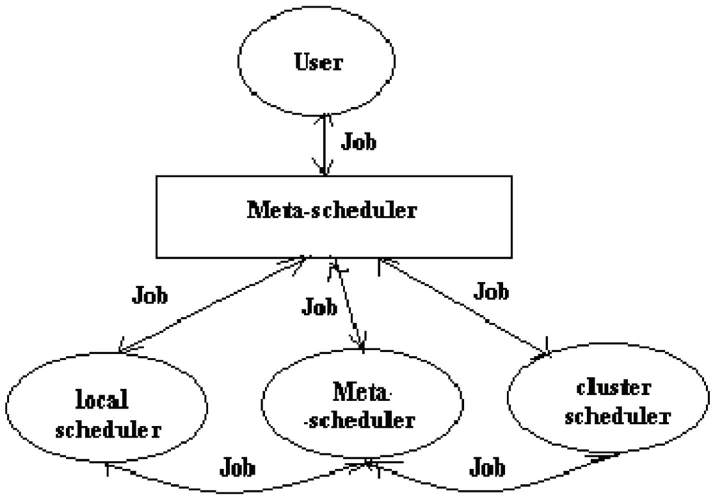


Figure 2.3. Scheduler Hierarchy.

pushed outbound to the resources and can then pull the jobs based on the availability state and/or resources. This load balancing involves partitioning of jobs, identifying the resources, queuing of jobs, running multiple jobs in parallel, redistributing jobs to other resources if needed. These load balancing also support failure detection and management. This load balancing feature is integrated into schedulers and resource brokering system to avoid processing delays and over commitment of resources [67].

QOS Services

Another basic requirement of a Grid Computing system is the ability to provide the quality of service (QoS) requirements necessary for the end-user. These QoS features can be response time measures, aggregated performance, secure fulfillment, resource scalability, and availability, autonomic features such as event correlation and configuration management, and partial failover mechanisms.

There have been a number of activities addressing the above definitions of Grid Computing and the requirements for a grid system. These descriptions and directives for Grid Computing are framed using a variety of technologies and open standards. Global Grid Forum (GGF) is the most prominent organization responsible for standardization and refining process of the grid [67].

GLOBUS GRID FORUM

The GGF was established as a public community forum for the discussion of grid technology issues. The GGF enables a means for coordinating Grid Computing technology efforts, promoting reuse and interoperability, and sharing the results. GGF defines the best practice guidelines for the scientific and industrial usage of the grid. There are more than 400 organizations involved with GGF from around the world. The GGF's primary objective is to promote and support the development, deployment, and implementation of grid technologies and applications via the creation and documentation of best practices specifications, use cases, architecture, and implementation guidelines.

The basic goals of the GGF are:

i. Create an open process for the development of grid agreements and specifications
ii. Create grid specifications, architecture documents, and best practice guidelines
iii. Manage and version controls the documents and specifications
iv. Handle intellectual property policies
v. Provide a forum for information exchange and collaboration
vi. Improve collaboration among the people involved with grid research, grid framework builders, grid deployment, and grid users.
vii. Create best practice guidelines from the experience of the technologies associated with Grid Computing.
viii. The GGF based on their plans and policies designed architecture for Grid environment. The following discusses the Grid Architecture [67].

GRID COMPUTING ARCHITECTURE

The Grid Architecture was developed for the establishment, management and sharing of cross-organizational resources within Grid environment. The grid architecture identifies the basic components of a grid system. It defines the purpose and functions of every component and indicates how each of these components interacts with one another. The interoperability necessitates a set of protocols at each layer. This protocol architecture defines common mechanisms, interfaces, schema and protocols by which users and resources can negotiate, establish, manage, and share resources. Figure 2.4 shows the Grid Architecture [67]. The functions of each layer are explored in detail as

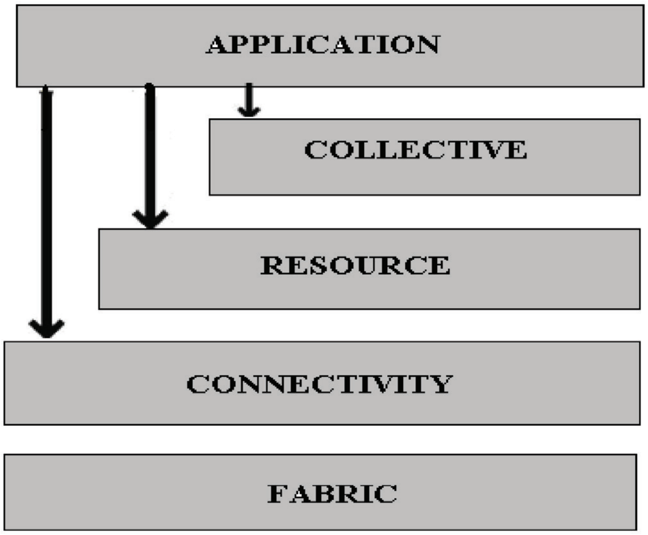


Figure 2.4. Layers of Grid Architecture.

Fabric Layer

The fabric layer is comprised of resources that can be shared among the users in the grid environment. The resource can be a logical resource such as a distributed file system, computer clusters or the distributed computer pool or physical resources such as computational resources, storage systems, catalogues, network resources or sensors. These logical resources are implemented by their own internal protocols (e.g., network file systems [NFS] for distributed file systems, and clusters using logical file systems [LFS]). The resources that are integrated in Grid system are recommended to have the basic capabilities such that they can be discovered against their own resource capabilities, structure, and state of operations. They should provide appropriate resource management capabilities and to maintain Quality of Service the grid solution promises or has been contracted to deliver. The resources should also posses managing capabilities such as start and stop activations, problem resolution, configuration management, scheduling, load balancing, workflow and complex event correlation [65].

Connectivity layer

The connectivity layer provides the core communication and authentication protocols required for grid-specific network transactions. Communication protocols are used for exchanging data between the resources that are included in the fabric layer. The functions of communication protocols are routing, naming and transferring data between resources. The authentication protocols provide secure authentication and data exchange between resources. The functions the authentication protocols include single sign on system, which authenticates' the user for multiple entities with a single authentication process and provide data integrity and confidentiality using various cryptographic and data encryption mechanisms [67].

Resource Layer

The Resource layer utilizes the communication and security protocols defined in the networking communication layers, to control, secure negotiation, initiation, monitoring, metering, accounting and payment involving the sharing of operations across individual resources [67].

Collective Layer

The Collective layer is responsible for all global resource management and interaction with a collection of resources. The protocols of the layer enable the virtual organizational participants to discover the existence of the specific available resources and their properties for a specific task, during a specific period of time and schedule the tasks to the appropriate resources. It also manages resource failure, recovery capabilities, monitoring of the networking and device services and diagnostic services that include common event logging and intrusion detection. It is in this layer, the grid discovers and selects the best software implementation(s) to achieve maximum performance with respect to response time, reliability, and costs [67].

APPLICATION LAYER

This is the final layer in grid architecture that contains user applications, constructed by utilizing the services defined at each lower layer. It includes languages and frameworks which help to directly access the resource or can access the resource through the Collective Service interface APIs [67].

GRID MIDDLEWARE

Grids enable the discovery, selection, sharing, exchange and combine geographically distributed heterogeneous resources such as computers, databases, visualization devices, and scientific instruments. The challenges in grid environment are the heterogeneity, multiple administrative domain, autonomy, scalability, dynamicity, adaptability and quality-of-services. These divergences and deviations of the Grid environment are hidden by the prominent entity called Middleware. The Grid Computing Middleware form the basic building block for constructing a grid. The Middleware is a software layer between the operating system and the applications that provides a higher degree of abstraction in Grid computing environments. These middleware systems aim to provide a grid-computing infrastructure where users link to computer resources without knowing where and how the computing cycles are generated and the job is executed. Middleware systems customize their behaviour dynamically and use the available resources and services efficiently and effectively. It is the middleware which is responsible for carrying out each and every task in Grid environments. The most prominent organizations responsible for the toolkits, middleware, and a framework for Grid Computing are UNICORE, GLOBUS, Gridbus, Legion, Condor and Nimrod-G. Figure 2.5 shows the hardware and software stack within a typical Grid architecture. The functions of the Grid middleware at the four layers within the Grid architecture is demonstrated as fabric, core middleware, user-level middleware, and applications and portals layers.

Core Grid middleware offers services such as remote process management, security, uniform access, dynamic discovery, dynamic aggregation and quality-of-services. These services solve the complexity and heterogeneity. The core middleware follows different approaches in creating a grid, such as scavenging CPU cycles from existing desktops, grabbing dedicated servers and utilizing them in computational grid and to have both scavenging and dedicated resources for the computational grid [64].

User-level Grid middleware utilizes the interfaces provided by the low-level middleware to provide higher level abstractions and services. This layer of middleware includes application development environment, programming tools and resource brokers which help in scheduling the application tasks to global resources and manage them effectively.

Grid applications and portals are typically developed using Grid-enabled languages and utilities such as HPC++ or MPI. Grid portals also offer Web-enabled application services, where users can submit and collect results for their jobs on remote resources through the Web [60].

GLOBUS

The Globus Toolkit GT3 is open standard software developed by the Globus Project that can be used to build computational grids and grid

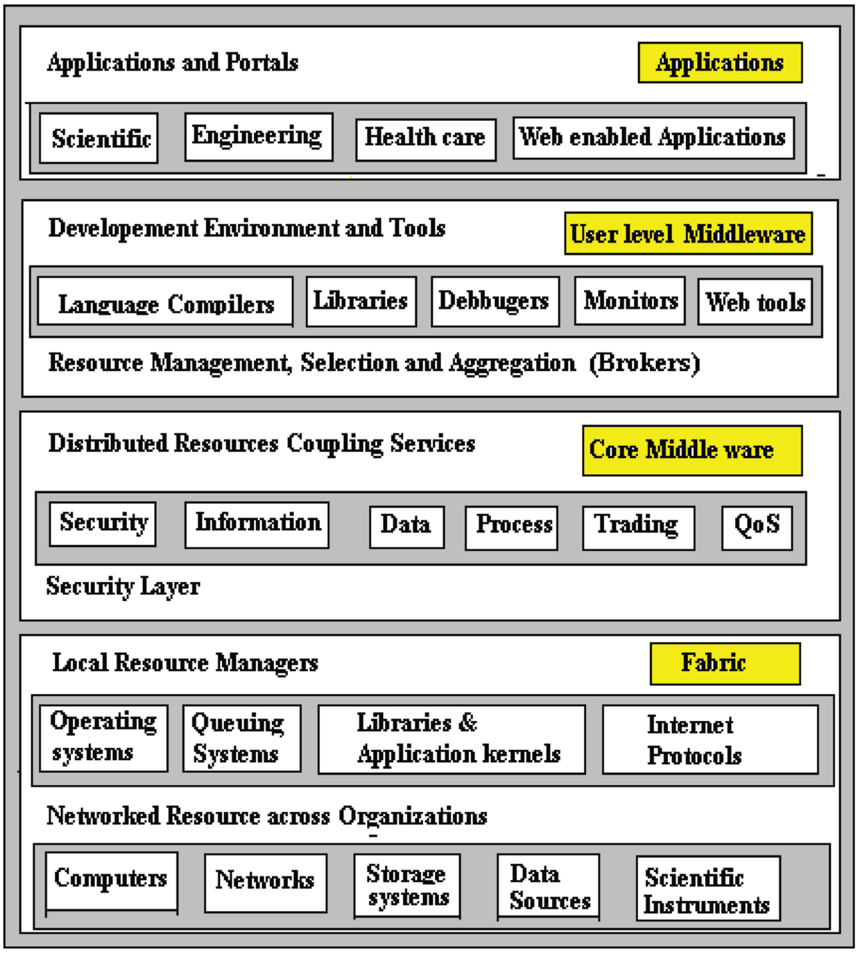


Figure 2.5. Grid Architecture and Components.

based applications. GT3 provides components to implement resource management, data management, and information services. Figure 2.6 shows an overview of the Globus Toolkit. The following paragraphs discuss GT3 Toolkit [69].

Resource management services

Resource management services handle the tasks of resource allocation, job submission and job execution management. This functionality is implemented in a variety of different components. Grid Resource Allocation and Management (GRAM) is a protocol that supports the remote submission of a computational request to a remote computer resource, and subsequent monitoring and control of the resulting computation [26, 27].

When a job is submitted by a client, the globusrun command is executed. This command helps to submit the job to the remote resource and manages it. It first checks for mutual authentication between clients and servers, and verify the rights for submitting the job. It transfers the executable files to the remote host. The Resource Specification Language (RSL) is an XML standard used to describe a job. The RSL includes XML standard elements for specifying.

Executables, directories, arguments, and data files. The RSL is created by the client and passed to the remote host. The gatekeeper daemon located in the remote host handles the request. The gatekeeper creates a job manager to start and monitor the job. When the job is completed, the job manager sends the status information back to the client. Figure 2.7 shows the overview GRAM. The architecture illustrates the aggregation and collective functionality of GRAM [69].

The Globus run command gets the standard output of job results from remote machines. It uses GASS to provide secure file transfer between grid machines. A typical use of GASS is to transfer a job's executable and

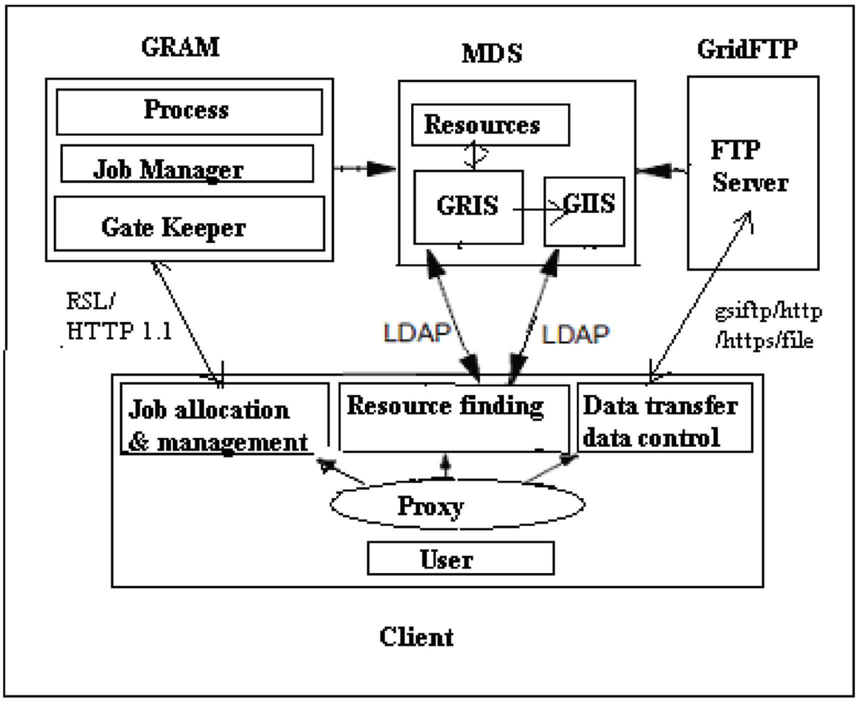


Figure 2.6. Overview of Globus Tool Kit.

any of its inputs to a remote resource, and transfer the resulting output back to the client. Globus resource management supports co-allocation through its Dynamically-Updated Request Online Co-allocator (DUROC) mechanism. DUROC allows jobs to be submitted to resources managed by different resource managers and coordinates the transactions between these managers [29].

The replica location service (RLS) is a part of GRAM, which maintains and provides access to mapping information from logical names regarding data items to target names. These target names may represent physical locations of data items, or it may map to another level of logical naming for the data item. The resource information provider service (RIPS) is another part of GRAM. This service utilizes service data providers to execute system-level information-gathering scripts and tools. Its purpose is to monitor forked processes, schedule queues, and host system statistics. The interested parties can register for subscriptions, based on service data of interest.

Information services

Information services handle the task of collecting information about resources in the grid and responding to queries about this information.

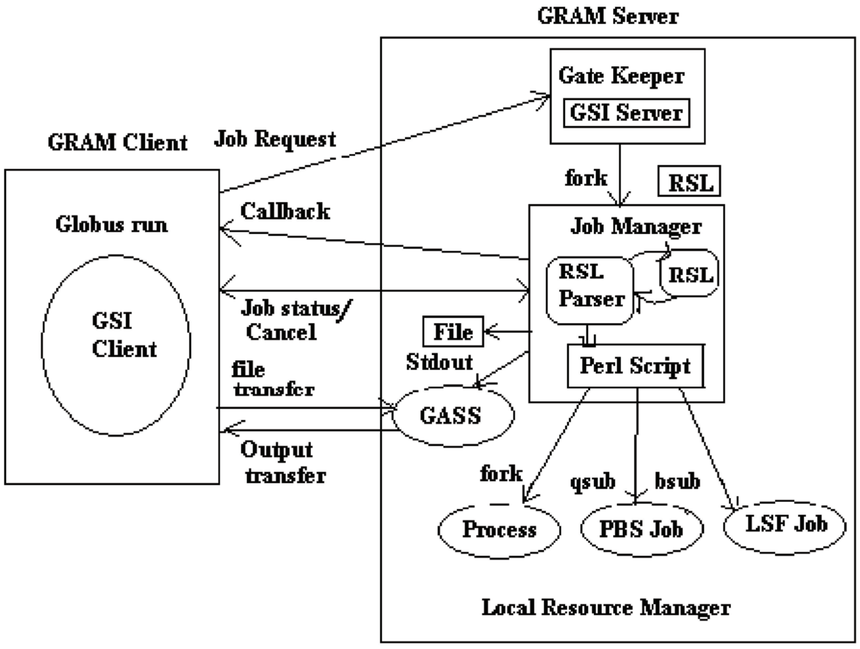


Figure 2.7. Overview of GRAM.

This functionality is implemented in a component called the Monitoring and Directory System **(MDS)** [20].

Figure 2.8 represents the conceptual view of interconnection of the MDS components. Based on the Lightweight Directory Access Protocol (LDAP), the Grid Resource Information Service (GRIS) and Grid Index Information Service (GIIS) components can be configured in a hierarchy to collect the information and distribute it. These two services are called the Monitoring and Discovery Service (MDS). The

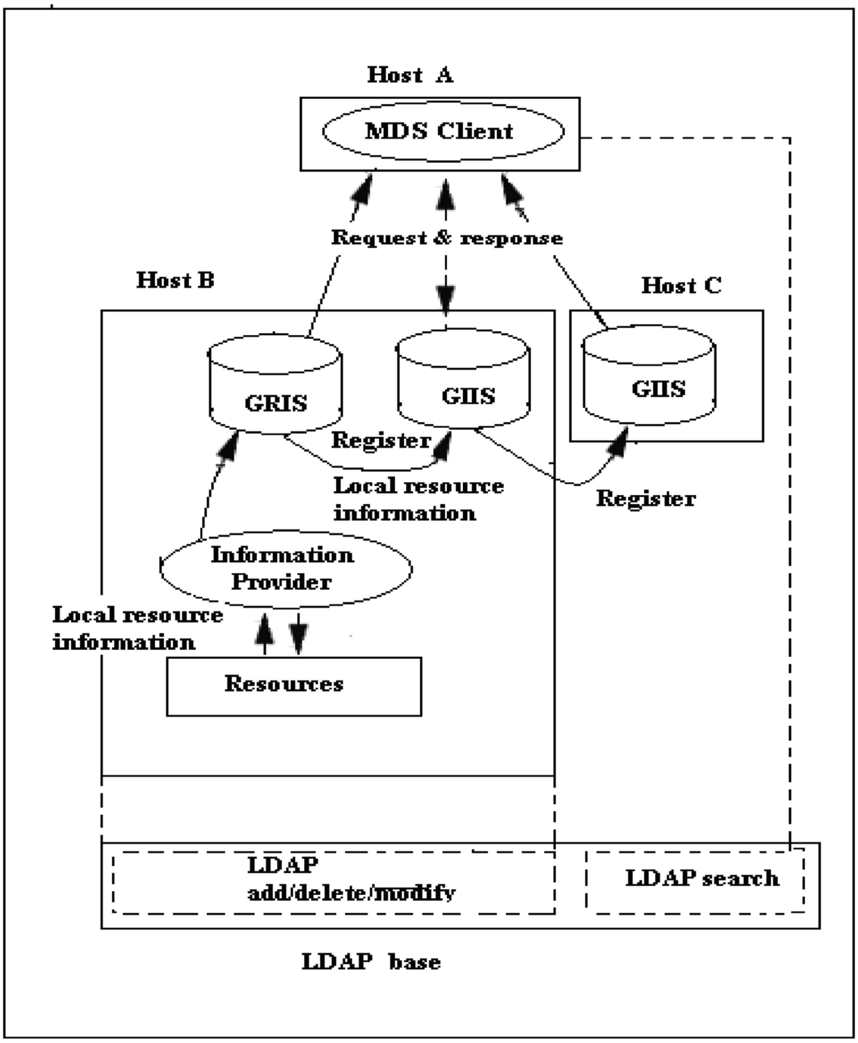


Figure 2.8. Overview of MDS.

information provider collects resource information in the form of objects which represent Infrastructure components such as the name of the job manager or the name of the running job and Computer resources such as network interface, IP address, or memory size and it is passed to GRIS. The information collected can be static information about the machines as well as dynamic information showing current CPU or disk activity. The information providers translate the properties and status of local resources to the format defined in the schema and configuration files. GRIS stores the resource information derived from information providers. GRIS registers its local information with the GIIS, which also registers with another GIIS, and so on. MDS clients can get the resource information directly from GRIS for local resources and from GIIS for grid-wide resources. Clients can query the GIIS using LDAP query language to retrieve the desired information about resources that build a grid environment [69].

Data management services

Data management services handle file transfers and file transfer management. This functionality is implemented in GridFTP [4], a secure and reliable grid data transfer protocol. The Globus toolkit provides an implementation of both the GridFTP client and the GridFTP server. GridFTP is a protocol intended to be used in all data transfers on the grid. It is based on FTP, but extends the standard protocol with facilities such as multi streamed transfer, auto-tuning, and Globus based security. A set of GridFTP tools is distributed with Globus as additional packages. Globus Project has selected some features and extensions defined already in IETF RFCs and added a few additional features to meet requirements from current data grid projects. Globus Toolkit provides the GridFTP server and GridFTP client, which are implemented by the in.ftp daemon and by the globus-url-copy command, respectively. They support most of the features defined in the GridFTP protocol. The GridFTP server and client support two types of file transfer: standard and third-party. The standard file transfer is where a client sends the local file to the remote machine, which runs the FTP server, as illustrated in Figure 2.9. Third-party file transfer is where there is a large file in remote storage and the client wants to copy it to another remote server, as illustrated in Figure 2.10. The Globus Toolkit provides a set of tools to support the GridFTP type of data transfers. The gsi-ncftp package is one of the tools used to communicate with the GridFTP Server. The GASS API package is also part of the GridFTP tools. It is used by the GRAM to transfer the output file from servers to clients [69].

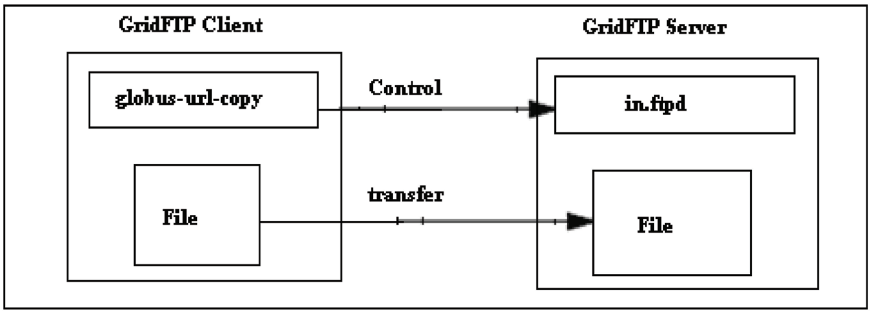


Figure 2.9. Standard File Transfer.

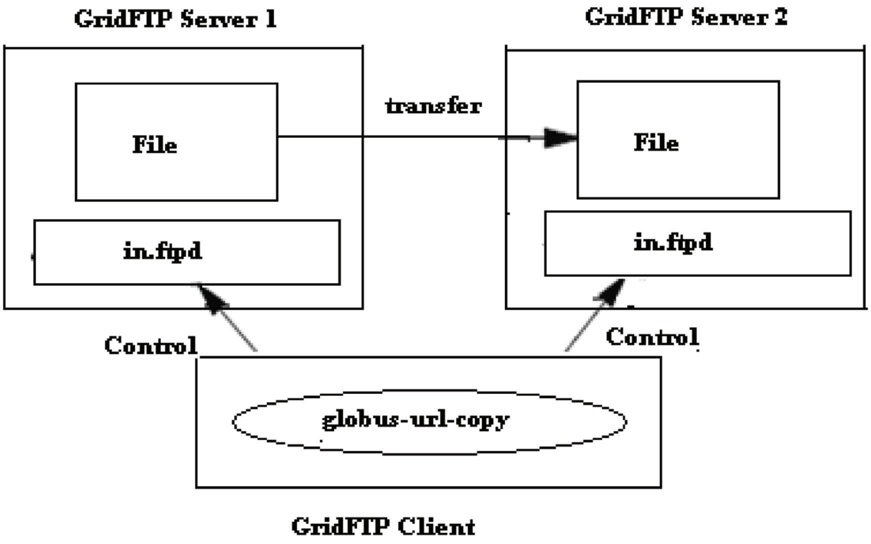


Figure 2.10. Third-Party File Transfer.

Grid Security Infrastructure (GSI)

GSI provides elements for secure authentication and communication in a grid. The infrastructure is based on the SSL protocol (Secure Socket Layer), public key encryption, and x.509 certificates. For a single sign-on, Globus adds some extensions on GSI. It is based on the Generic Security Service API which is a standard API promoted by the Internet Engineering Task Force (IETF). The main functions implemented by GSI are Single/mutual authentication, Confidential communication, Authorization and Delegation.

UNICORE

UNICORE is an open architecture based on the concept of object oriented programming. It has consistent security architecture, minimal interface with local administrative procedures and exploitation of the existing and emerging technologies including Web and Java. The UNICORE middleware has client side software and server side software. The UNICORE Client assists in creating, manipulating and managing complex, interdependent multi-system jobs, multi-site jobs, synchronization of jobs and movement of data between systems, sites and storage spaces. The client creates an Abstract Job Object (AJO) represented as a serialized Java Object or in XML format. The UNICORE Server (NJS) manifest the AJO into a target system specific actions, synchronizes action and work flows, transfers, job and data between User Workstation, Target Systems and other sites and Monitors the status.

The UNICORE transfers seamless specification of the work to be done to the remote site and transfer results and related data. The seamless specification in UNICORE is dealt with by a collection of Java classes known loosely as the AJO (Abstract Job Object). The UNICORE Protocol Layer (UPL) is designed as a protocol that transmits data regardless of its form. The classes concerned with the UPL are included in the org. unicore.package and the org.unicore.upl package, with some auxiliary functions from the org.unicore.utility package [67].

NIMROD-G GRID RESOURCE SCHEDULER

Nimrod-G, a software system from Nimrod and Globus is a computational economy-based global Grid resource management and scheduling system that supports deadline and budget-constrained algorithms for scheduling task and parallel execution of applications on distributed resources. Nimrod-G also provides a persistent Task-Farming Engine (TFE), which supports job management protocols and APIs. Nimrod has an associated dispatcher capable of deploying computations (jobs) on Grid resources. The most remarkable property of Nimrod is that, it can be used to create and plug-in user-defined scheduling policies, algorithms and customized problem solving environments. Figure 2.11 shows Architecture of Nimrod-G [67].

CONDOR AND CONDOR-G

Condor-G is a job management service for grid applications which is derived from Condor and Globus. This is a mixture of inter domain

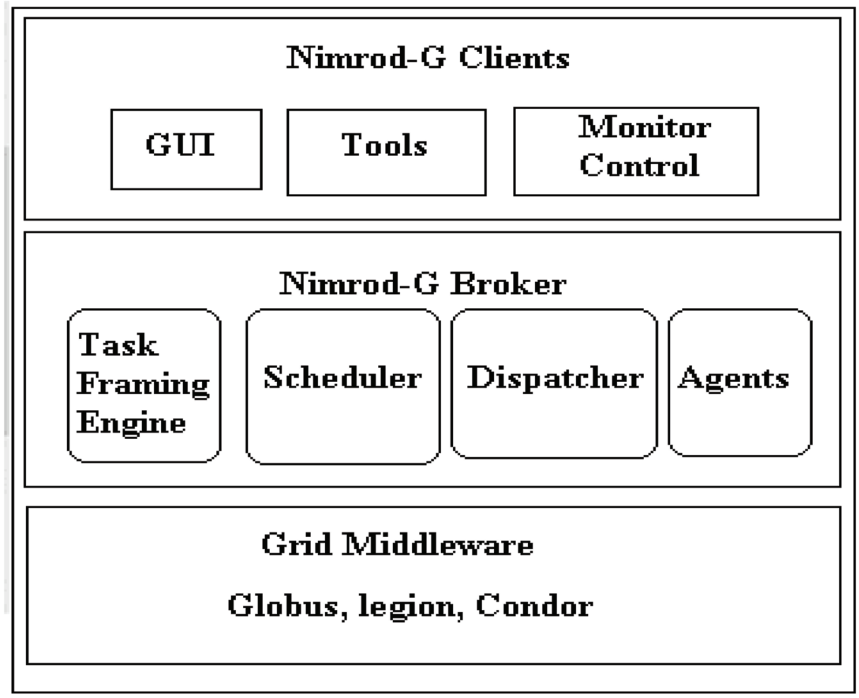


Figure 2.11. Architecture of Nimrod-G.

resource management protocols of Globus with the intra domain resource management methods of Condor.

Condor-Globus Job manager

Condor harnesses the capacity of idle workstations for computational tasks. The Condor is compatible for parameter studies and high throughput computing. Condor can be classified as a specialized workload management system for compute-intensive jobs. Condor provides a job queuing mechanism, scheduling policy, priority scheme, resource monitoring, and resource management like other full-featured batch systems. The Condor system places the jobs in a queue upon receiving serial or parallel jobs from the user. It then chooses when and where to run the jobs based on policies and carefully monitors their progress. It ultimately informs the user upon completion. Condor can be used to manage a cluster of dedicated compute nodes.

Figure 2.12 shows a sample usage of Condor-G in combination with Globus. As shown, Condor-G contains a GASS Server, which is used

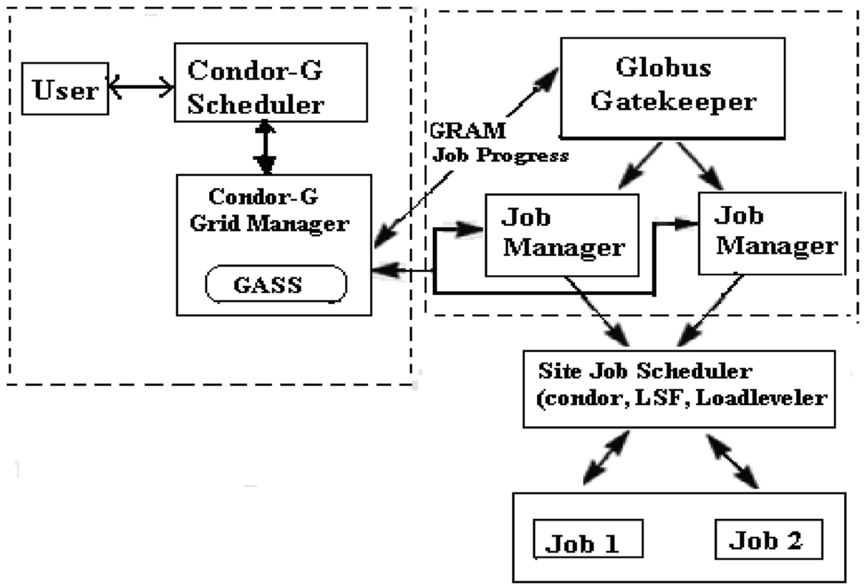


Figure 2.12. Remote Execution of Condor-G.

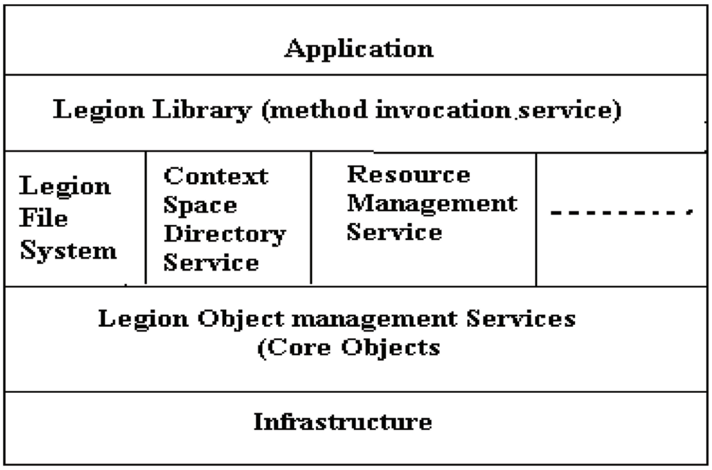
to transfer jobs to and from the execution centre. The Condor-G Grid manager uses GRAM to get the Job progress information from the Globus Gate Keeper.

Condor software is used by commercial and scientific organizations. The major scientific projects that utilize Condor, includes NSF Middleware Initiative (NMI), the Grid Physics Network (GriPhyN), International Virtual Data Grid laboratory (iVDGL), TeraGrid, and so on. Some of the major commercial companies where condor software is applied are Micron Technologies, CORE Digital Pictures, and NUG30 Optimization Problem Solver [67].

LEGION

Legion is a middleware project initiated by the University of Virginia. Generally, Middleware is a software layer between the operating system and the applications that provides a higher degree of abstraction in distributed programming. A Legion Grid consists of workstations, vector supercomputers, parallel supercomputers and specialized equipment connected by a variety of networks. Legion deals with a wide range of distributed systems problems, including application-adjustable fault-tolerance, wide area parallel processing, interoperability, scalability, security, efficient scheduling, and comprehensive resource management. It is object-based Meta system software for grid applications. The Legion project promotes the principle design of distributed system software by providing standard object representations for processors, data systems, file systems, and so on. Legion applications are developed in terms of these standard objects. Group of users can construct a shared virtual workspace to collaborate on research and exchange information. Legion's fundamental object models are described using Interface Description Language (IDL), and are compiled and linked to implementations in a given language. This approach enables component interoperability between multiple programming languages and heterogeneous execution platforms. The objects in legion can communicate with one another regardless of location, heterogeneity, or implementation details.

Figure 2.13 shows the architecture of a Legion system. Legion sits on top of the user's operating system and acts as mediator between its own host(s) and other required resources. Legion's scheduling and security policies act on behalf of user in undertaking negotiations with outside systems and system administrators. Legion offers a user-controlled naming system called context space, with which users can easily create and use objects in distributed systems. The Grid tools and test bed that make use of the Legion as their low-level middleware include: NPACI Testbed, Nimrod-L and NCBioGrid [60]. The major disadvantage of the legion is that, it does not support the dynamic behavior of the Grid. The Legion has considerable issues when working with mobile devices [67].


Application
Legion Library (method invocation service)
Legion File System
Context Space Directory Service
Resource Management Service

Legion Object management Services (Core Objects
Infrastructure

Figure 2.13. Legion Application Architecture.

GRIDBUS

The Gridbus Project [17] is an open-source, multi-institutional project led by the GRIDS Lab at the University of Melbourne. It develops service-oriented cluster and grid middleware technologies to support e-Science and e-Business applications. It extensively leverages related software technologies and provides an abstraction layer. It hides the heterogeneous resources and low-level middleware technologies from application developers. The Gridbus uses economic models for efficient management of shared resources and promotes commoditization of their services. Thus, it enhances the tradability of the grid services and manages efficiently the supply for the demand of resources [60].

MOBILE COMPUTING

Mobile Computing can be defined as a paradigm of computing in which users who carry portable devices have access to information and

Table 2.1. Grid Middleware based on Characteristics.

Middle/ Property	**UNICORE**	**GLOBUS**	**LEGION**	**GRIDBUS**
Focus	High level Programming models	Low level services	High level Programming models	Abstractions and market models
Category	Uniform job submission and monitoring	Generic computational	Generic computational	Generic Computational
Architecture	Vertical multi tiered system	Layered and modular toolkit	Vertically integrated system	layered component and utility model
Implementation Model	Abstract Job Object	Hourglass model at system level	Object-oriented meta system	Hourglass model at user level
Implementation Technologies	Java	C and Java	C++	C, Java, C# and Perl
Runtime Platform	Unix	Unix	Unix	Unix and Windows with .NET

(*Continued*)

Table 2.1. (Continued)

Middle/ Property	**UNICORE**	**GLOBUS**	**LEGION**	**GRIDBUS**
Programming Environment	Workflow environment	Replacement libraries for Unix & C libraries, Special MPI library, CoG kits in Java, Python, CORBA, Matlab, Java Server Pages, Perl and Web services	Legion Application Programming Interfaces (API). Command line utilities	Broker Java API XML-based parameter-sweep language Grid Thread model via Alchemi
Distribution Model	Open Source	Open Source	Not open source, commercial version available	Open Source
Some Users and Applications	EuroGrid, Grid Interoperability Project	Apple, Ninf, Nimrod-G, NASA IPG, Condor-G, Gridbus Broker	NPACI Testbed, Nimrod-L, NCBioGrid	e-Physics Portal, Belle Analysis Data Grid, Nero Grid, Natural language Engineering

services through a shared infrastructure, regardless of their physical location or movement behavior. The infrastructure allows its users to access and process desired information from anytime anywhere in the space. The state of the user, either static or mobile, does not affect the information management capability of the mobile platform. The mobile computing discipline creates an illusion that the desired data and sufficient processing power are available and can be utilized without being subjected to spatial and temporal constraints. The devices that participate in mobile computing include Laptops with Wireless Local Area Networks technology, mobile phones, and Personal Digital Assistants (PDAs) with Bluetooth or Infrared Data Association (IrDA) interfaces [66].

For realizing the mobile environment, the system requires the creation of communication infrastructures and the modification of computer networks, operating systems, and application programs resource in contrast to a fixed one. (Gai, S., 1998). The algorithms pertained in this environment are inferred to treat in much detail, space and coordination of mobile systems. In particular, algorithms should consider location changes, the frequency of disconnection, power limitations and the dynamic changes in the connectivity pattern of mobile systems. Hence, communication in mobile computing is accomplished by means of Cellular technology.

EXISTING CELLULAR NETWORK ARCHITECTURE

The Mobile telephony system succeeds with the introduction of Cellular technology, which offers very efficient use of the available frequency spectrum, enabling the connection of a large number of users. The frequencies used for communication vary according to the cellular network technology implemented. For GSM, 890–915 MHz range is used for transmission and 935–960 MHz for reception. The DCS technology uses frequencies in the 1800 MHz range while PCS in the 1900 MHz range. An overall cellular network contains a number of different elements from the base transceiver station (BTS) with its antenna back through a base station controller (BSC), and a mobile switching centre (MSC) to the location registers (HLR and VLR) and the link to the public switched telephone network (PSTN). In the cellular network, the BTS provides the direct communication with the mobile phones. The links between the BTS

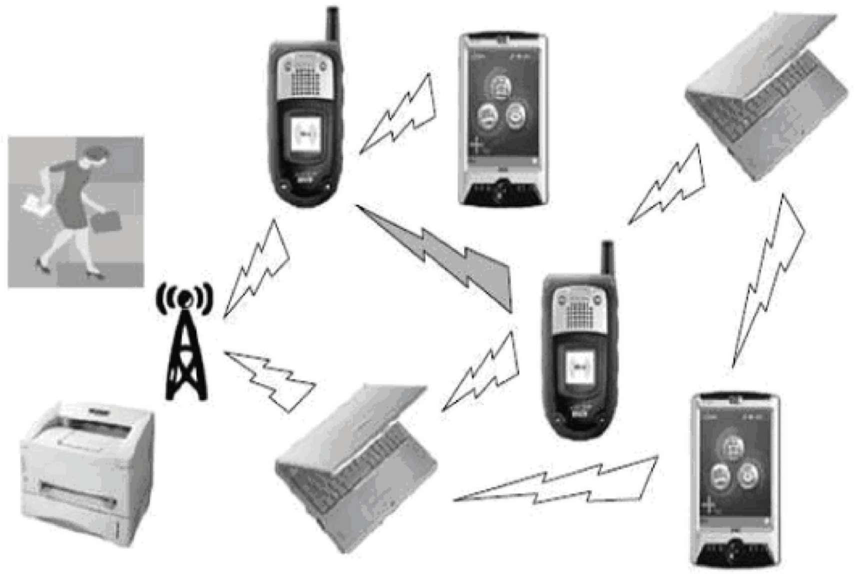


Figure 2.14. Mobile ad-hoc Network.

and the BSC may use either land lines or even microwave links. The BSC interfaces with the mobile switching centre. This acts as an interface and redirects calls to land line based PSTN as well as the HLR and VLR. Figure 2.15 shows the Architecture of Mobile Grid.

Base transceiver station BTS

The base transceiver station or system provides direct connection with the mobile devices. A BTS consists of a number of different elements. The antennas on top of masts and tall buildings enable them to cover the required area. The base of the antenna tower contains the electronics for communicating devices that includes radio frequency amplifiers, radio transceivers, radio frequency combiners, control, communication links to the BSC, and power supplies with backup. Afeeder connects the antenna to the electronic system. The electronic control system which contains control logic and software connects base station and its controller.

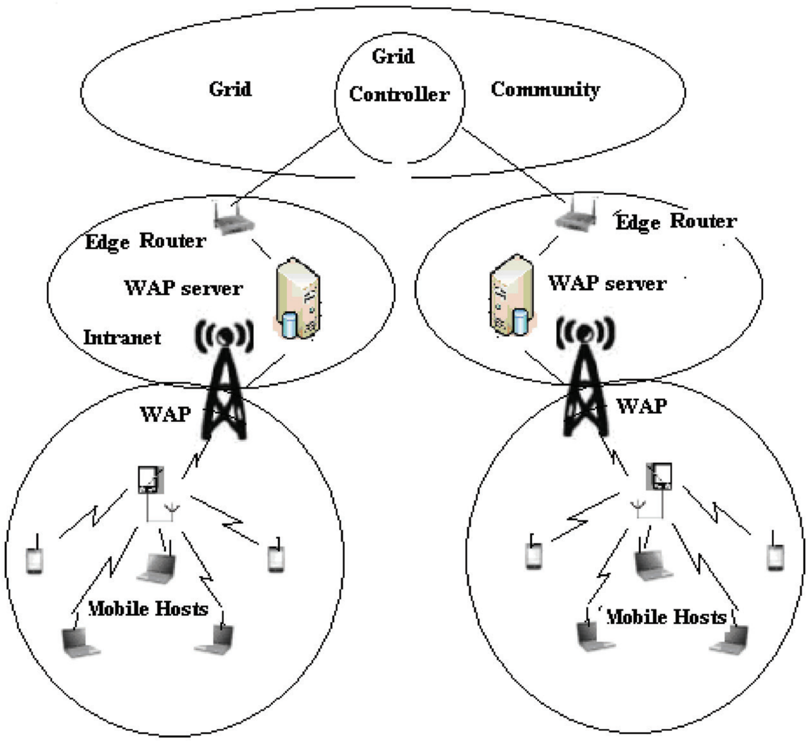


Figure 2.15. Architecture of Mobile Grid.

Mobile switching centre (MSC)

The MSC is the control centre of the cellular system. It coordinates the actions of BSCs. It acts as a switch in the public telephone network. The MCS controls and route calls to the BSC through communication links like fiber-optic, microwave links and some copper wire cables. It maintains databases, namely Home register and Visitor Location Register which gives detailed information about the last known locations of the mobiles. It also contains Authentication information for endorsing the mobile devices into the network. In addition to this it also contains the facilities to generate the billing information for the individual accounts. In view of the importance of the MSC, it contains reliable backup units that prevent whole MSC from failures.

Cellular registration

There are few incredible steps that need to be undertaken when a phone is turned on. The first step is the software startup for the phone that it takes a few seconds from switching on the phone before it is ready for use. Next the registration process. During the registration process, the mobile makes contact with the base station using "attach" message through a paging or control channel to register itself so as to allow it to make use of the network. Registration also allows only those mobiles that have permission to access the network to communicate with it.

Once accepted onto the network, two further registers are normally required. These are the Home Location Register (HLR) and the Visitors Location Register (VLR). These two registers are required to keep track of the mobile so that the network knows where the mobile device is. This information allows it to route the calls to the correct base station or general area of the network. The registration is updated periodically. The mobile, even in its idle state periodically communicates with the network to update its position and status.

When the mobile is switched off it sends a detach message. This informs the network that it is switching off and enables the network to update the last known position of the mobile. Thus, by undergoing a registration procedure when the mobile is turned on, the cellular network is able to communicate correctly with it, provide access for outgoing calls, and also route any incoming calls to it in the most efficient manner.

Cellular roaming

By carrying out a registration procedure when the mobile is turned on the cellular network is able to communicate correctly with the Base station providing access for outgoing calls and route incoming calls in a more efficient manner. However, when the mobile handset moves out of one cell to the other, it must likely hand over the call from the current base station of the cell to the next with no disruption to the call. This process is called as cellular handover in Europe and cellular handoff in North America.

The cellular network needs to decide when and where handover or handoff is necessary. Also, when the handover occurs, it is necessary to re-route the call to the relevant base station along with changing the communication between the mobile and the base station to a new channel. All of this needs to be undertaken without any noticeable interruption to the call.

The signal strength is the most prominent parameter that needs to be known to determine whether a handover is required. The signal strength between the mobile device and the current base station and surrounding stations is monitored by the BSC. Additionally the availability of channels is also monitored. This information is fed back to the base station. When the signal strength of the base station to that of the mobile device starts to fall to a level, the cellular network looks at the reported strength of the signals from other cells to the mobile device. It checks the channel availability and informs the new cell to reserve a channel for the incoming mobile. When the channel is ready, the current base station passes the information for the new channel to the mobile. The mobile now belongs to the new cell. Once the mobile sends a message on the new channel to inform the network it has arrived, the network shuts down communication with the mobile on the old channel, freeing it up for other users, and all communication takes place on the new channel. During the switch over from one cell to another, the line is lost for about 400ms. Its location information is updated, in both home network and foreign network [68].

Challenges in Mobile computing

The mobile computing faces few challenges because of the nature of communication channel, mobility and the fabrication of mobile devices. Some of the challenges are of mobile computing are:

Communication challenges

In mobile computing, wireless medium acts as channel between devices. The vulnerabilities of the wireless environment are interaction of radio signals, blocking of signals and introduction of echoes and noises. These issues cause communication challenges such as lower bandwidths, higher error rates and frequent disconnections.

Mobility challenges

Mobility results in instability in the information transmitted by the mobile devices, data considered static for stationary systems become dynamic in Mobile Computing, the network address of the mobile device changes dynamically, the communication bandwidth varies according to its position and finally the current location affects configuration parameters.

Device challenges

The Mobile devices need to be small, light, durable and operational under wide environmental conditions and they require minimal power usage for a long battery life. The mobile systems and applications designed should agree with mobile devices.

Security Challenges

Mobile Computing requires special security requirements, particularly with regard to identification and certification. Mobile Technology may hide the identities of the malicious communication parties. Hence, privacy and security are at risk while utilizing mobile devices and insecure channels.

Performance challenges

These are twofold. Firstly, mobile devices are usually less powerful compared to the static counterparts. This is mostly due to constraints in size, weight, and ergonomics for mobile devices. Second, mobility greatly slows down traditional optimization.

The availability of portable computing devices and advances in wireless networking technologies have contributed to the growing acceptance of mobile computing applications and opened the door for the possibility of seamless and pervasive services in mobile environments. However, due to limited device capabilities, network connectivity, transmission range, and device mobility, a considerable burden is placed on applications that are deployed in a mobile environment. Upon solving the few set backs of mobile devices, a wonderful mobile computing environment can be attained [61, 62].

MOBILE GRID COMPUTING

Mobile Grid is the emerging grid technology of Grid computing used for scientific and engineering applications where large amounts of data processing units and storage devices are handled. The purpose and usage of Mobile Grid are enormous. The main advantage of Mobile Grid is the leveraging of Grid facilities anytime anywhere. The mobile grid maintains session even when they move from the limited location to another. It supports mobile users and resources in a seamless, transparent, secure and efficient manner. Figure 2.16 shows Mobile Grid computing network.

Mobile Grid enables the mobile devices to act as users as well as resource providers. Both cases have their own limitations and constraints. In the first case the devices of the mobile users act as interfaces to the Grid enabling job submission, monitoring and management of the activities in an 'anytime, anywhere' mode, while the Grid provides them with a high reliability, performance and cost-efficiency. In the second case as Mobile Grid resource providers, since the performance of current

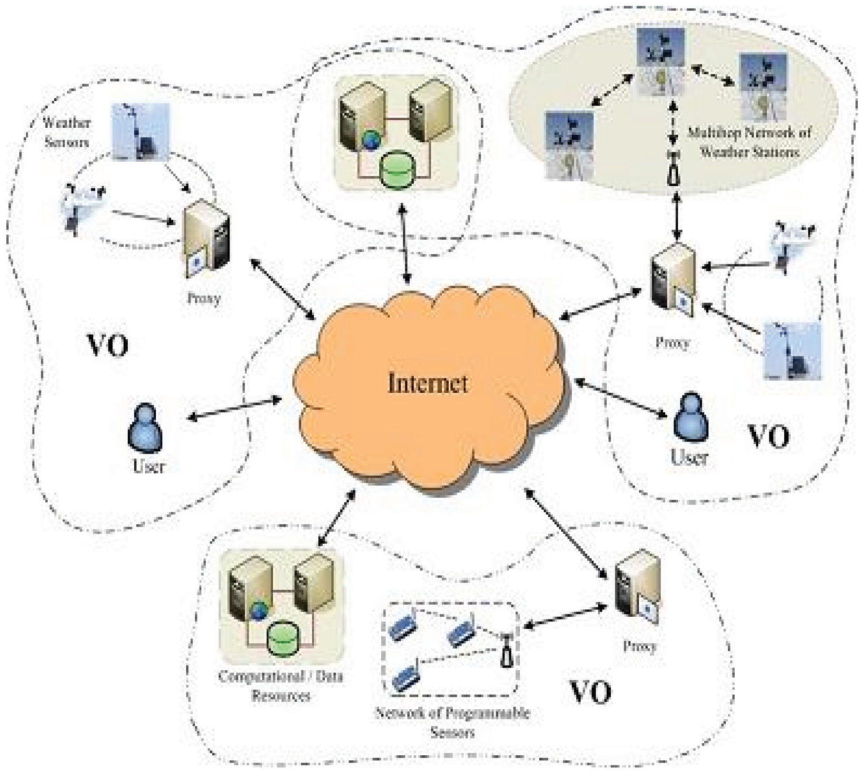

Weather Sensors
Proxy
VO
User
Multihop Network of Weather Stations
Proxy
Internet
User
VO
Computational / Data Resources
Network of Programmable Sensors
Proxy
VO

Figure 2.16. Mobile Grid Computing.

mobile devices is significantly improved Laptops and PDAs can provide the aggregated computational capability when gathered in hot spots forming a Grid on Site [66].

However, Programming in a mobile environment is a very tedious for a developer, who has to manage all the features related to the network communication such as bandwidth, frequent signal loss, noise level and signal loss due to handover. The common issues of mobile devices are CPU, RAM, storage concurrency control, transaction management, addressing, error handling and data representation. The mobile devices are characterized by several other limitations and constraints. The processing power is low in most of these devices, the built-in memory is low, battery lifetime is short and the storage capacity is very limited. Despite the limitations, the Mobile Grid computing has advantages that include mobile-to-mobile and mobile-to-desktop collaboration for resource sharing, improving user experience, convenience and contextual relevance and novel application scenarios. Figure 2.17 shows Dynamic and Fixed Wireless Grid. A mobile-based Grid environment would allow mobile devices to become more efficient by offloading resource-demanding work to more powerful devices or computers [64].

Traditional middleware solutions are inadequate to manage the issues discussed in the previous paragraph. A special middleware is to be designed which works between the operating system and the applications that provides a higher degree of abstraction in a mobile environment. The middleware should hide the details of services and components when the session movement is required and maintains session handoff transparently to the user and possibly to the application components [59]. Some of the Mobile Grid requirements are discussed below.

MOBILE GRID REQUIREMENTS

Traditional middleware solutions are adequate for static context. However, the middleware for mobile environment should be reconfigured to the changes in operating context that provide mobility support. The mobile middleware should meet the requirements such as:

i. To search for the requested service's actual position.
ii. To select the ones, more appropriate resource for the user.
iii. Establish and rate the quality of service (QoS) level to be used.
iv. Allocating the resources in advance for using the service.
v. Managing the changes of address.
vi. Fault tolerance Environment.
vii. Providing security to the data as well as resources.

viii. To manage the load so as to optimize the resources by changing the server from which the streaming will be retrieved.
ix. Format the information to be presented according to the hardware and Software features of the device used.

The system should also be capable to support computation and service migration, replication, run-time monitoring, and application recovery [63].

BASIC CONCEPTS OF MOBILE GRID MIDDLEWARE

The mobile middleware adopts the behaviour of an RPC. RPC middleware was created to give developers some kind of mechanism for accessing remote resources using just a local procedure call to hide all the details on the connection setup, marshal all the parameters, and hide all the problems related to the heterogeneity of different platforms. When the client performs a call, it is intercepted by the middleware, which runs the code for gathering the parameters, opening a socket, and so on. The middleware generates an appropriate message containing all the data related to that call and transmits it to the server, which is able to understand the request sent

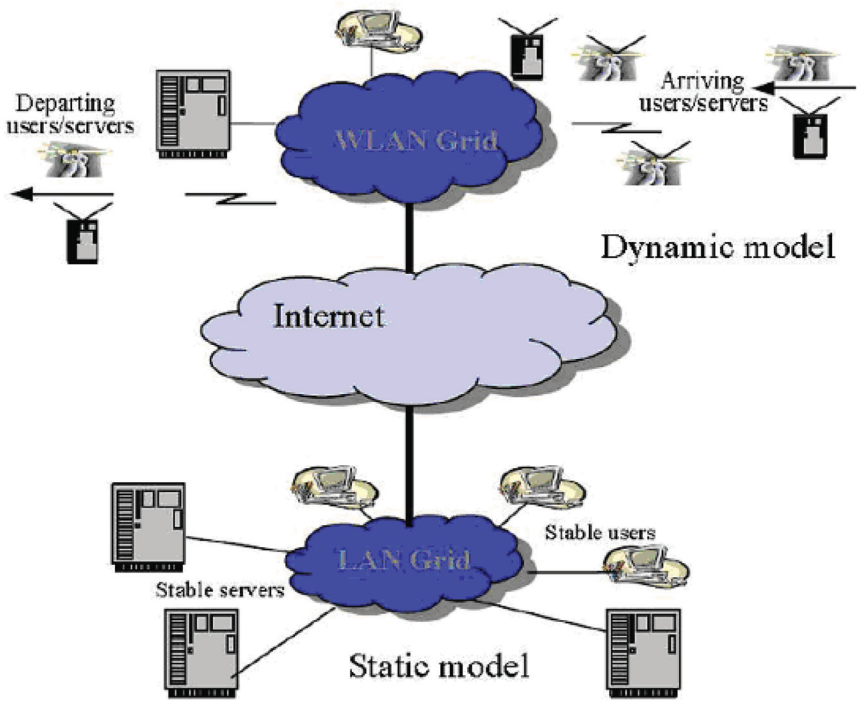


Figure 2.17. Dynamic and Fixed Wireless Grid.

from the client and execute the procedure. When the results are ready, they are sent back to the client middleware, which returns to the application. The user is completely unaware of what happens. Figure 2.18 shows an RPC model for constructing middleware [64].

The following paragraphs discuss different middleware for the mobile Grid environment.

Mobile OGSI.NET

Mobile OGSI.NET extends an implementation of Grid computing for Mobile Environment. This work was done by the National Partnership for Advanced Computational Infrastructure (NPACI), and Microsoft Research. It addresses the issues related to resource limitations and intermittent network connectivity which differentiates from traditional computers.

Mobile OGSI.NET Architecture

The Mobile OGSI.NET architecture consists of three main layers: The Mobile Web Server that manages end to end message reception and transmission. The Grid Services Module, which handles parsing and multiplexing messages to the appropriate Grid Service. The Grid Service, which carry out applications, logic and processing. Figure 2.19 illustrates this system architecture.

Mobile Web Server

The Mobile Web Server is an HTTP server for sending and reconstruction of messages. The Mobile Web Server functions for

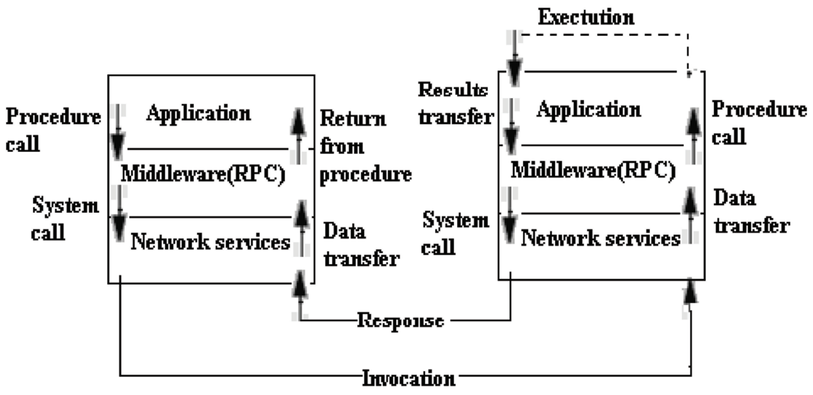


Figure 2.18. Remote Procedure call (RPC).

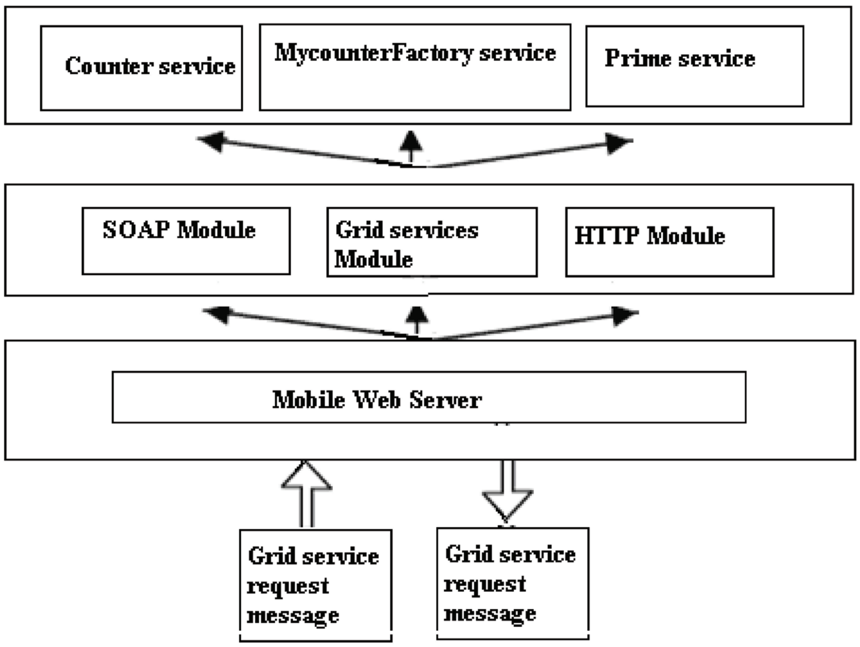


Figure 2.19. Mobile OGSI.NET Architecture.

both traditional HTTP requests and advanced SOAP requests. The traditional web servers support only one request at a time, whereas the Mobile Web Servers support thousands of simultaneous requests by thread pool model. The server also supports dynamic Grid services. The Mobile Web Server de-multiplexes the incoming messages and sends to appropriate processing modules such as the HTTP Module and SOAP Module.

Grid Services Module

The Grid Services Module supports the GXA family of protocols such as WS-Security which supports message authentication and encryption and WS-Addressing which de-multiplex Grid Service. This module uses single contact object rather than semantically unorganized details like Contact Name, Contact Phone, Contact Street, Contact City, etc.

Mobile Grid Services

The Grid Service involves basic services such as opening files, socket connections and other local host resource functions. In addition to this it supports mobility features which allows services to migrate from host to host, allows Grid Service state saving and loading and Battery power constraints of mobile devices.

The Grid Service can save its state which another Grid Service of the same type can then load and continue its execution process. It uses SaveCounterState() method to save the Grid service state and loads the state object via LoadCounterState(). The Grid service upon detecting less than 20% battery power remaining on Host A, the GroupManagerService initiates calls to migrate Service X on Host A to Host B. The migration process from one host to another is done via GroupManagerService, which either periodically or on an event basis, examines host and service metadata. Thus Mobile OGSI.NET middleware supports Mobile Grid Environment.

Grid-M Middleware for Mobile Grid Computing

Grid-M is a middleware designed for enabling mobile computing devices to participate in the Grid Computing environment. This was designed at Federal University of Santa Catarina, Florianópolis, Brazil. This Grid-M acts as a platform for connecting Java-developed applications with the devices. It follows protocol standard to enable interoperability, common infrastructure and deliver nontrivial qualities of service. Figure 2.20 depicts the Grid-M's node architecture whose modules are detailed below:

The communication module receives packets through an HTTP-server. It converts the XML-representation into internal data structures. It then passes the received information to the Dispatcher. The DISPATCHER module contains sub-routines which dispatch the received messages to the plug-in services. The service interface module sets call-backs to service execution entry points. New services can be plugged in through SERVICE INTERFACE. The SCHEDULER MODULE generates communication requests in establishing periods of time. The Grid service may be of a single host or a cluster of nodes. It can be implemented as a distributed service throughout the system. The MESSAGE ROUTER module implements the routing of communication messages, through name resolution and accessing information about physical network address from the directory database. Moreover, in the case of Grids of Mobile and Embedded devices some of these devices might not be directly networked or available as port listener. This module implements the message routing to these devices which might require proxy, store-and-forward facility or other techniques. GRID BROKER connects third-party Grid Services. It promotes the message translation and encoding throughout different grid services (i.e. gateway).

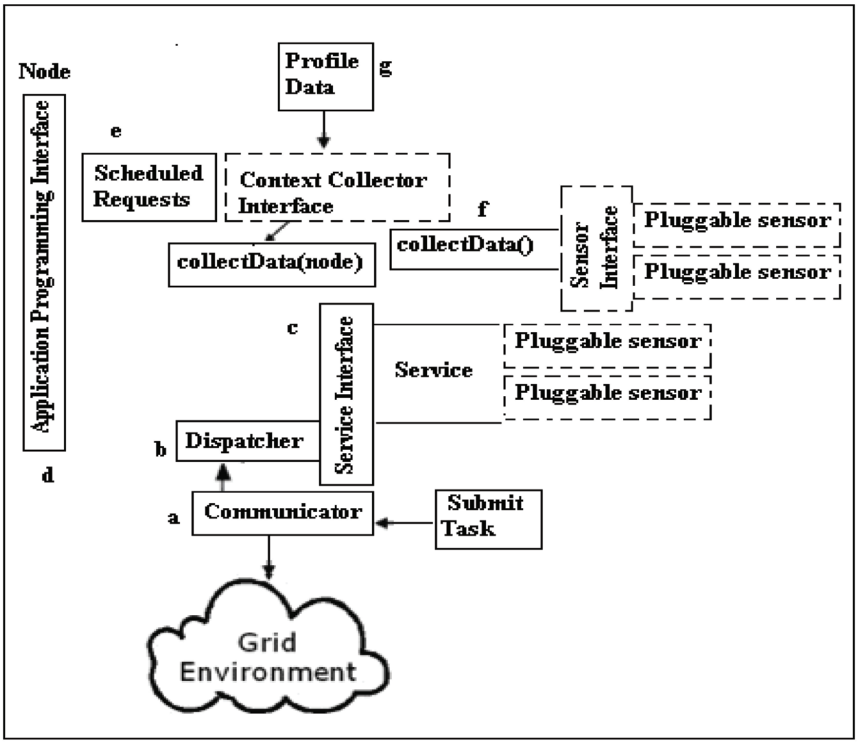


Figure 2.20. Grid-M node Architecture.

The Table 2.2 summarizes the support provided by different middleware services for mobile computation. From the analysis, we concluded that the existing packages for development of Grids Computers don't supply the needs for the creation of mobile services. They lack to support characteristics such as collaboration support, context sensibility support, allocation resources support, dynamic environment support and mobile execution support.

THE MAJOR ISSUES OF MOBILE GRID ENVIRONMENT

The following ponder some of the issues in Mobile Grid Environment.

i. Resources or Service discovery.
ii. Job Scheduling.
iii. Fault Tolerance Environment.
iv. Energy constraints due to Battery Power.

Table 2.2. Comparison of Mobile Grid Middleware.

	GRID-M	GLOBUS	GRIDBUS	LEGION	UNICORE
Collaboration Support	Yes	Yes	Yes	Yes	Yes
Context Sensibility support	Yes	No	No	No	No
Allocation resources support	Yes	Yes (GRAM)	Yes	Yes (LOA)	Yes (IACT)
Dynamic environment support	Yes	No	No	No	No
Mobile device execution support	Yes	No	Yes	No	No

v. Security concerning data transmission and resources.
vi. Limited resources (such as CPU speed or memory).
vii. Mobile device owner's privacy and personal data protection
viii. Encouraging mobile device owner to connect their devices to a Grid.