Chapter 3

Service-Oriented Grid Architecture for Distributed Computational Economies

This chapter identifies challenges in managing resources in a Grid computing environment and proposes computational economy as a metaphor for effective management of resources and application scheduling. To realize this, we propose a framework called Grid Architecture for Computational Economy (GRACE) that leverages the existing technologies and provides additional services for resource trading and aggregation. We present related systems, both historical and emerging, for cooperative and competitive trading of resources such as CPU cycles, storage, and network bandwidth. The chapter discusses the use of real-world economic models and strategies such as commodity market, posted prices, bargaining, tendering, auction, proportional resource sharing, and cooperative bartering for resource management and scheduling within the GRACE framework. For each model, high-level protocols and strategies, that can be used by both resource owners and consumers to meet their respective objectives and goals, will be highlighted.

3.1 Introduction

Computational Grids [48] and Peer-to-Peer (P2P) [4] computing systems are emerging as a new paradigm for solving large-scale problems in science, engineering, and commerce. They enable the *sharing* and *aggregation* of millions of resources (e.g., SETI@Home [141]) geographically distributed across organizations and administrative domains. They comprise heterogeneous resources (PCs, work-stations, clusters, and supercomputers), fabric management systems (single system image OS, queuing systems, etc.) and policies, and applications (scientific, engineering, and commercial) with varied requirements (CPU, I/O, memory, and/or network intensive). The resources are owned by different organizations with their own management policies, usage and cost models for different users at different times. Also, the availability of resources and the load on them dynamically varies with time.

In such Grid environments, the *producers* (resource owners) and *consumers* (resource users) have different goals, objectives, strategies, and supply-and-demand patterns. More importantly both resources and end-users are geographically distributed with different time zones. In managing such complex environments, traditional approaches to resource management, that attempt to optimize system-wide measure of performance, cannot be employed. Traditional approaches use centralized policies that need complete state information and a common fabric management policy, or a decentralized consensus based policy. Due to the complexity in constructing a Grid environment, it is impossible to define an acceptable system-wide performance matrix and common fabric management policy [22]. (The concepts discussed in this chapter apply to both P2P and Grid systems although we can argue about some of their technical, social, and political differences. However, the term Grid will be used for simplicity and brevity).

We propose and explore the use of an economic framework, called Grid Architecture for Computational Economy (GRACE), for managing resources and scheduling applications in Grid computing environments. The economic approach provides a fair basis in successfully managing the decentralization and heterogeneity that is present in human economies. Competitive economic models provide algorithms/policies and tools for resource sharing or allocation in Grid systems. These models can be based on bartering or prices. In the *bartering-based model*, all participants need to own resources and trade

resources by exchanges (e.g., storage space for CPU time). In the *price-based model*, the resources have a price, based on the demand, supply, value, and the wealth in the economic system.

Most of the related systems for Grid resource management and scheduling (such as Legion [121], Condor [79], AppLeS PST [28][42], NetSolve [41], PUNCH [85], and XtremWeb [32]) adopt a *conventional strategy*, where a scheduling component decides which jobs are to be executed at which resource based on cost functions driven by system-centric parameters. They aim to enhance the system throughput, utilization, and complete execution at the earliest possible time rather than improving the utility of application processing. They do not take resource access cost (price) into consideration, which means that the value of processing applications at any time is treated the same, which is not the case in reality—the value should be higher when there is a production schedule deadline. The end user does not want to pay the highest price but wants to negotiate a particular price based on the demand, value, priority, and available budget. In an economic-based approach, the scheduling decisions are made dynamically at runtime and they are driven and directed by the end-users requirements. Whereas a conventional cost model often deals with software and hardware costs for running applications, the economic model primarily charges the end user for services that they consume based on the value they derive from it. Pricing based on the demand of users and the supply of resources is the main driver in the competitive, economic market model. Therefore, in the Grid environments, a user is in competition with other users and a resource owner with other resource owners.

The main contribution of this chapter is to provide a generic distributed computational economy for Grids along with system architecture and policies for resource management for different economic models. Currently, the user community and the technology are still rather new and not well accepted and established in commercial settings. However, we believe the Grid can become established in such settings by providing incentive to both consumers and resource owners for being part of the Grid. Since the Grid uses the Internet as a carrier for providing remote services, it is well positioned to create a cooperative problem solving environment, and means for sharing computational and data resources in a seamless manner. Up to now, the idea of using Grids for solving large computationally intensive applications has been more or less restricted to the scientific community. However, even if, in the scientific community, the pricing aspect seems to be of minor importance, funding agencies need to support the hardware and software infrastructure for Grids. Economic models can help them manage and evaluate resource allocations to user communities. For example, the system managers may impose quota limitations and value different resources with a different number of tokens [56]. In such environments, resource consumers certainly prefer to use economic driven schedulers to effectively utilize their tokens by using lightly loaded, cheaper resources.

An economic approach to Grid computing introduces a number of new issues like resource trading and quality of service-based scheduling in addition to those such as site autonomy, heterogeneous substrate, policy extensibility, online control already addressed by existing Grid systems. To address these new issues, the economy based Grid systems need to support the following:

- An Information and Market directory for publicizing Grid entities
- Models for establishing the value of resources
- Resource pricing schemes and publishing mechanisms
- Economic models and negotiation protocols
- Mediators to act as a regulatory agency for establishing resource value, currency standards, and crisis handling.
- Accounting, Billing, and Payment Mechanisms
- Users' Quality of Service (QoS) requirements driven brokering/scheduling systems.

In this chapter we identify the requirements of users (resource providers and consumers) in the Grid economy and various resource management issues that need to be addressed in realizing such a Grid system. We briefly discuss popular economic models for resource trading and present related work that employs computational economy in resource management. We propose a scalable architecture and new services for the Grid that provide mechanisms for addressing user requirements. The implementation of computational economy with the Nimrod-G resource broker will be discussed in the next chapter.

3.2 Computational Economy and its Benefits

The current research and investment into computational Grids is motivated by an assumption that coordinated access to diverse and geographically distributed resources is valuable. In this paradigm, we need mechanisms that allow such coordinated access, but also sustainable, scalable models and policies that promote precious Grid resource sharing. Based on the success of economic institutions in the real world as a sustainable model for exchanging and regulating resources, goods and services, we propose a computational economy framework. Among other things, this framework provides a mechanism to indicate which users should receive priority.

Like all systems involving goals, resources, and actions, computations can be viewed in economic terms. With the proliferation of networks, high-end computing systems architecture has moved from centralized toward decentralized models of control and action; the use of economic driven market mechanisms would be a natural extension of this development. The ability of trade and price mechanisms to combine local decisions by diverse entities into globally effective characteristics, imply their value for organizing computations in large systems such as Internet-scale computational Grids.

The need for an economy driven resource management and scheduling system comes from the answers to the following questions:

- What comprises the Grid and who owns its resources?
- What motivates resource owners to contribute their resource to the Grid?
- Is it possible to have access to all resources in the Grid by contributing our resource?
- If not, how do we have access to all Grid resources?
- If we have access to resources through collaboration, are we allowed to use them for any other purposes?
- Do resource owners charge the same or different price for different users?
- Is access cost the same for peak and off-peak hours?
- How can resource owners maximize their profit?
- How can users solve their problems within a minimum cost?
- How can a user get high priority over others?
- If the user relaxes the deadline by which results are required, can solution cost be reduced?

To date, individuals or organizations that have contributed resources to the Grid have been largely motivated by the public good, prizes, fun, fame, or collaborative advantage. This is clearly evident from the construction of private Grids (but on volunteer resources) or research test-beds such as Distributed.net [24], SETI@Home [141], Condor pool [54], GUSTO [38], DAS (Distributed ASCI Supercomputer) [40], eGrid [31], and World-Wide Grid [111]. The computational resource contributors to these test-beds are mostly motivated by the aforementioned reasons. The chances of gaining access to such computational test-beds for solving commercial problems are low. Furthermore, contributing resources to a testbed does not guarantee access to all of the other resources in the testbed. For example, although we were part of the GUSTO testbed, gaining automatic access to all of its resources was not possible. Unless we have some kind of collaboration with contributors, it is difficult to get access to their resources. In this situation, we believe that a model that encourages or offers incentive for resource owners to let their resources for others' use can itself lead to computational economy. Also, as both resource owners and users want to maximize their profit (i.e., the owners wish to earn more money and the users wish to solve their problems within a minimum possible cost), the Grid computing environment needs to support this economy of computations.

Even commercial companies such as Entropia, ProcessTree, Popular Power, United Devices, and Parabon are exploiting idle CPU cycles from desktop machines to build a commercial computational Grid infrastructure based on P2P networks [97]. These companies are able to develop large-scale infrastructure for Internet computing and use it for their own financial gain by charging for access to CPU cycles for their customers, without offering fiscal incentive to all resource contributors. In the long run, this model does not support the creation of a maintainable and sustainable infrastructure, as the resource contributors have no incentive for their continued contribution. Therefore, a Grid economy seems a better model for managing and handling requirements of both Grid providers and consumers. It can be observed that, even in electricity Grids, bid-based electricity trading over the Internet has been adopted to develop competitive forces in the electricity marketplace [53].

The Grid resource management systems must dynamically trade for the best resources based on a metric of the price and performance available and schedule computations on these resources such that they meet user requirements. The Grid middleware needs to offer services that help resource brokers and resource owners to trade for resource access.

The benefits of economic-based resource management include the following:

- It helps in building a large-scale Grid as it offers incentive for resource owners to contribute their (idle) resources for others to use and profit from it.
- It helps in regulating the supply and demand for resources.
- It offers an economic incentive for users to back off when solving low priority problems and thus encourages the solution of time critical problems first.
- It removes the need for a central coordinator (during negotiation).
- It offers uniform treatment of all resources. That is, it allows trading of everything including computational power, memory, storage, network bandwidth/latency [112], data, and devices or instruments.
- It allows users to express their requirements and objectives.
- It helps in developing scheduling policies that are user-centric rather than system-centric.
- It offers an efficient mechanism for allocation and management of resources.
- It helps in building a highly scalable system as the decision-making process is distributed across all users and resource owners.
- It supports a simple and effective basis for offering differentiated services for different applications at different times.
- Finally, it places the power in the hands of both resource owners and users—they can make their own decisions to maximize the utility and profit.

3.3 Requirements for Economic-based Grid Systems

We envision a future in which economically intelligent and economically motivated peer-to-peer and Grid-like software systems will play an important role in establishing a distributed service-oriented computing paradigm. To deliver greater value to users than traditional systems, economic-based resource management systems need to provide mechanisms and tools that allow resource consumers (end users) and providers (resource owners) to express their requirements and facilitate the realization of their goals. That is, they need (i) the means to express their requirements, valuations, and objectives [*value expression*], (ii) scheduling policies to translate them to resource allocations [*value translation*], and (iii) mechanisms to enforce selection and allocation of differential services, and dynamic adaptation to changes in their availability at runtime [*value enforcement*]. Similar requirements are raised [9] for market-based systems in a single administrative domain environment such as clusters and they are limited to co-operative economic models since they aim for social welfare. Grids need to use *competitive economic models* as different resource providers and resource consumers have different goals, objectives, strategies, and requirements that vary with time.

Essentially, resource consumers need a *utility model*—to allow them to specify resource requirements and constraints. For example, the Nimrod-G broker allows the users to specify the deadline and budget constraints along with optimisation parameters such as optimise for time [*value expression*]. They need *brokers* that provide strategies for choosing appropriate resources [*value translation*] and dynamically adapt to changes in resource availability at runtime to meet user requirements [*value enforcement*]. The Nimrod-G broker, discussed in the next chapter, supports all these requirements. The resource owners need mechanisms for *price generation schemes* to increase system utilization and *protocols* that help them offer competitive services [*value expression*]. For the market to be competitive and healthy, coordination mechanisms are required that help the market reach an equilibrium price—the price at which the supply of a service equals the quantity demanded. Grid resources have their schedulers (e.g., OS or queuing system) that allocate resources [*value translation*]. Some research systems support resource reservation in advance (e.g., reserving a slot from time t1 to t2 using the Globus GARA [50] and bind a job to it) and allocate resources during reserved time [*value enforcement*]. A number of research systems have explored QoS based resource (e.g., CPU time and network bandwidth [112][3]) allocation in operating systems and queuing systems, but the inclusion of QoS into mainstream systems has been slow paced (e.g., Internet mostly uses the best effort allocation policy [71], but this is changing with IPv6 [7]).

3.4 Grid Architecture for Computational Economy (GRACE)

A distributed Grid Architecture for Computational Economy (GRACE) is shown in Figure 3.1. This architecture is generic enough to accommodate different economic models used for resource trading for determining the service access cost. The key components of the Grid include,

- Grid User with Applications (sequential, parametric, parallel, or collaborative applications)
- User-Level Middleware—Higher Level Services and Tools
  - Programming Environments
  - Grid Resource Brokers
- Core Grid Middleware (services resource trading and coupling distributed wide area resources)
- Grid Service Providers

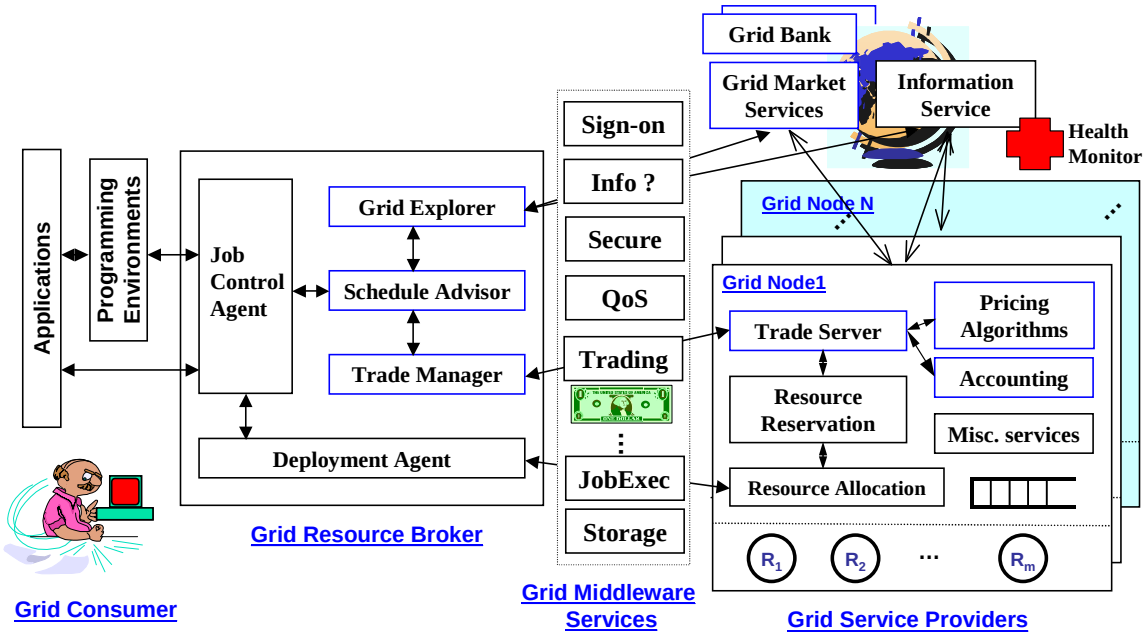

Figure 3.1: A generic Grid architecture for computational economy.

The two key players in market oriented computational Grids are *resource providers* (hereafter referred as GSPs—Grid Service Providers) and *resource consumers* (hereafter referred as GRBs—Grid Resource Broker that acts as a consumer's representative or software agent). Both have their own expectations and strategies for being part of the Grid. In the Grid economy, resource consumers adopt the strategy of solving their problems within a required timeframe and budget. Resource providers adopt the strategy of obtaining best possible return on their investment. The resource owners try to maximize their resource utilization by offering a competitive service access cost in order to attract consumers. The users (resource consumers) have an option of choosing the providers that best meet their requirements. If resource providers have local users, they will try to recoup the best possible return on "idle/leftover" resources. In order to achieve this, the Grid systems need to offer tools and mechanisms that allow both resource providers and consumers to express their requirements. The Grid resource consumers interact with brokers to express their requirements such as the *budget* that they are willing to invest for solving a given problem and a *deadline*, a timeframe by which they need results. They also need capability to trade between these two requirements and steer the computations accordingly. The GSPs need tools for expressing their pricing policies and mechanisms that help them to maximize the profit and resource utilization. Various economic models, ranging from commodity market to auction-based, can be adopted for resource trading in Grid computing environments. The Grid resource management framework and strategies for these economic models is discussed in Section 3.6.

GRACE provides services that help both resource owners and users maximize their objective functions. The resource providers can contribute their resources to the Grid and charge for services. They can use

GRACE mechanisms to define their charging and access policies and the GRACE resource trader works according to those policies. The users interact with the Grid by defining their requirements through high-level tools such as resource brokers (also known as Grid schedulers). The resource brokers work for the consumers and attempt to maximize user utility. They can use GRACE services for resource trading and identifying GSPs that meets its requirements.

As mentioned earlier, our goal is to realize this Grid economy architecture by leveraging existing technologies such as Globus and Legion and develop new services that are particularly missing in them. Therefore, we mainly focus on two things: first, to develop middleware services for resource trading using different economic models; second to use these services along with other middleware services in developing advanced user-centric Grid resource brokers. The remainder of this section presents how we are realizing the Grid economy vision and show co-existence of our modules with other systems.

3.4.1 Grid Resource Broker (GRB)

The resource broker acts as a mediator between the user and Grid resources using middleware services. It is responsible for resource discovery, resource selection, binding of software, data, and hardware resources, initiating computations, adapting to the changes in Grid resources and presenting the Grid to the user as a single, unified resource. The resource broker consists of the following components:

- **Job Control Agent (JCA):** This is a persistent control engine responsible for shepherding a job through the system. It coordinates with schedule adviser for schedule generation, handles actual creation of jobs, maintenance of job status, interacting with clients/users, schedule advisor, and dispatcher.
- **Schedule Advisor (Scheduler):** This is responsible for resource discovery (using the Grid explorer), resource selection and job assignment (schedule generation) to ensure that the user requirements are met.
- **Grid Explorer (GE):** This is responsible for resource discovery by interacting with the Grid-information server and identifying the list of authorized machines, and keeping track of resource status information.
- **Trade Manager (TM):** This works under the direction of resource selection algorithm (the schedule advisor) to identify resource access costs. It uses market directory services and GRACE negotiation services for trading with Grid service provides (i.e., their representative trade servers).
- **Deployment Agent (DA):** It is responsible for activating task execution on the selected resource as per the scheduler's instruction and periodically updates the status of task execution to JCA.

The Nimrod-G resource broker follows this architecture and offers functionalities that are expected from economic-based grid scheduling systems. It allows the users to submit their application created using its parameter specification language; and express their requirements and objectives in the form of: deadline, budget with time or cost as the optimisation parameter [value expression]. The broker uses scheduling algorithms to select resources dynamically at runtime depending on their availability, capability, and cost to meet user requirements [value translation]. It continuously adapts to changes in resource availability conditions by performance profiling (establishing job completion rate) and reschedules jobs appropriately to ensure that users requirements are met [value enforcement].

3.4.2 GRACE Framework—Leveraging Globus Tools

The Grid middleware offers services that help in coupling a Grid user and remote resources through a resource broker or Grid enabled application. It offers core services such as remote process management, co-allocation of resources, storage access, directory information, security, authentication, and Quality of Service (QoS) such as resource reservation for guaranteed availability and trading for minimizing computational cost. Many of these services are already offered by Globus [49] components and they include,

- Resource allocation and process management (GRAM).
- Resource Co-allocation services (DUROC)
- Unicast and multicast communications services (Nexus)
- Authentication and related security services (GSI)
- Distributed access to structure and state information (MDS)
- Status and Health Monitoring components (HBM)

- Remote access to data via sequential and parallel interfaces (GASS)
- Construction, caching, and location of executables (GEM)
- Advanced resource reservation (GARA)

A layered architecture for the realization of the GRACE framework is shown in Figure 3.2. It offers Grid economy infrastructure that co-exists with or built on top of the existing middleware such as Globus:

- Applications (e.g., Molecular modelling for drug design as parameter sweep application)
- Problem solving environments built on schedulers (e.g., ActiveSheets [20] on Nimrod-G)
- Programming frameworks and development tools (e.g., Nimrod parameter specification language[21])
- A resource broker (e.g., Nimrod-G)
- Various resource trading protocols
- A mediator for negotiating between users and Grid service providers (Grid Market Directory)
- A deal template for specifying resource requirements and services offers
- A trade server
- A pricing policy specification
- Accounting (e.g., QBank [124]) and payment management (GBank)

The new middleware services being proposed are designed to offer low-level services that co-exist with Globus services and infrastructure. Higher-level services and tools such as the Nimrod-G Resource Broker, which uses economic models suitable for meeting the user requirements, can use these core services.

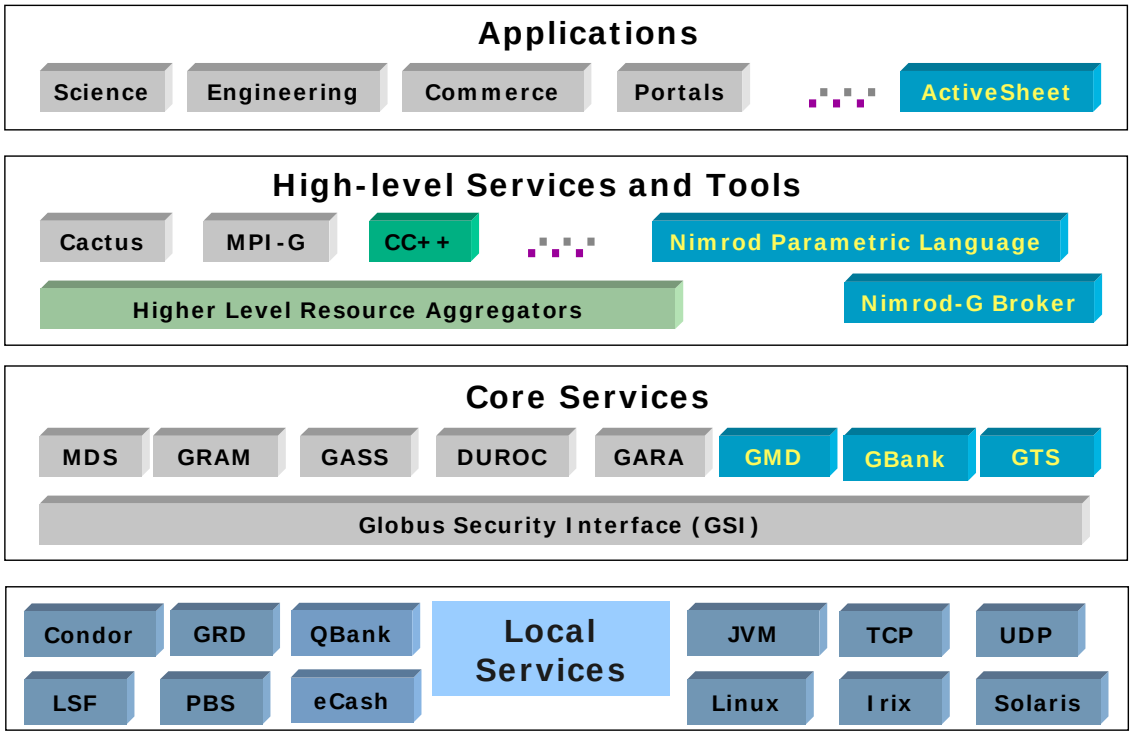

Figure 3.2: GRACE framework realization within Globus context.

The Grid service providers specifically deal with the following components along with Globus components:

- **Grid Market Directory: (GMD)**: It allows resource owners to publish their services in order to attract consumers.
- **Grid Trade Server (GTS)**: This is a resource owner agent that negotiates with resource users and sells access to resources. It aims to maximize the resource utility and profit for its owner. It consults pricing policies during negotiation and directs the accounting system for recording resource consumption and billing the user according to the agreed pricing policy.

- **Pricing Policies**: These define the prices that resource owners would like to charge users. The resource owners may follow various policies to maximise their profit and resource utilisation and the price they charge may vary with time and one user to another user. The pricing can also be driven by demand and supply like in the real market environment. That is, in this commodity market model, pricing is essentially determined by objective functions of service providers and users. The pricing policy can also be based on auction. In this auction based economic model, pricing is driven by how much users value the service and the highest bidder wins the access to Grid services.
- **Resource Accounting and Charging** components (such as GBank along with QBank) are responsible for recording resource usage and bill the user as per the usage agreement between resource broker (TM, a user agent) and trade server (resource owner agent).

The service providers publish their services through the Grid market directory (GMD). They use Grid trading services' declarative language for defining cost specification and their objectives such as access price for various users for different times and durations, along with possibilities of offering discounts to attract users during off-peak hours. The Grid trading server (GTS) can employ different economic models in providing services. The simplest would be a commodity model wherein the resource owners define pricing strategies including those driven by the demand and resource availability. The GTS can act as auctioneer if the Auction-based model is used in deciding the service access price or an external auctioneer service can be used.

3.4.3 Grid Open Trading Protocols and Deal Template

The resource trading protocols define the rules and format for exchanging commands between a GRACE client (Trade Manager), which is part of the Grid broker and a Trade Server, which is part of the Grid service providers. Figure 3.3 shows a sample multilevel negotiation protocol that both client and server need to follow while trading for the cost of resource access [99]. The wire-level (low-level) details of these protocols are skipped, as they are obvious.

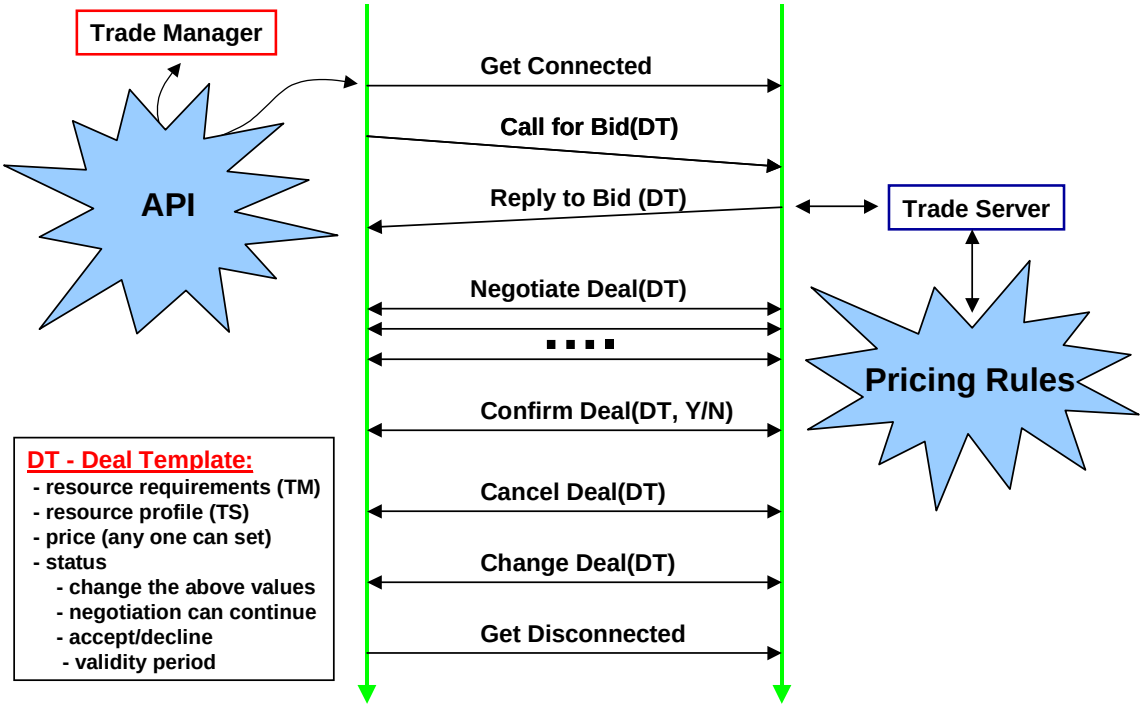

Figure 3.3: GRACE Open Trading Protocols.

A finite state machine representation of multilevel negotiation protocols that both client and server need to follow for the bargaining/tender model is shown in Figure 3.4. In this model, the broker's Trade Manager (TMs) contacts the resource owner's Trade Server (TS) with a request for a quote. The TM

specifies resource requirements in a Deal Template (DT), which can be represented by a simple structure with its fields corresponding to deal items or by a "Deal Template Specification Language", similar to the *ClassAds* mechanism employed by the Condor [54] system. The contents of DT include expected start time, usage duration, memory, and storage requirements along with its initial offer. The TM looks into DT and updates its contents and sends back to TS. This negotiation between TM and TS continues until one of them indicates that its offer is final. Following this, the other party decides whether to accept or reject the deal. If accepted, then both work as per the agreement mentioned in the deal. The overhead introduced by the multilevel point-to-point protocol can be reduced when resource access prices are announced through Grid information services (e.g., MDS) or market directory.

A number of interaction protocols for a business negotiation on the Internet have been presented in [78]. It highlights some commonalities in the structure of different price negotiation mechanisms such as fixed price sales, auctions, and brokerages. These business negotiation models and protocols are also applicable for our resource trading and we have explored such models and protocols in our resource management and scheduling system.

Grid Open Trading APIs

The GRACE infrastructure supports generic Application Programming Interfaces (APIs) that can be used by the Grid tools and application programmers to develop software supporting the computational economy. The following trading APIs are C-like functions (high level view of trading protocols) that GRACE clients/brokers can use to communicate with trading servers:

- `grid_trade_connect(resource_id, tid)`
- `grid_request_quote(tid, DT)`
- `grid_trade_negotiate (tid, DT)`
- `grid_trade_confirm(tid, DT)`
- `grid_trade_cancel(tid, DT)`
- `grid_trade_change( tid, DT)`
- `grid_trade_reconnect(tid, resource_id)`
- `grid_trade_disconnect(tid)`

where,

```
tid = Trade Identification code
DT = Deal Template
```

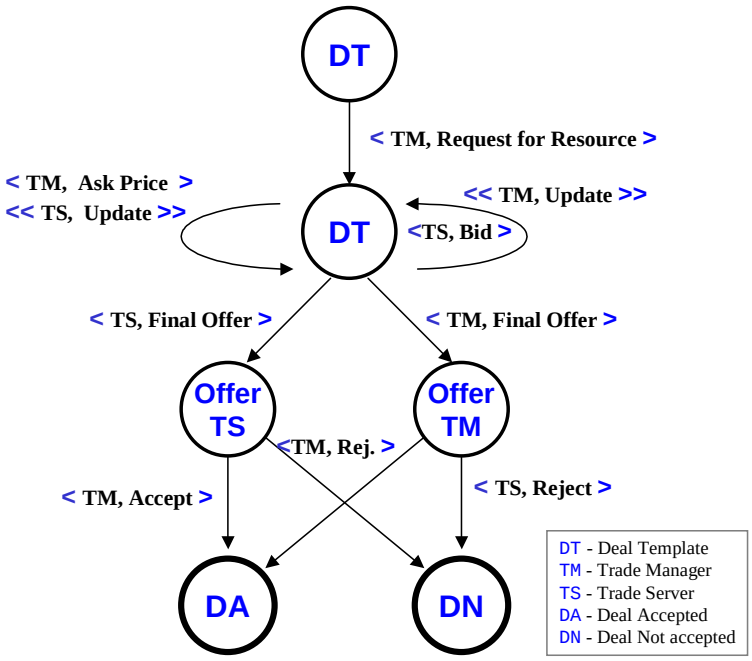

Figure 3.4: A finite state representation of resource trading (for bargain model).

3.4.4 Pricing, Accounting, and Payment Mechanisms

In a computational Grid economy environment, both resource owners and users want to maximize their benefits. As there will be many GSPs offering similar services, they need to have a competitive pricing structure in order to attract users, efficiently utilize resources, and maximize profit. The resources consumed by the user applications need to be accounted for and charged. Various payment mechanisms need to be supported. The users can purchase resource access credits in advance or pay-after-usage. Each GSP can maintain this by using systems like QBank or there can be global Grid-wide bank called GridBank (GBank) that mediates payment for services accessed by the user. Figure 3.5 shows various components at a GSP node and their interactions during resource trading, consumption, metering (measuring), billing, and payment handling.

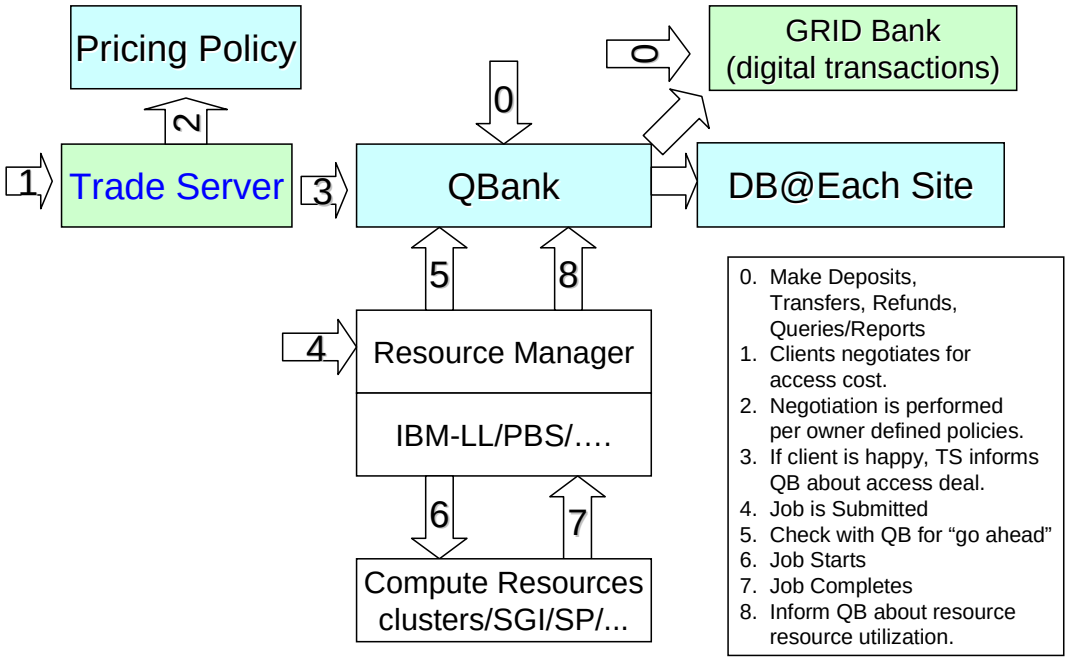

Figure 3.5: An Interaction between GSP resource management components.

How to Determine the Price?

A simple pricing scheme is a fixed price model, but this does not work when the users place QoS demands that vary with applications and time. In the context of software agents [36][60], many researchers have investigated pricing schemes based on the supply and demand for resources and the QoS requirements. The pricing schemes based on different parameters include,

- A flat price model (the same cost for applications and no QoS like in today's Internet [71])
- Competitive economic models (e.g., auctions and contract-net)
- Usage timing (peak, off-peak, lunch time like pricing telephone services)
- Usage period and duration (short/long)
- Demand and supply (e.g., Smale model [126])
- Foresight-based [36] (i.e., an ability to model and predict responses by competitors)
- Loyalty of Customers (like Airlines favoring frequent flyers!)
- Historical data
- Advance agreement/contract with service provides
- Calendar based
- Bulk Purchase
- Voting in which trade unions decide pricing structure
- Resource capability as benchmarked in the capital market
- Application areas in which academic R&D or public good applications can be offered at cheaper rate compared to commercial applications.

In [60], five different pricing strategies, ranging widely from ones that require perfect knowledge and unlimited computational power to ones that require very little knowledge or computational capability, are employed in two different buyer populations, namely quality-sensitive and price sensitive buyers. The resulting collective dynamics have been investigated using a combination of analysis and simulation. In a population of *quality-sensitive buyers*, all pricing strategies lead to a price equilibrium predicted by a game-theoretic analysis. However, in a population of *price-sensitive buyers*, most pricing strategies lead to large-amplitude cyclical price wars. These pricing strategies and issues are also applicable to the Grid and strategies need to be designed such that the resource providers benefit through efficient resource utilization and consumers will have the ability to trade-off between cost and timeframe in the Grid marketplace.

Service Items to be Charged and Accounted

User applications have different resource requirements depending on computations performed and algorithms used in solving problems. Some applications can be CPU intensive while others can be I/O intensive or a combination. For example, in CPU intensive applications it may be sufficient to charge only for CPU time whilst offering free I/O operations. This scheme cannot be applied for I/O intensive applications. Therefore, the consumption of the following resources needs to be accounted and charged:

- CPU - User time (consumed by user App.) and System time (consumed while serving user App.)
- Memory
- Maximum resident set size - page size
- Amount of memory used
- Page faults
- Storage used
- Network bandwidth consumption
- Signals received, context switches
- Software and Libraries accessed (particularly required for the emerging ASP world).

Access to each of these entities can be charged individually or in combination. Combined pricing schemes need to have a costing matrix that takes a request for multiple resources in pricing. An economic model proposed by Smale [126] allows formulation of such pricing schemes for resource allocation. G-commerce [118] is one such framework investigating the use and enhancing the Smale model for devising pricing strategies in the context of allocating resources for Grid users. By simulating hypothetical resource consumers and resource producers, they measured the efficiency of resource allocation under two different market conditions: commodities markets and auctions. By comparing the results of both market strategies in terms of price stability, market equilibrium, consumer efficiency, and producer efficiency, the G-commerce concludes that commodities markets are a better choice for controlling Grid resources than the existing auction strategies.

Payment Mechanisms

A computational economy Grid framework needs to support various payment mechanisms. They include:

- Prepaid – Pay and use in which users need to buy credits in advance from GSPs or Grid Bank
- Use and pay later
- Pay as you go
- Grants based

Each GSP can bill their users directly and handle all payment processing issues themselves. This method introduces a great burden for both providers and users in a large-scale Grid environment. This can be simplified by having mediators like a scalable Grid Bank. The user brokers can (automatically) inform the GSPs about the user Grid Bank account details for which they can charge directly or users can pay by other electronic cash systems. This can be achieved by using digital currency mechanisms such as:

- NetCheque: [13] – Users registered with NetCheque accounting servers can write electronic cheques and send them to service providers. When deposited, the balance is transferred from sender to receiver account automatically.
- NetCash [34] – This supports anonymity and it uses the NetCheque system to clear payments between currency servers.
- Paypal [89] – This is an example of a credit-card based automated mediator for payments processing.

- Tokens – This mechanism can be used when (a) resources are allocated based on grants; (b) resources are bartered and used by earning or exchanging credits like in the MojoNation peer networks.

Such electronic payment mechanisms satisfy the diverse requirements of service providers and their users. We believe that these payment mechanisms can be easily integrated into our Grid economy infrastructure.

3.5 Related Work

As in the conventional marketplace, the users' community (GRBs) represents the demand, whereas the resource owners' community (GSPs) represents the supply. Consumers interact with their own brokers (such as Nimrod-G) for managing and scheduling their computations on the Grid. The GSPs make their resources Grid enabled by running software systems (such as Globus [49] and Legion [121]) along with *Grid Trading Services* (GTS) to enable resource trading and execution of consumer requests directed through GRBs. The interaction between GRBs and GSPs during resource trading is mediated through a *Grid Market Directory* (GMD) (see Figure 3.6 to Figure 3.10). They use various economic models or interaction protocols for deciding service access price.

Numerous economic models including microeconomic and macroeconomic principles for resource management have been proposed in the literature [99][80][53][3][77][117]. Some of the commonly used economic models that can be employed for managing resources in Grid environment, include:

- Commodity Market Model
- Posted Price Model
- Bargaining Model
- Tendering/Contract-Net Model
- Auction Model
- Bid-based Proportional Resource Sharing Model
- Cooperative Bartering Model
- Monopoly and Oligopoly

Various criteria used for judging effectiveness of a market model are [131]: social welfare (global good of all), Pareto efficiency (global perspective), individual rationality (better off by participating in negotiation), stability (mechanisms that cannot be manipulated, i.e., behave in the desired manner), computational efficiency (protocols should not consume too much computation time), and distribution and communication efficiency (communication overhead to capture a desirable global solution).

Several research systems (see Table 3.1) have explored the use of different economic models for trading resources to manage resources in different application domains: CPU cycles, storage space, database query processing, and distributed computing, They include Spawn [14], Popcorn [87], Java Market [144], Enhanced MOSIX [145], JaWS [125], Xenoservers [23], D'Agents [55], Rexec/Anemone [8], Mojo Nation [84], Mariposa [83], Mungi[33], Stanford Peers [10], G-Commerce [118], OCEAN [76], Nimrod-G [100], and GridSim [95]. These systems have been targeted to manage single or multiple resources for application domains as follows:

- *Single domain computing systems:* Enhanced MOSIX and Rexec/Anemone.
- *Agent-based systems:* Xenoservers and D'Agents.
- *Distributed database management system:* Mariposa
- *Shared storage management system:* Mungi.
- *Storage Space Trading system:* Stanford Peers
- *Web-based distributed systems:* Popcorn, Java Market, and JaWS.
- *Multi-domain distributed Grid system:* Nimrod-G and GridSim Resource Broker

Among the above, the three most recent systems are Stanford Peers, G-Commerce, and GridSim Resource Broker and all of them happen to use simulation techniques to demonstrate distributed resource trading. However, each have a different focus: the Stanford Peers project is exploring storage trading for data replication; the G-Commerce project is exploring pricing strategies for selling access to resources; and the GridSim project provides a general purpose Grid simulation toolkit and the economic resource broker that allows exploration of quality of services driven algorithms for scheduling task and data parallel applications in large-scale distributed environments.

Table 3.1: Computational economy based distributed resource management systems.

SYSTEM NAME	ECONOMIC MODEL	PLATFORM	REMARKS
Mariposa [83] (UC Berkeley)	Bidding (Tendering/ ContractNet). Pricing based on load and historical info.	Distributed database.	It supports budget-based query processing and storage management.
Mungi [33] (University of New South Wales)	Commodity market (renting storage space that increases as available storage runs low, forcing users to release unneeded storage.)	Storage servers.	It supports storage objects based on bank accounts from which rent is collected for the storage occupied by objects. .
Popcorn [87] (Hebrew University)	Auction. (Highest bidder gets access to resource and it transfers credits from buyer to the seller account.)	Web browsers. (P*opcorn parallel code* runs within a browser of CPU cycles seller.)	Popcorn API-based parallel applications need to specify a budget for processing each of its modules.
Java Market [144] (Johns Hopkins University)	QoS based computational market. (The resource owner receives f(j, t) award for completing f in time t.)	Web browsers. (JavaMarket runs *standard Java Applets* within a browser).	One can sell CPU cycles by pointing Java-enabled browser to Portal & allow execution of Applets.
Enhanced MOSIX [145] (Hebrew U., Israel)	Commodity market (resource cost of each node is known)	Clusters of computers (Linux PCs)	It supports process migration such that overall cost of job execution is kept low.
JaWS [125] (University of Crete)	Bidding (Tendering)	Web browsers	It is similar to Popcorn.
Xenoservers [23] (University of Cambridge)	Bidding (Proportional resource sharing)	Single computer	Accounted execution of un-trusted code.
D'Agents [55] (Dartmouth College)	Bidding (Proportional resource sharing)	Single computer or Mobile Agents	Agents bid function is proportional to benefit.
Rexec/Anemone [8] (UC Berkeley)	Bidding/Auction (for proportional resource sharing)	Clusters (A market-based Cluster Batch Queue System)	Users assign utility value to their application and system allocates resources proportionally.
Mojo Nation [84] (Autonomous Zone Industries, CA)	A Credit-based partnership and/or bartering model. (Contributors earn credits by sharing storage and spend them when required)	Network storage.	It is a content-sharing community network. It combines marketplace and bartering approach for file/resource sharing.
Spawn [14] (Xerox PARC)	Second-price/Vickery auction (uses sponsorship model for funding money to each task depending on some requirements)	Network on workstations. Each WS executes a single task per time slice	It supports execution of concurrent program expressed in the form of hierarchy of processes that expand and shrink size depending on the resource cost.
CSAR Supercomputing center [56] (University of Manchester)	Commodity market and priority-based model (they charge for CPU, memory, storage, and human support services)	MPPs, Crays, and Clusters, and Storage servers.	Any application can use this service and QoS is proportional to user priority and scheduling mechanisms.

Nimrod-G [100] (Monash University)	It supports economy models such as commodity market, spot market, and contract-net for price establishment.	World Wide Grid (having resource Grid enabled using middleware systems like Globus)	It is a *real* system that supports deadline and budget constrained algorithms for scheduling task-farming and data parallel applications on world-wide distributed resources depending on their cost, power, availability and users quality of service requirements.
GridSim [95] (Monash University)	Currently, it supports economic models similar to those used in Nimrod-G, but limited to them.	A Java-based discrete event toolkit for simulating Grid resources, users, applications, and brokers.	The economic Grid resource broker supports deadline and budget based time, cost, cost-time, and conservative time optimisation scheduling algorithms.
G-Commerce [118] (U. of California Santa Barbara)	Commodity and auctions	Simulates hypothetical consumers and produces.	It is exploring strategies for pricing Grid resources to enable resource trading.
OCEAN [76] (U. of Florida)	Continuous double auction	A Java based platform with distributed PCs.	It is exploring the use of continuous double auction for trading computational resources – in development.
Stanford Peers [10] (Stanford University)	Auctions with cooperative bartering in a cooperative sharing environment.	Simulates storage trading for content replication and archiving.	It demonstrates distributed resource trading policies based on auctions by *simulation* – in development.

Each of the resource management systems presented in Table 3.1 follows a single model for resource trading. They have been designed with a specific goal in mind either for CPU or storage management. In order to use some of these systems, applications have to be designed using their proprietary programming models, which is generally discouraging, as applications need to be specifically developed for executing on those systems. Also, resource trading and job management modules have been developed using monolithic system architecture that limits their extensibility.

In the GRACE framework, we have separated these two concerns through a layered design approach to support different middleware technologies that co-exist with trading strategies and user-level resource brokers. The resource trading services are offered as core services and they can be used by different higher-level services/tools such as resource brokers and resource-aware applications. Another key advantage of Nimrod-G system is that it allows the execution of legacy applications on large wide-area distributed systems.

Typically, in a Grid marketplace, the resource owners, and users can use any one or more of these models or even combinations of them in meeting their objectives [98]. Both have their own expectations and strategies for being part of the Grid. The resource consumers adopt the strategy of solving their problems at low cost within a required timeframe. The resource providers adopt the strategy of obtaining best possible return on their investment while trying to maximize their resource utilization by offering a competitive service access cost in order to attract consumers. The resource consumers can choose providers that best meet their requirements. The design and architecture for the development of Grid systems using these economic models is discussed in Section 3.6.

Both GRBs and GSPs can initiate resource trading and participate in the interaction depending on their requirements and objectives. GRBs may invite bids from a number of GSPs and select those that offer the lowest service costs and meet their deadline and budget requirements. Alternatively, GSPs may invite bids in an auction and offer services to the highest bidder as long as its objectives are met. Both GSPs and GRBs

have their own utility functions that must be satisfied and maximized. The GRBs perform a cost-benefit analysis depending on the deadline (by which the results are required) and budget available (the amount of money the user is willing to invest for solving the problem). The resource owners decide their pricing based on various factors. They may charge different prices for different users for the same service or it can vary depending on the specific user demands. Resources may have different prices based on environmental influences such as the availability of larger core memory and better communication bandwidth with an outside world.

Grid brokers (note that in a Grid environment each user has his/her own broker as his agent) may have different goals (e.g., different deadlines and budgets), and each broker tries to maximize its own good without concern for the global good. This needs to be taken into consideration in building automated negotiation infrastructure. In a *cooperative distributed computing or problem-solving environment* (like cluster computers), the system designers impose an *interaction protocol* (possible actions to take at different points) and a *strategy* (a mapping from one state to another and a way to use the protocol). This model aims for global efficiency as nodes cooperate towards a common goal. On the other hand, in Grid systems, brokers and GSPs are provided with an interaction protocol, but they choose their own private strategy (like in multi-agent systems), which cannot be imposed from outside. Therefore, the negotiation protocols need to be designed assuming a *non-cooperative, strategic* perspective. In this case, the main concern is what social outcomes follow given a protocol, which *guarantees that each broker/GSP's desired local strategy is best for that broker/GSP and hence the broker/GSP will use it*.

3.6 Economic Models in the Context of GRACE Framework

In the previous section we identified a few popular models that are used in human economies. In this section we discuss the use of different economic models and propose architecture for realizing them. The discussion on realizing negotiation protocols based on different economic models is kept as generic as possible. This ensures that our proposed architecture is free from any specific implementation and provides a general framework for any other Grid middleware and tools developers. Particular emphasis will be placed on framework and heuristics that Grid resource brokers (*G-Brokers*) can employ for establishing a service price depending on their customers' requirements.

For each of the economic models, the economic model theory, its parameters and strategies are discussed and then a possible solution is given for a current Grid environment and how they can be mapped to existing Grid tools and architectures or what needs to be extended. In the classical economic theory there are different models for specific environmental situations and computing applications. Since the end-user interaction is the main interest of this chapter, we point out possible interactions with the broker.

3.6.1 Commodity Market (Flat or Supply-and-Demand Driven Pricing) Model

In the commodity market model, resource owners specify their service price and charge users according to the amount of resource they consume. The pricing policy can be derived from various parameters and can be *flat* or *variable* depending on the resource *supply and demand*. In general, services are priced in such a way that supply and demand equilibrium is maintained. In the *flat price model*, once pricing is fixed for a certain period, it remains the same irrespective of service quality. It is not significantly influenced by the demand, whereas in a *supply and demand* model, prices change very often based on supply and demand changes. In principle, when the demand increases or supply decreases, prices are increased until there exists equilibrium between supply and demand. Pricing schemes in a Commodity Market Model can be based on:

- Flat fee
- Usage Duration (Time)
- Subscription
- Demand and Supply-based [71]

The resource owners publish their prices and rules in the Grid market directory (GMD) service (see Figure 3.6) similar to publishing through yellow pages. This is accomplished by defining the price specification that GTS can use for publishing service access price in the market directory. A simple price specification may contain the following parameters.

- consumer_id, which is the as same Grid-ID
- peak_time_price (say, between 9am-6pm: office hours on working days)

- lunch_time_price
- offpeak_time_price
- discount_when_lightly_loaded (i.e., if the load is less than 50% at any time)
- raise_price_high_demand (i.e., raise in price if the average load is above 50%)
- price_holiday_time (i.e., during holidays and week ends)

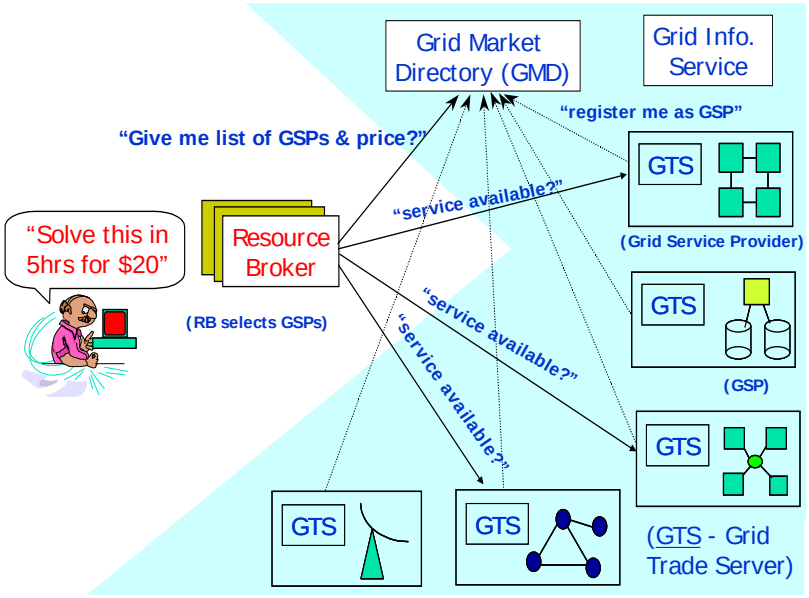

Figure 3.6: Interaction between GSPs and users in a commodity market Grid for resource trading.

Traditionally, computational services are priced based on their production cost and desired profit margin. However, the consumers' perception of value is based on parameters such as supply and demand for resources, priority and service quality requirements. Therefore, the resource value in Grid economy needs to be defined as a function of many parameters as follows:

- Resource Value = Function (Resource strength, Cost of physical resources, Service overhead, Demand, Value perceived by the user, Preferences);

The last three parameters are difficult to capture from consumers unless they see any benefit in disclosing them as they vary with time and application. However, there are consumers who prefer regular access to resources during a particular period of the day. For example, those involved in making regular decisions on supply chain management of goods shipping from inventory to the departmental stores prefer calendar-based guaranteed access, and stable but competitive pricing to resources unlike spot-market based access to services [69]. In this case demand and preferences are clear, pricing policy can be easily negotiated in advance in a competitive and reasonable manner and resource quality of services can be guaranteed through reservation during the required period as agreed in advance.

Consumers can be charged for access to various resources including CPU cycles, storage, software, and network. The users compose their application using higher-level Grid programming languages. For example, in our Nimrod problem-solving environment we provide a declarative programming language for composing parameter sweep applications and defining application and user requirements such as deadline and budget. The resource broker (working for the user) can carry out the following steps for executing applications:

a. The broker identifies service providers.
b. It identifies suitable resources and establishes their prices (by interacting with GMD and GTS).
c. It selects resources that meet its utility function and objectives. It uses heuristics and/or historical knowledge base while choosing resources and mapping jobs to them.
d. It uses resource services for job processing and issues payments as agreed.

As we are focusing on a generic framework, implementation specific details for releasing the above steps are not presented. For example, implementation specific details of our Nimrod-G resource broker vary from other related systems.

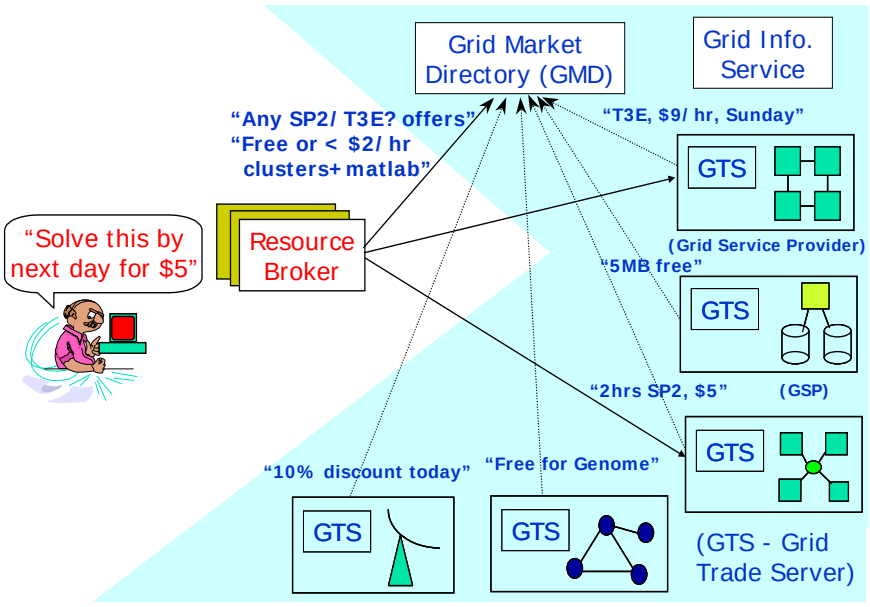

Figure 3.7: Posted price model and resource trading in a computational market environment.

3.6.2 Posted Price Model

The posted price model is similar to the commodity market model, except that it advertises special offers (see Figure 3.7) in order to attract (new) consumers to establish market share or motivate users to consider using cheaper slots. In this case, brokers need not negotiate directly with GSPs for price, but use posted prices as they are generally cheaper compared to regular prices. The posted-price offers will have usage conditions, but they might be attractive for some users. For example, during holiday periods, demand for resources is likely to be limited and GSPs can post tempting offers or prices aiming to attract users to increase resource utilization. The activities that are specifically related to the posted-price model in addition to those related to the commodity market model are:

a. Grid Service Providers (GSPs) post their special offers and associated conditions etc. in Grid Market Directory.
b. Broker looks at GMD to identify if any of these posted services are available and fits its requirements.
c. Broker enquires (GSP) for availability of posted services.
d. Other steps are similar to those pointed out in commodity market model.

3.6.3 Bargaining Model

In the previous models, the brokers pay access prices, which are fixed by GSPs. In the bargaining model, resource brokers bargain with GSPs for lower access price and higher usage duration. Both brokers and GSPs have their own objective functions and they negotiate with each other as long as their objectives are met. The brokers might start with a very low price and GSPs with a higher price. They both negotiate until they reach a mutually agreeable price (see Figure 3.8) or one of them is not willing to negotiate any further. This negotiation is guided by user requirements (e.g., deadline is too relaxed) and brokers can take risk and negotiate for cheaper prices as much as possible and can discard expensive machines. This might lead to lower utilization of resources, so GSPs might be willing to reduce the price instead of wasting resource cycles. Brokers and GSPs generally employ this model when market *supply-and-demand* and service prices are not clearly established. The users can negotiate for a lower price with promise of some kind favour or use of GSPs services in the future.

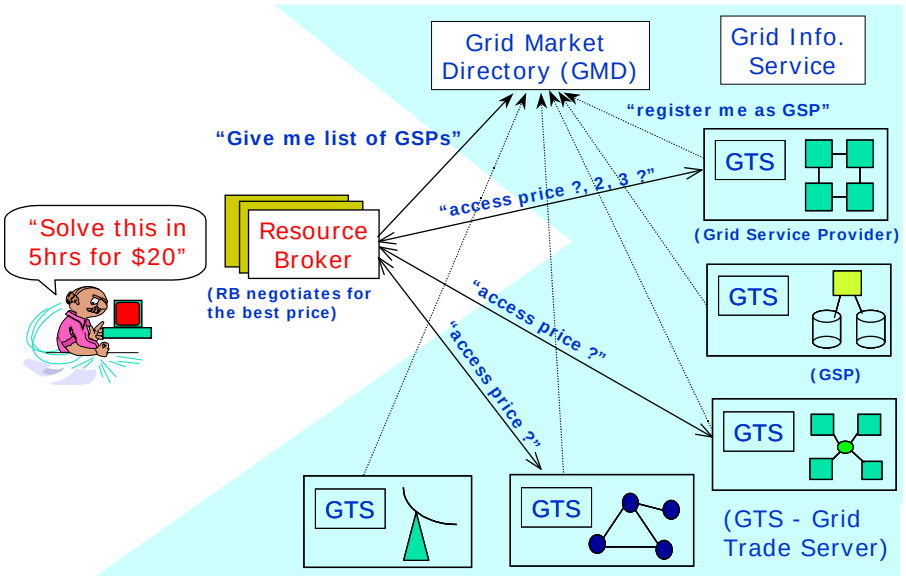

Figure 3.8: Brokers bargaining for lower access price for minimizing computational cost.

3.6.4 Tender/Contract-Net Model

Tender/Contract-Net model is one of the most widely used models for service negotiation in a distributed problem-solving environment [117]. It is modeled on the contracting mechanism used by businesses to govern the exchange of goods and services. It helps in finding an appropriate service provider to work on a given task. Figure 3.9 illustrates the interaction between brokers and GSPs in their bid to meet their objectives. A user/resource broker asking for a task to be solved is called the *manager* and a resource that might be able to solve the task is called potential *contractor*.

From a manager's perspective, the process is:

1. Consumer (Broker) announces its requirements (using deal template) and invites bids from GSPs.
2. Interested GSPs evaluate the announcement and respond by submitting their bids.
3. Broker evaluates and awards the contract to the most appropriate GSP(s).
4. The broker and GSP communicate privately and use the resource (R).

The contents of the deal template used for work announcement include, addressee (user), eligibility requirements specifications (for instance, Linux, x86arch, and 128MB memory), task/service abstraction, optional price that the user is willing to invest, bid specification (what should offer contain), expiration time (deadline for receiving bids).

From a contractor's/GSP perspective, the process is:

1. Receive tender announcements/advertisements (say in GMD).
2. Evaluate service capability.
3. Respond with bid.
4. Deliver service if bid is accepted.
5. Report results and bill the broker/user as per the usage and agreed bid.

The advantage of this model is that if the selected GSP is unable to deliver a satisfactory service, the brokers can seek services of other GSPs. This protocol has certain disadvantages. A task might be awarded to a less capable GSP if a more capable GSP is busy at award time. Another limitation is that the GRB manager has no obligation to inform potential contractors that an award has already been made. Sometimes, a manager may not receive bids for several reasons: (a) all potential GSPs are busy with other tasks, (b) a potential GSP is idle but ranks the proposed tender/task below the other tasks under consideration, (c) no GSPs, even if idle, are capable of offering service (e.g., resource is Windows NT-based, but user wants Linux). To handle such cases, a GRB can request quick response bids to which GSPs respond with messages such as *eligible*, *busy*, *ineligible* or *not interested*. This helps the GRB in making

changes to its work plan. For example, the user can change deadline or budget to wait for new GSPs or attract existing GSPs to submit bids.

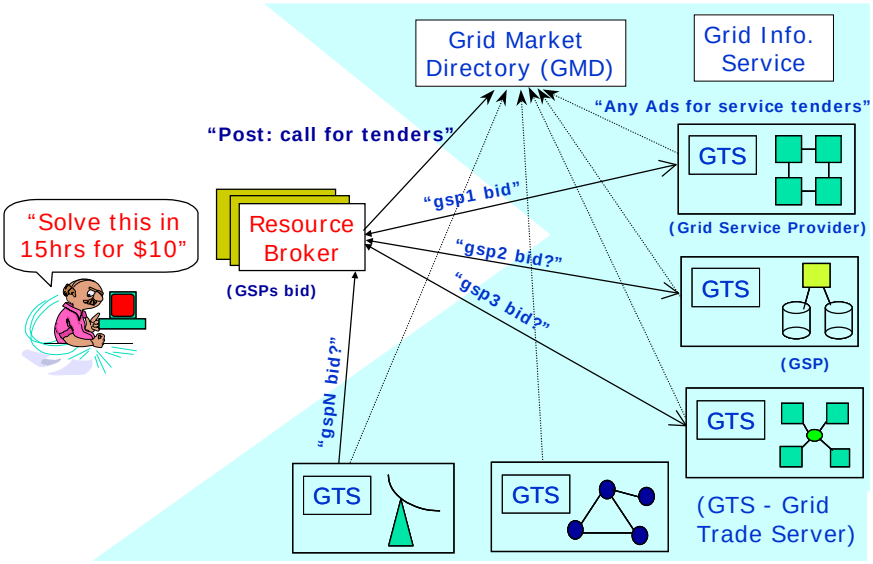

Figure 3.9: Tender/ContractNet model for resource trading.

The tender model allows directed contracts to be issued without negotiation. The selected GSP responds with an *acceptance* or *refusal* of award. This capability can simplify the protocol and improve the efficiency of certain services.

3.6.5 Auction Model

The auction model supports one-to-many negotiation, between a service provider (seller) and many consumers (buyers), and reduces negotiation to a single value (i.e., price). The auctioneer sets the rules of auction, acceptable for the consumers and the providers. Auctions basically use market forces to negotiate a clearing price for the service.

In the real world, auctions are used extensively, particularly for selling goods/items within a set duration. The three key players involved in auctions are: resource owners, auctioneers (mediators), and buyers (see Figure 3.10). Many e-commerce portals such as Amazon.com and eBay.com are serving as mediators (auctioneers). Both buyers' and sellers' roles can also be automated. In a Grid environment, providers can use an auction protocol for deciding service value/price (see Figure 3.11). The steps involved in the auction process are:

a. GSPs announce their services and invite bids.
b. Brokers offer their bids (and they can see what other consumers offer if they like - depending on open/closed).
c. Step (b) goes on until no one is willing to bid higher price or auctioneer stops if the minimum price line is not met.
d. GSP offers service to the one who wins.
e. Consumer uses the resource.

Auctions can be conducted as open or closed depending on whether they allow back-and-forth offers and counter offers. The consumer may update the bid and the provider may update the offered sale price. Depending on these parameters, auctions can be classified into four types:

- English Auction (first-price open cry)
- First-price sealed-bid auction
- Vickrey (Second-price sealed-bid) auction [142]
- Dutch Auction
- Double Auction (Continuous)

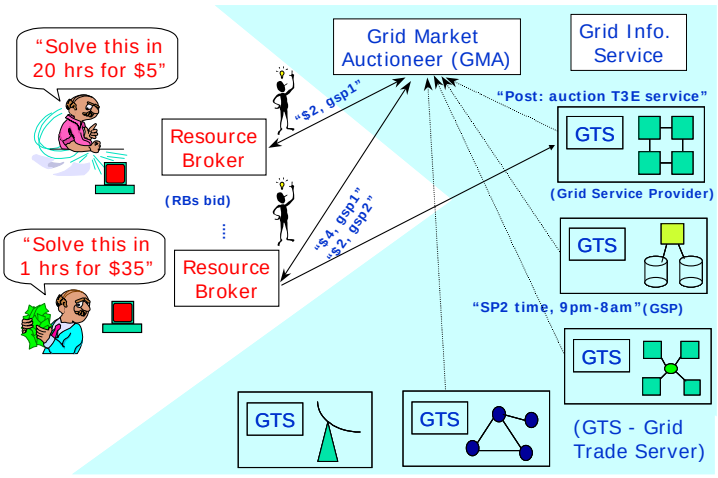

Figure 3.10: Auctions using external auctioneer.

English Auction (first-price open cry) — all bidders are free to increase their bids exceeding other offers. When none of the bidders are willing to raise the price anymore, the auction ends, and the highest bidder wins the item at the price of his bid. In this model, the key issue is how GRBs decide how much to bid. A GRB has a private value (as defined by the user) and can have a strategy for a series of bids as a function of its private value and prior estimation of other bidder's valuations, and the past bids of others. The GRB decides the private value depending on the user-defined requirements (mainly deadline and budget that he is willing to invest for solving the problem). In the case of private value English auctions, a GRB's dominant strategy is to always bid a small amount "higher" than the current highest bid, and stop when its private value price is reached. In correlated value auctions, the policies are different and allow the auctioneer to increase the price at a constant rate or at the rate he wishes. Those not interested in bidding anymore can openly declare so (open-exit) without re-entry possibility. This information helps other bidders and gives a chance to adjust their valuation.

First-price sealed-bid auction — each bidder submits one bid without knowing the others' bids. The highest bidder wins the item at the price of his bid. In this case a broker bid strategy is a function of the private value and the prior beliefs of other bidders' valuations. The best strategy is bid less than its true valuation and it might still win the bid, but it all depends on what the others bid.

Vickrey (Second-price sealed-bid) auction — each bidder submits one bid without knowing the others' bids. The highest bidder wins the item at the price of the second highest bidder [142]. The implementation architecture and strategies are similar to the ContractNet/Tender model discussed earlier.

Dutch Auction — the auctioneer starts with a high bid/price and continuously lowers the price until one of the bidders takes the item at the current price. It is similar to first-price sealed-bid auction because in both cases the bid matters only if it is the highest, and no relevant information is revealed during the auction process. From the broker's bidding strategic point of view, Dutch auction is similar to English (first-price sealed-bid auction). The key difference between them is that in an English auction bids start with low opening and increase progressively until demand falls whereas, in a Dutch auction bids start with high opening and decrease progressively until demand rises to match supply.

The interaction protocols for Dutch auction are as follows: the auction attempts to find market price for a good/service by starting at a price much higher than the expected market value, then progressively reducing the price until one of the buyers accepts the price. The rate of reduction in price is up to the auctioneer and they have a reserve price below which not to go. If the auction reduces the price to reserve price with no buyers, the auction terminates. In terms of real time, Dutch auction is much more efficient as the auctioneer can decrease the price at a strategic rate and first higher bidder wins. In an Internet wide auction, it is appealing in terms of automating the process wherein all parties can define their strategies for agents that can participate in multiple auctions to optimize their objective functions.

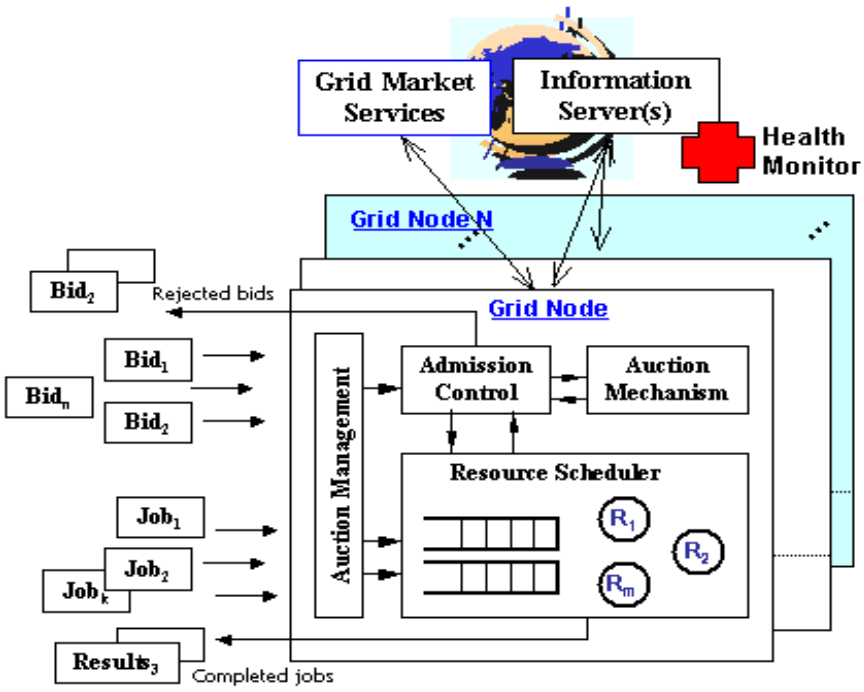

Figure 3.11: Auctions using their own Auctioneer.

Double Auction — This is one of the most common exchange institutions in the marketplace whose roots go back to ancient Egypt and Mesopotamia [66]. In fact, it is the primary economic model for trading of equities, commodities, and derivatives in stock markets (e.g., NASDAQ). In the double auction model, buy orders (*bids*) and sell orders (*asks*) may be submitted at anytime during the trading period. If at any time there are open bids and asks that match or are compatible in terms of price and requirements (e.g., quantity of goods or shares), a trade is executed immediately. In this auction orders are ranked highest to lowest to generate demand and supply profiles. From the profiles, the maximum quantity exchanged can be determined by matching *asks* (starting with lowest price and moving up) with demand *bids* (starting with highest price and moving down). Researchers have developed software-based agents mechanisms to automate a double auction for stock trading with or without human interaction [113].

The double auction model has high potential for Grid computing. The brokers can easily be enabled to issue *bids* depending on budget, deadline, job complexity, scheduling strategy, and resource characteristics requirements and GSPs can issue *asks* depending on current load and perceived demand, and price constraints. Both orders can be submitted to GMD agents that provide continuous clearance or matching services. Since bids are cleared continuously, both GRBs and GSPs can make instant decisions with less computational overhead and complexity.

All the above auctions differ in terms of whether they are performed as open or closed auctions and the offer price for the highest bidder. In open auctions, bidding agents can know the bid value of other agents and will have an opportunity to offer competitive bids. In closed auctions, the participants' bids are not disclosed to others. Auctions can suffer from collusion (if bidders coordinate their bid prices so that the bids stay artificially low), deceptive auctioneers in the case of a Vickrey auction (auctioneer may overstate the second highest bid to the highest bidder unless that bidder can vary it), deceptive bidders, counter speculation, etc.

3.6.6 Bid-based Proportional Resource Sharing Model

Market-based proportional resource sharing systems are quite popular in cooperative problem-solving environments like clusters (in single administrative domain). In this model, the percentage of resource share allocated to the user application is proportional to the bid value in comparison to other users' bids. The users are allocated credits or tokens, which they can use for having access to resources. The value of each credit depends on the resource demand and the value that other users place on the resource at the time

of usage. For example, consider two users wishing to access a resource with similar requirements, but the first user is willing to spend 2 tokens and the second user is willing to spend 4 tokens. In this case, the first user gets 1/3 of resource share whereas the second user gets 2/3 of resource share, which is proportional to the value that both users place on the resource for executing their applications.

This can be a good way of managing a large shared resource in an organization or resource owned by multiple individuals (like multiple departments in a university) who can have credit allocation depending on the investment they made. They can specify how much of a credit they are willing to offer for running their applications on the resource. For example, a user might specify low credits for non-interactive batch jobs and high credits for interactive jobs with high response times. GSPs can employ this model for offering a QoS for higher price paying customers in a shared resource environment (as shown in Figure 3.12). Systems such as Rexec/Anemone and Xenoservers, D'Agents CPU market employ proportional resource sharing model in managing resource allocations [98].

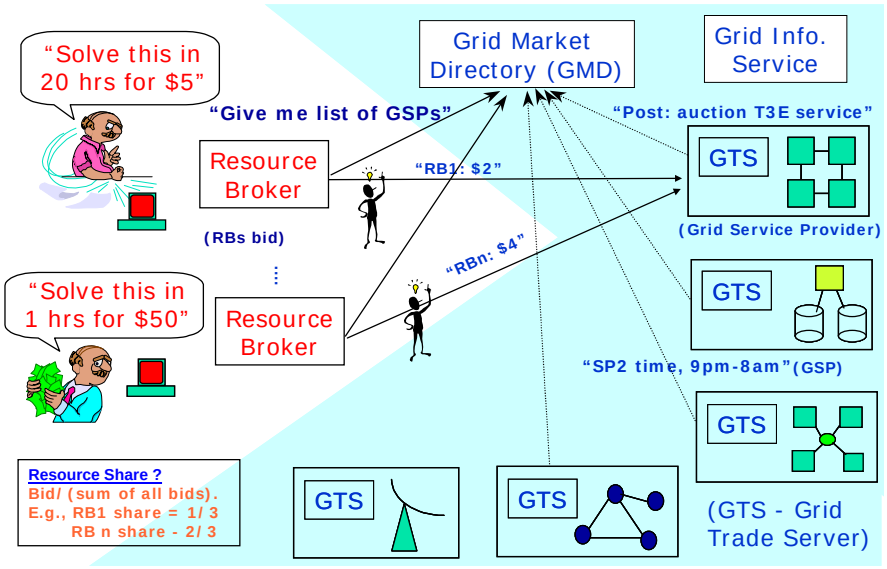

Figure 3.12: Market-based proportional resource sharing.

3.6.7 Cooperative Bartering Model

A community of individuals shares each other's resources to create a cooperative computing environment. Those who are contributing their resources to a common pool can get access to that pool. A sophisticated model can also be employed here for deciding how much resources share contributors can get. It can involve credits that one can earn by sharing a resource, which can then be used when needed. A system like Mojonation.net employs this model for storage sharing. This model works when those participating in the Grid act as both service providers and consumers.

3.6.8 Monopoly/Oligopoly

In the previously mentioned models we have assumed a competitive market where several GSPs and brokers/consumers determine the market price. However, there exist cases where a single GSP dominates the market and is the single provider of a particular service. In economic theory this model is known as a monopoly. Users cannot influence the prices of services and have to choose the service at the price given by the single GSP who monopolized the Grid marketplace. As regards the technical realization of this model, the single site puts the prices into the GMD or information services and brokers consult it without any possibility to negotiate prices.

The competitive markets are one extreme and monopolies are the other extreme. In most of the cases, the market situation is *oligopoly*, which is in between these two extreme cases: a small number of GSPs dominate the market and set the prices.

3.6.9 Other Influences on Market Prices

We now state more influences on price setting strategies in competitive, international markets. Supply and demand is the most common one but one also has to take into account national borders and different pricing policies within different countries such as taxation, consumer price index, inflation, etc. These factors are not dealt in this chapter, but implementations may need to consider them. There are micro and macro-economic factors that play an important role. One can also neglect them and build a price model on which all the Grid consumers have to agree. So this would correspond to an international market with special rules. Then, a model has to be formed for price changes. What is the factor for that change? Is there a monopoly that can decide what to do? Is the market transparent with optimally adapted prices? These are some of the main questions that need to be answered by GSPs when they decide their prices in an international market. A broker may consult the Grid Information Service to find out where the price for a particular service is minimal. For instance, one country might impose special taxes on a service whereas another country does not.

There are occasions where resources are not valued as per the actual cost of resources and overhead involved in offering services. When new players enter the market, in order to attract customers from the existing GSPs they are likely to offer access at minimal price by under valuing resources. This leads to price wars as GSPs are caught in a price cutting round to compete with each other. Measures such as intervention of *price regulation authorities* can be in place to prevent the market from collapsing or leaving it to the market to consolidate naturally.

3.7 Summary and Conclusion

We have discussed motivations for the use of computational economy as a metaphor for the management of resources and application scheduling in Grid computing environments. We proposed the Grid Architecture for Computational Economy (GRACE) framework and discussed an architecture that can be realized by leveraging existing middleware services. We have presented economic models such as commodity market, posted prices, bargaining, tendering, auction, proportional resource sharing or shareholder, and cooperative bartering models along with architecture and strategies for releasing them within the GRACE framework.

The computational economies driven brokering system can be applied to peer-to-peer computing [12] applications that enable content sharing. Systems like Napster [88] or Gnutella [39] could use infrastructure that is similar to GRACE for encouraging people to share files, contents, or music in larger scale by providing them economic incentive. The brokering systems like Nimrod-G can discover the best content provider that meets consumers QoS requirements. We believe this approach, of providing an economic incentive for resource owners to share their resources and resource users to trade-off between the deadline and budget, promotes the Grid as a platform for mainstream computing, which can lead to the emergence of a new service oriented computing industry.

In the next chapter, we present an implementation of grid resource broker, called Nimrod-G, driven by GRACE framework. The Nimrod-G resource broker supports deadline and budget requirements driven resource allocation and application scheduling on the World Wide Grid.