Node-level Architecture Design and Simulation of the MAGOG Grid Middleware

Jeremy Cohen[1] **Uli Harder**[2] **Fernando Martínez Ortuño**[2]
Colin Richardson[1] **John Darlington**[1]

[1] Internet Centre, Department of Computing,
Imperial College London, 180 Queen's Gate
London SW7 2AZ, UK
Email: {jhc02|clrich|jd}@doc.ic.ac.uk

[2] Department of Computing,
Imperial College London, 180 Queen's Gate
London SW7 2AZ, UK
Email: {uh|fermaror}@doc.ic.ac.uk

Abstract

The Middleware for Activating the Global Open Grid (MAGOG) provides a novel solution to the problem of discovering remote resources in a globally interconnected environment such as the Internet, in situations where users want to gain access to such resources to carry out remote computation. While existing Grid middleware enables the building of Grid infrastructures within closed environments where all users are known to each other, or where there is some pre-existing relationship between resource providers and users, the true Grid model should enable any users at any location to access remote resources without any prior relationship with the provider. MAGOG is a peer-to-peer based architecture that provides the means to enable discovery of resources in such an environment and to enable the agreement of pricing and Service Level Agreements (SLAs) for the use of these resources. This paper provides a high-level overview of the design of MAGOG and early simulation work that has been carried out to verify this design. It then focuses on the initial design for the middleware client that players in the market will need to deploy in order to become a node in the environment.

Keywords: Grid computing, middleware, peer-to-peer, resource discovery, negotiation, marketplace, Internet markets, Grid economics

1 Introduction

As computational problems in science and industry get larger, the computational power required to simulate, model and develop solutions to these problems also increases. In turn processors, storage capacity and network bandwidth have increased rapidly in power. There are now many types of computational task that can consume more power than is practical for any one entity to manage and host locally. At the other end of the scale, the rise of advanced networks and collaborative working has resulted in many individuals recognising the benefits of using remote resources to carry out some computational tasks so as to preserve their limited local capacity for other tasks. This is particularly true of those working in industries where there is variable or infrequent demand for access to HPC power, such as various forms of graphics or film rendering and mathematical modelling and analysis within industries such as finance. Computational Grids have continued to evolve and various middleware systems have been developed that are able to create Grids of resources that allow remote computation to be carried out. However, while such systems support remote computation for known individuals or those with whom prior agreements have been made and specific resource configuration has been carried out, they are not generally able to handle access by unknown users and lack an economic model to support the pricing of resources and the various additional services to provide agreement of SLAs and accounting for use of resources.

With the aim of developing a solution to these problems, a design for a middleware based on a peer-to-peer (P2P) model has been developed. The Middleware for Activating the Global Open Grid (MAGOG) enables the reliable discovery of resources attached to a global computational infrastructure and determination of pricing and Service Level Agreements (SLAs) for the use of these resources.

As conceived by Dr. Colin Richardson (Richardson 2007), MAGOG is based on three core concepts: The *Catallaxy* paradigm, the *Small Worlds* principle and *Double Message Flooding.*

Catallaxy: The term Catallaxy was first used by Austrian economist Friedrich von Hayek. In the case of this work, the Catallaxy paradigm guarantees that for each and every resource, an approximate supply = demand equilibrium will emerge at a roughly stable price (after some transient behaviour due to a short period of "learning by doing" – so-called "false trading" at a non-equilibrium price). The use of the Catallaxy paradigm in the study of computational infrastructures has also been attempted by the European CATNETS project (Eymann & et al. 2005) and the Mercato architecture (Broberg et al. 2007).

Small Worlds: The Small Worlds Principle / Six Degrees of Separation was originally observed by Stanley Milgram during his research at Harvard in the late 1960s (Milgram 1967). His experiments resulted in the observation that there are, on average, six degrees of separation between any two individuals on earth. The concept of a small worlds network applies to computational networks that approximate this model. The Watts and Strogatz model (Watts & Strogatz 1998) builds on this

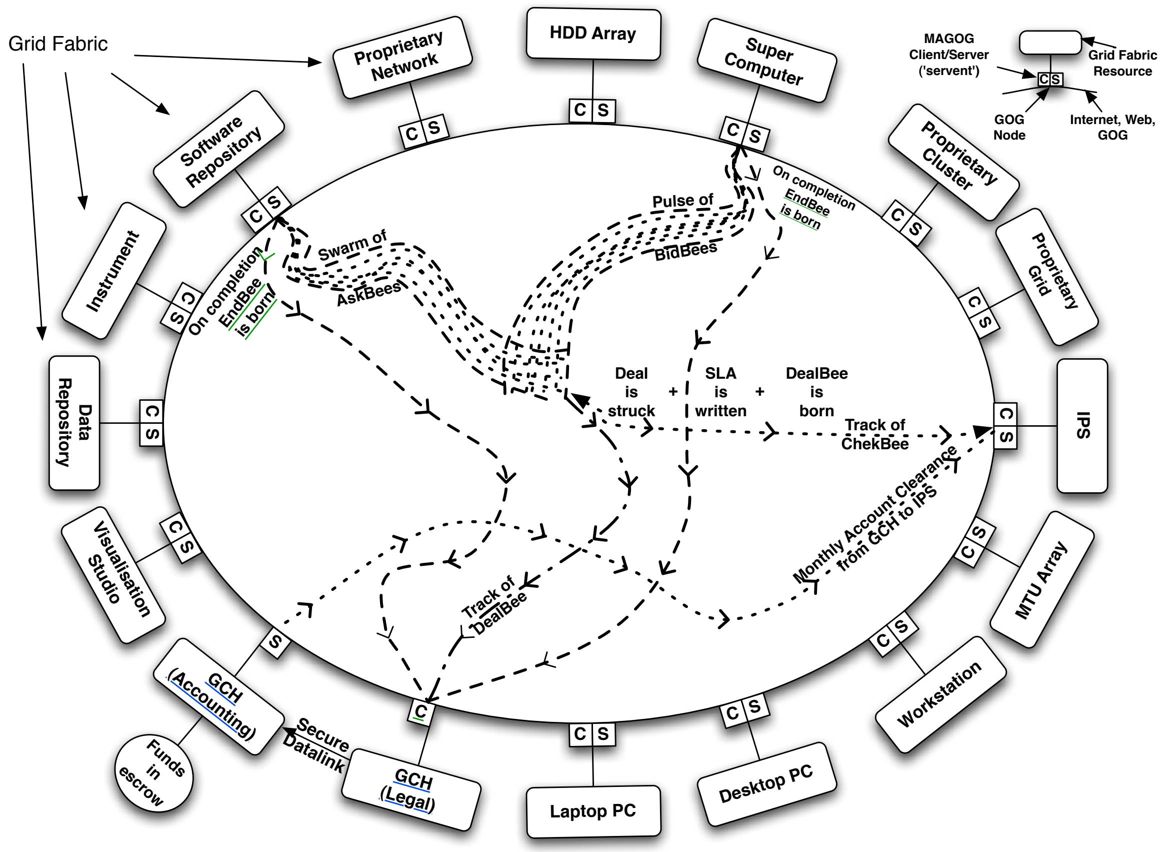


Figure 1: **The GOG network.**

principle to generate graphs that have 'small-world' properties. Applying this to MAGOG, with the underlying network approximating a small world network, each MAGOG node can potentially contact any other node in the network within an average of 6 hops given a list of 'useful' friends/neighbours. The optimal number of neighbours that each node needs to be aware of will be determined through simulation.

Double Message Flooding: Double Message Flooding is an extension of the traditional message flooding approach where large numbers of packets are sent out onto a network and propagated through as many links as possible in order to reach some destination, the location of which is not known. Message flooding is highly inefficient and can rapidly result in significant network congestion. Its use is therefore to be avoided. However, the problems with standard message flooding result from the number of hops that messages make. As the number of hops increases, the total number of copies of a message in the network increases vastly. The aim of double message flooding is to restrict the number of hops made by a message while maintaining the benefits of the message flooding concept. In double message flooding, messages are sent out by both those seeking to find another node and those seeking to be found. Pairs of messages originating from different nodes are evaluated as they meet at remote locations within the network. If a match is found between a pair of messages, the two nodes that originally generated the messages are put in direct contact. By combining this concept with a small world network, the number of hops that messages make before finding a suitable match can be significantly reduced when compared to standard message flooding, vastly reducing the chances of serious network congestion that results from traditional message flooding.

These three concepts provide the basis for MAGOG and have resulted in a novel architecture to support the discovery and pricing of resources in a complex computational Grid environment. The architecture does not prescribe a method for the usage of resources once they have been discovered and price and SLA have been agreed since this can be handled by existing Grid deployment services. The problem of acquisition of resources within a Grid environment is also addressed by the NOMAD system (Bubendorfer 2002) and work by Buyya et al. in the Grid Economy project (Buyya et al. 2000, 2005) where a variety of different market models have been investigated in (Buyya et al. 2002). The Nimrod-G resource broker (Abramson et al. 2002) supports the discovery and scheduling of resources onto computational grids based on an economic model.

This paper presents an overview of the MAGOG architecture and its key properties. The results of recent simulation work are then presented. The simulation work is being used to validate the architectural model and test that desired properties emerge. Development of the technical specification for the system is then covered. This involves the development of schemas for the various message types that are exchanged within the system. The paper finishes with conclusions and details of further work.

2 MAGOG Architecture

The architecture represents an open global network to which nodes can be connected representing either one or more Grid resources that have capacity available for sale, or a user that wants to purchase access to remote resources. The widespread adoption of such a system would enable an open market in computational resources and result in the commoditisation of these resources and their trading at the lowest possible sustainable price: long-run marginal cost.

This section provides an overview of both the high level Gobal Open Grid (GOG) network on which MAGOG nodes operate and the MAGOG stack that defines the structure and facilities each node must provide in order to interact with other nodes in the GOG network.

2.1 High-level System Design: The GOG Network

MAGOG operates on the GOG network shown in Figure 1. This is an open network that is a pure unstructured peer-to-peer (P2P) mesh overlay network on top of the existing Internet. Many common P2P networks such as Gnutella (*The Gnutella Protocol Specification v0.4* n.d.) and FastTrack (*The FastTrack Protocol* n.d.) follow an unstructured model. The network is shown as a ring in Figure 1 for simplicity but each node is in fact logically connected to a set of 'neighbours' that it communicates directly with. Every node connected to the network has a client/server interface that enables it to send messages to and receive messages from other nodes. The client/server interface presented to the network by each node is termed the MAGOG "servent". The network provides access to Grid fabric resources. These resources may include a vast array of different computational capabilities from very small low powered devices such as mobile phones, through individual desktop PCs, clusters and large HPC hardware to storage arrays, software and scientific instruments. A node may represent a single grid fabric resource, or may act as an interface to a local network consisting of a number of resources. For all resources represented by a node, metadata is stored describing those resources (see section 5.2). A reservation schedule is kept for all resources accessible via a MAGOG node in order to manage the available computational capacity. Nodes also have the ability to manage application workflows by requesting access to multiple grid fabric services in order compose a package of deals that can provide all the services necessary to execute a workflow.

2.2 The MAGOG Stack

The MAGOG stack is shown in Figure 2. The stack defines the five layers that form the internal structure of each MAGOG node and provide the functionality necessary to communicate with other nodes and make deals for sale or purchase of resource capacity.

The bottom two layers of the stack provide the network and security protocols. These layers provide the ability to securely transport messages across the GOG P2P mesh network. Messages are generated at higher levels of the stack and passed to level 2 where any necessary encryption or other security processes are handled. Messages are then passed on to level 1 for transmission onto the GOG network.

When describing the messages generated in levels 3 and 4 of the stack we use the metaphor of bees to represent the different packet types that are generated by a MAGOG node. This is due to the 'swarms' of packets generated by a node that are sent out onto the network in search of a compatible packet to pair with.

Two different types of GridBees are defined: BidBees and AskBees. These are generated to represent resource capacity that is required or offered by potential consumers or providers. A node that has one or more grid fabric resources to offer sends out pulses

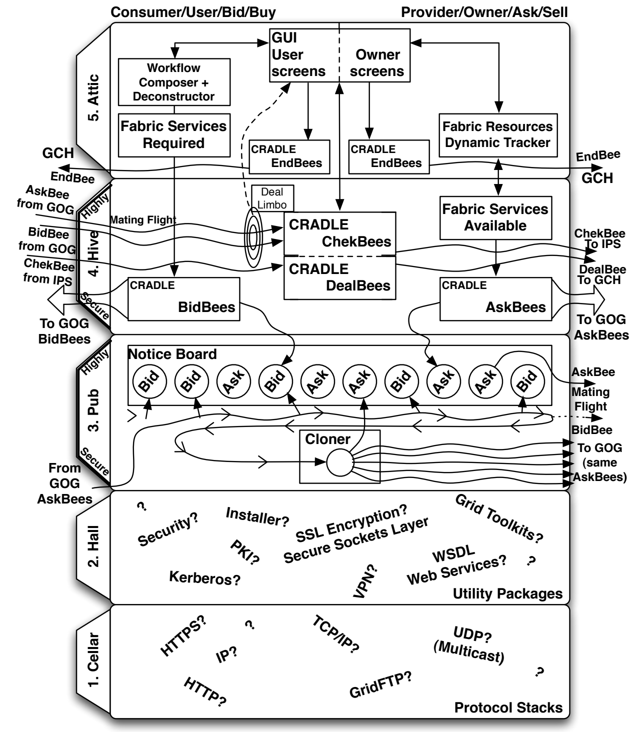


Figure 2: **The five layers of the MAGOG stack.**

of AskBees, at timed intervals, containing details of the resource that is being offered and the cost of that resource. These AskBees represent a Service Level Agreement (SLA) for use of the resource. In turn, at irregular intervals, swarms of BidBees are sent out by potential consumers containing details of the resources they require and the sum they're willing to pay. Each node has a list of a small set of neighbours and when GridBees are pulsed or swarmed onto the network, they are sent to all the node's neighbours. The approach of sending bids and asks directly to a small set of peers rather than undertaking a full auction of many goods among all market participants limits the potential for combinatorial explosion, an issue identified with the NOMAD system in (Bubendorfer et al. 2006). All GridBees have two significant properties – a Time To Live (TTL) and a Time Of Expiry (TOE). The TTL is a maximum hop count. This property is decremented every time a node handles a bee and passes it on. When its TTL reaches zero, that bee is not forwarded on. The TOE determines when a bee 'expires' and is no longer considered for matches against other GridBees.

When a node generates a BidBee or AskBee, the bee is pinned to the Notice Board of the local node and then transmitted to all the node's neighbours. These neighbours decrement the TTL of the bee, pin it to their own Notice Board and then forward it on to all their neighbours if the TTL is greater than zero. On the local Notice Board, the bee is compared against all other bees of the opposite type, that is, BidBees are compared against AskBees and vice versa. If a match is found, the pair of bees are cloned and sent to both the generator of the original AskBee and the generator of the BidBee. Both parties determine whether they are happy with the potential deal and, if so, a DealBee and ChekBee are created. The ChekBee flies to an Internet Payment Service (IPS) where the identity of the unknown party is verified and, in the case of the provider, the potential consumer's credit rating is checked to ensure that they have the ability to pay for the transaction.

If these checks are successful, the DealBee flies to a Grid Clearing House (GCH) to lodge details of the impending deal. The parties then interact to carry out the agreed deal and once completed, an EndBee is generated by both parties and sent to the GCH to denote successful completion of the deal.

At some point in the future, the GCH carries out an account clearance service with payment data being sent to various Internet Payment Services (IPS) to clear money between the accounts of various consumers and providers for the transactions that have been carried out. These transactions represent the trading of resources that has taken place between entities within the environment. It should be noted that there may be multiple GCHs and IPSs operating within the GOG network. All GCHs and IPSs are highly secure entities that need to ensure the non-repudiation of all communications and provide an audit trail of all communications and transactions in order to aid resolution of any disputes. Where possible, the tasks of identity verification and message content verification are assigned to these entities because their highly secure nature and relationship with their clients (the users of the GOG network) puts them in a good position to manage these tasks.

Clients may want to gain access to a set of resources to undertake a workflow of tasks. This may require a set of deals to be made to gain access to the necessary resources if a single provider is unable to offer all the necessary resources to satisfy the various tasks within the workflow. The resources obtained may be offered by multiple providers that do not have a pre-existing relationship. While MAGOG doesn't aim to provide the full deployment infrastructure for Grid jobs, there is the opportunity for the GOG network to configure a secure virtual private network layer that links such a group of independent resources at multiple sites so that they temporarily become part of a secure common network in order to collaborate to execute a workflow of tasks. This approach may be used to simplify the workflow execution process and take advantage of the network's security infrastructure to guarantee the identity and trustworthiness of a set of distributed sites. It is expected that, as in real-world economies, brokers will emerge. They will compete with providers of Grid and cloud computing services by reserving ensembles of popular grid fabric resources from a range of suppliers and offering secure 'virtual machines' capable of providing processing capacity to execute workflows submitted by users.

3 Simulation

The MAGOG system has been introduced in the preceding section. This section describes ongoing simulation work investigating certain aspects of the system. For simulation of the environment, a number of simplifying assumptions have to be made about the system and the node behaviour. The system has previously been analysed by simulation in (Harder & Martinez 2008).

To simplify the system for the simulation it is assumed that there is only a single type of resource and that nodes only bid for a single resource at a time and that nodes do not sell or buy multiple resources.

There is also a minimum price nodes can buy and sell resources for. To ensure that the price does not grow too large, which would cause computational problems, buying nodes have a budget that they use to pay for resources. This budget gets topped up to a maximum amount on a regular basis irrespective of the money left in their budget.

One of the main challenges for any simulation is to get some realistic behaviour into the system. In this case the challenge is greater since a real system implementation of MAGOG does not yet exist and there is no past data to analyse. To keep things as simple as possible nodes try to buy and sell single units of a resource. Once they have acquired or sold a resource, the two corresponding nodes become satisfied and are said to be in a deal. These nodes only start to try and buy and sell again when the deal comes to an end, which happens after a fixed time.

Prices for bids change at two possible points in the simulation. First they change after a deal has come to an end, buyers lower the price they use for their next bid and sellers increase theirs. Similarly, if a node finds itself without a partner with whom to strike a deal, it lowers its ask price in the case of a seller node and increases the price as a buyer node. In both cases the price P changes as

$$P' = P(1 \pm \Delta P).$$

The same ΔP is used for all nodes. Nodes send messages to their nearest neighbours and store messages in a buffer. The size of all buffers is the same for all nodes. If the buffer is full, messages are dropped. Messages in the simulation also have a time to live (TTL). Each time the message is passed between nodes the TTL is decreased and messages with a zero TTL are dropped. When a node has not managed to reach a deal it sends a new message with the new price.

The network for the nodes is chosen to be a Barabsi-Albert network (A.-L.Barabasi & R.Albert 1999). During the simulation the network structure is not changed. Barabasi-Albert networks are small world networks with a power law degree distribution. This results in a network where there are many nodes with a small degree (in/out going links) and a few with many. Barabasi-Albert networks are meant to resemble social human networks and are also found in P2P networks, for instance Freenet (Oram 2001). However, other network structures can also be found (Wang et al. 2006). There are also some models that allow to grow an arbitrary degree distribution for a network using local knowledge (Ghoshal & Newman 2007). The Barabasi-Albert model is one particular kind of preferential attachment network where the network is grown by adding new nodes by preferring attachment to a certain type of node, in this case well connected ones.

In the simulation, nodes are picked at random and their status is updated, i.e. messages are forwarded and possible deals are struck. For a network of size N we say that one Epoch has elapsed after N have been chosen. As this happens at random, nodes may be chosen no or multiple times during one epoch. This kind of stochastic simulation avoids artifacts produced by parallel or simultaneous updates (Huberman & Glance 1993) and also mimicks the fact that time is not necessarily synchronous in all nodes. On average, messages will survive TTL Epochs.

As the demand and supply in the simulation are fixed from the outset, looking at market clearing is not necessarily possible or is of limited value, we have first looked at the average deal price evolution over a long time for different network sizes. In any case the price stabilises after a sufficiently long time. In fact in this particular case the final price appears to be independent of the network size (see Figure 3).

The confidence intervals are not shown in the graph as they are small and would be hard to display. The overlay networks for the simulations were created using *Igraph* (Csàrdi & Nepusz 2006).

The simulation should also be able to tell whether the MAGOG architecture provides better service than

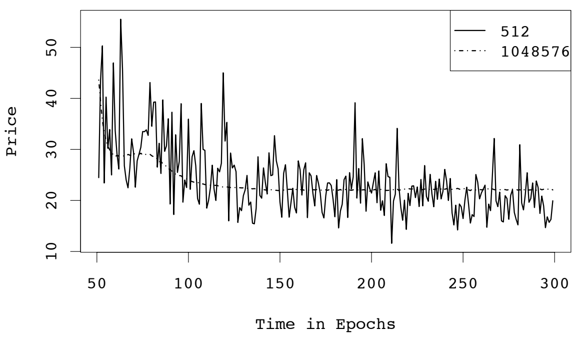


Figure 3: **The average price during the simulation for two different networks. The price appears to be independent of the network size, indicated in the legend.**

other possible systems, such as a central server. This could for instance be measured by looking at the time it takes nodes to sell or acquire resources. There are many other topics of possible future work for the simulation:

- Nodes buy/sell multiple resources
- Nodes right now are almost the simplest possible satisficing agents (Simon 1957). They need to be made more realistic by changing their demand/supply. Also they should have better or at least a larger number of strategies for price adjustment, similar to Brian Arthur's El Farol model (Arthur 1994).
- Nodes in the real system will be able to be consumers and suppliers. The simulation should reflect this.
- In a real system nodes will have failures and also nodes will join and leave the system. A simulation should be able to explore the stability of the system under these conditions.

4 Peer-to-Peer Network Infrastructure

The underlying peer-to-peer (P2P) infrastructure that MAGOG will make use of is still to be determined. The P2P framework will provide an overlay network that sits above the standard Internet, allowing resources that have recently joined the network to discover their neighbours and to rebuild their internal index of neighbours when one or more leave the network, or when "better neighbours" are identified.

It should be noted that neighbours are considered to be logical neighbours that may be located anywhere in the network rather than neighbours that are physically local in geographic terms. Each node maintains a small list of neighbours to which bid and ask messages are broadcast in order for them to be propagated through the network. These neighbours then, in turn, broadcast the messages on to their neighbours. As explained in section 2, a message has a Time To Live (TTL), a maximum hop count that is decremented every time a node passes the message on to one of its peers. At each node that a message reaches, it is stored in a resolution pool – the implementation of the "Notice Board" that is present at level 3 of the stack – where groups of Bid and Ask messages go through a matchmaking process to identify deals. Messages remain in this pool until their Time Of Expiry (TOE) is reached. When the TTL of a message reaches zero, a node does not pass it on to neighbours but still places it within its resolution pool and holds it there until it reaches its TOE.

Since a node has only a small number of neighbours, if some of those neighbours continually send GridBees that very rarely result in a deal, this limits a node's opportunities to make deals. To resolve this problem, a node may re-evalute its neighbours and in some cases replace them with better neighbours from which there is more chance of receiving GridBees that result in a deal, thus allowing a form of dynamic network optimisation. How and when these optimisations are made is a property that can be tuned at either the node or overall network level and this allows significant scope for modifying the network behaviour. Testing the effect that modification of these properties has on overall network performance is something that is planned for evaluation in ongoing simulation work.

Nodes joining the network 'bootstrap' from one of a set of seed nodes that give them an initial leaf-set of neighbours to communicate with. Nodes then learn of the existence of other nodes by looking at the origins of the various bid and ask messages that pass through their local notice board. As identified in (Tsoumakos & Roussopoulos 2006) which provides an analysis of various search methods for unstructured P2P networks, keeping direct pointers to peers is much more efficient than blind forwarding. However this relies on a relatively static network. In this instance, the network structure has the potential to change regularly so the method used to manage neighbour lists and maintain the small world properties of the network is significant in maintaining efficiency. When nodes are spotted that are very successful in making deals, a given node may want to update its local list of neighbours to include such nodes providing greater opportunities for receiving packets that result in deals. If a node leaves the network then its neighbours will remove it from their neighbour lists and add a new neighbour. When new nodes are added to a neighbour list, others must be removed so as not to exceed the maximum number of neighbours and less useful nodes that do not regularly assist in creation of deals are therefore removed. Investigation into the peer-to-peer infrastructure is ongoing to ensure that the small world and double message-flooding capabilities are maintained but also to be able to reliably bootstrap nodes and keep neighbour lists up to date. Simulation and testing is to be carried out to identify how the unstructured design of the MAGOG P2P network compares with existing Distributed Hash Table (DHT) based frameworks, such as Chord (Stoica et al. 2001), Tapestry (Zhao et al. 2004), Pastry (Rowstron & Druschel 2001) and CAN (Ratnasamy et al. 2001), for the generation of neighbour lists when nodes join the network. While DHT-based P2P frameworks avoid the problems of earlier P2P frameworks that still had some form of central directory or were inefficient due to broadcasting of large quantities of messages, the MAGOG network avoids the need for a DHT approach in building its neighbour lists. DHT-based systems provide a standard table of name/value pairs and distribute the pairs among all nodes in such a way that any node can efficiently retrieve the value associated with a key. When nodes leave the network and new nodes join, lookup failures are minimised due to the way that the pairs are distributed between nodes in the network. MAGOG's approach of dynamically optimising its neighbour list by watching transiting packets is considered to offer greater efficiency than than a DHT.

5 The MAGOG Daemon

Every node in the MAGOG environment needs to be able to communicate with other nodes in order to carry out the necessary communications that enable resource discovery and price agreement. This communication initially involves the broadcasting of packets – the BidBees and AskBees used to request or offer resource capacity – and then direct interaction with individual nodes and a Grid Clearing House (GCH) and Internet Payment Service (IPS) in order to finalise a deal and the associated Service Level Agreement (SLA). The stack defines the various levels of the architecture for which an implementation must be present on each node. This section of the paper presents the early design work that has been undertaken to develop a technical specification for the MAGOG daemon, an implementation of the stack that runs on each node involved in the GOG network.

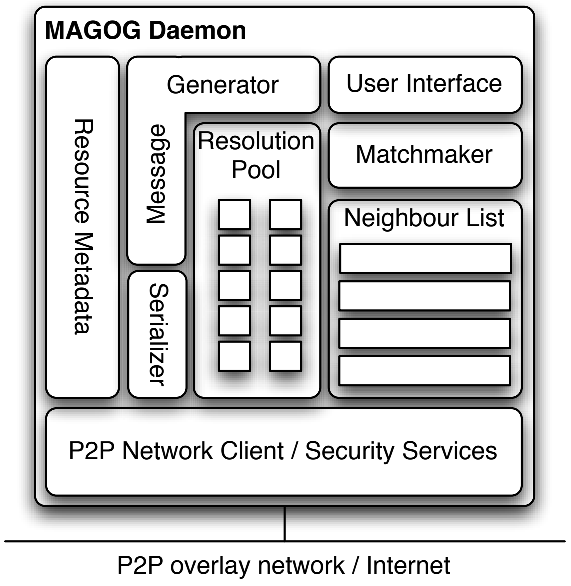


Figure 4: **The internal services and functions of the MAGOG daemon.**

The daemon is a piece of software that runs as a background process on every resource attached to the GOG network. The daemon is started during the startup of a resource and provides a server that listens on a network port for incoming messages and a client for sending messages. Figure 4 shows the internal services and functions of the daemon.

When a new user wishes to join the network, either to gain access to remote resource capacity, or to offer local capacity to the network, they download an implementation of the MAGOG client distribution. Resources that the user installs this distribution on become available as clients for discovering remote resources and agreeing usage terms for the remote execution of software on these resources. They are also enabled for discovery by remote nodes to allow local execution of software on behalf of remote users. The user must have an account with a GCH and an IPS. Configuration of the software requires the user to enter their account details for their GCH and IPS accounts and their identity is then checked with these entities before they are allowed to begin interacting with other nodes in the network. The GCH and IPS require several pieces of user information when a user signs up and these entities are then used to identify users in subsequent interactions with the GOG network. The following subsections look at the design of the various elements within the client.

5.1 Neighbour List

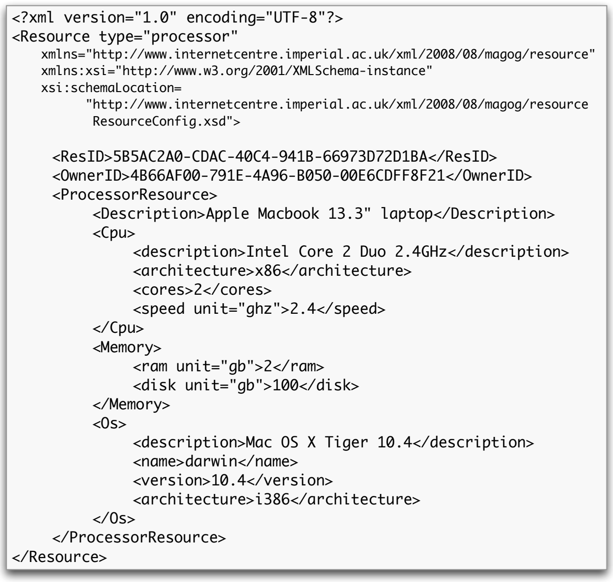
When a client joins the network for the first time, it carries out a discovery process by contacting seed nodes to find other available nodes and populate its neighbour list. This list of neighbours is stored in memory on the client and is also written to persistent storage on disk. Every time the node handles incoming messages that need to be forwarded, the messages are sent to every node in the neighbour list. The optimal size for the neighbour list will be determined through testing of the platform. The neighbour list undergoes dynamic optimisation by replacing neighbours that do not regularly provide deals despite significant traffic, ensuring that all a node's neighbours should provide a reasonable opportunity for a deal to be made. New potential neighbours are discovered by looking at the packets that pass through the local resolution pool. These packets may potentially originate from *any* resource on the GOG network.

When a node leaves the network and rejoins, the most recent neighbours can be obtained from the persistent storage on disk and these become the initial neighbour set. Any neighbours that have left the network since the list was stored to disk are removed and replaced with new neighbours. The optimisation process then ensures that this set of active neighbours is optimised over a period of operation.

5.2 Resource Metadata

Every Grid fabric resource that is connected to the MAGOG network has a description that contains a variety of resource specification information. The document begins with a type attribute. This defines the type of the resource and hence the content of the metadata that will be provided. Figure 5 shows an example XML document defining a processing resource attached to the network. A node holds metadata for all fabric resources that it presents to the network. The information contained in a metadata document includes details such as a unique ID to identify the resource and the unique ID of its owner, a human readable description of the resource, the CPU type and speed, memory size and storage capacity. Such a document may be generated automatically based on information entered by a user at the time the client is installed onto a resource. Once available, this information can be used to assist in the generation of ask messages when a user wishes to sell time on one of their resources since the AskBees pulsed onto the network will need to contain this information. Other resource types include clusters, software resources, various types of network attached storage, and scientific instruments.

Resource metadata documents conform to an XML Schema that specifies the structure and format of the content that may appear within a resource description document. The initial `type` attribute specified with the `Resource` element defines the resource type that dictates the child element that will contain the metadata. In the case of the example shown, the resource type is `processor` and the metadata for this type is contained within a child element of type `ProcessorResource`. The schema contains definitions for the accepted type attributes and the various available child element types that will be used to define metadata for a resource. This resource metadata is used within other packet types to define a resource specification (see sections 5.3.1 and 5.3.2). The use of a self-describing data interchange format such as XML simplifies interchange of information between nodes and there are a number of software libraries available for handling data in this format. For example when working in Java, the Java API for XML



```
<?xml version="1.0" encoding="UTF-8"?>
<Resource type="processor"
    xmlns="http://www.internetcentre.imperial.ac.uk/xml/2008/08/magog/resource"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation=
         "http://www.internetcentre.imperial.ac.uk/xml/2008/08/magog/resource
          ResourceConfig.xsd">

     <ResID>5B5AC2A0-CDAC-40C4-941B-66973D72D1BA</ResID>
     <OwnerID>4B66AF00-791E-4A96-B050-00E6CDFF8F21</OwnerID>
     <ProcessorResource>
          <Description>Apple Macbook 13.3" laptop</Description>
          <Cpu>
               <description>Intel Core 2 Duo 2.4GHz</description>
               <architecture>x86</architecture>
               <cores>2</cores>
               <speed unit="ghz">2.4</speed>
          </Cpu>
          <Memory>
               <ram unit="gb">2</ram>
               <disk unit="gb">100</disk>
          </Memory>
          <Os>
               <description>Mac OS X Tiger 10.4</description>
               <name>darwin</name>
               <version>10.4</version>
               <architecture>i386</architecture>
          </Os>
     </ProcessorResource>
</Resource>
```

Figure 5: **An example XML resource description document based on an XML Schema.**

Binding (JAXB) can be used to parse the incoming documents and convert them to object trees for easy programmatic access to the document content. Similarly, object trees can be generated and then serialised out to XML data for transmission across the network.

5.3 MAGOG Packets

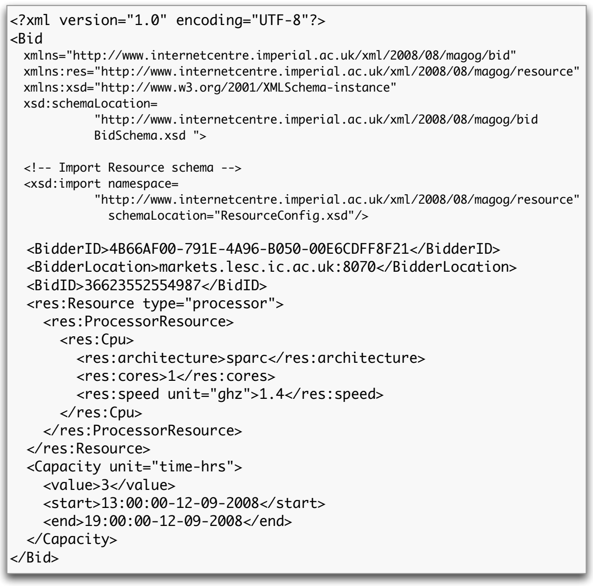
The network architecture specifies a set of different packets that the system uses for communication. These are represented by the various types of Bees used in the architecture description: BidBees, AskBees, DealBees, ChekBees and EndBees. A separate schema defines the content of each of these messages. They are currently defined in XML Schema but using serialised XML for packets is recognised to result in a relatively verbose form of communication and in order to minimise the network load, alternative approaches for messaging will be investigated as the work progresses. There is potential for a form of binary XML serialisation to provide a suitable reduction in message size but alternatives will also be investigated. This subsection of the paper now looks at each of the message types and their content. The schemas are versioned to support modifications as the system undergoes continued development. The descriptions provided here represent the initial versions of each schema.

5.3.1 Bid Packets - BidBees

Bid bees are sent out from a node in 'swarms' – large numbers of packets that are sent out onto the network at relatively high frequency – requesting access to a particular type of resource with a given specification for a particular sum of money. A bid packet schema defines the content of these messages. A bid needs to provide the ID of the bidder, details of the resource type and specification they are aiming to access, information on the amount they are willing to pay and details of any time constraints of when they wish to obtain access to the resource. A bid also contains a locally unique bid ID, that is, an ID that is unique to the specific bidder and when combined with the bidder ID represents a globally unique identifier for the bid. The resource details take the form of a sparse version of the resource metadata document that each resource must provide. For example, an individual may wish to purchase 1 hour of time on a resource that has an x86 processor with a minimum of 2Gb RAM. Any other resource properties may be irrelevant to the user and are not therefore provided. The bid packet schema therefore links to the resource metadata schema and imports the necessary types to save duplicating schema content and ensuring that the two schemas are linked and can more easily handle schema version changes.

In the case that the packet results in a match with an ask packet, a location must be provided where notification of the deal can be sent. This is the network address and port on which the MAGOG client for the given node is running and this information is also included within the packet.

Figure 6 shows an example of an XML bid packet for an entity wishing to gain 3 hours of time on a processing resource containing a 1.4GHz Sparc processor. The processing time can be provided anywhere between 13:00 and 19:00 on 12th September 2008.



```
<?xml version="1.0" encoding="UTF-8"?>
<Bid
  xmlns="http://www.internetcentre.imperial.ac.uk/xml/2008/08/magog/bid"
  xmlns:res="http://www.internetcentre.imperial.ac.uk/xml/2008/08/magog/resource"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema-instance"
  xsd:schemaLocation=
           "http://www.internetcentre.imperial.ac.uk/xml/2008/08/magog/bid
           BidSchema.xsd ">

  <!-- Import Resource schema -->
  <xsd:import namespace=
           "http://www.internetcentre.imperial.ac.uk/xml/2008/08/magog/resource"
             schemaLocation="ResourceConfig.xsd"/>

  <BidderID>4B66AF00-791E-4A96-B050-00E6CDFF8F21</BidderID>
  <BidderLocation>markets.lesc.ic.ac.uk:8070</BidderLocation>
  <BidID>36623552554987</BidID>
  <res:Resource type="processor">
    <res:ProcessorResource>
      <res:Cpu>
        <res:architecture>sparc</res:architecture>
        <res:cores>1</res:cores>
        <res:speed unit="ghz">1.4</res:speed>
      </res:Cpu>
    </res:ProcessorResource>
  </res:Resource>
  <Capacity unit="time-hrs">
    <value>3</value>
    <start>13:00:00-12-09-2008</start>
    <end>19:00:00-12-09-2008</end>
  </Capacity>
</Bid>
```

Figure 6: **An example XML bid packet.**

5.3.2 Ask Packets - AskBees

Ask bees are sent out in pulses rather than the much larger numbers or swarms of bid bees. A resource provider that wishes to offer some time on their resource will generate an ask packet and then send out a small pulse of these packets that are propagated through the network and attached to the notice boards of several nodes in order to await a match. The content of an ask bee provides the unique ID of the entity making the offer, a locally unique ask ID, details of the resource being offered which must be defined by a fully completed resource metadata document, again the resource metadata schema is linked to allow validation of the ask packet against the standard metadata definition, and details of cost and time constraints on the offer. As with a bid identifier, an ask identifier is unique to the entity providing the ask and when combined with the offering entity's unique identifier provides a globally unique identifier for the ask.

As with bid packets, in the case where an ask packet results in a match with a bid packet, a location must be provided where notification of the deal can be sent. The network address and port of the client on the node transmitting the ask packets is therefore also included within the packet.

5.3.3 Match Packets

Match packets are generated when a pair of bid and ask packets are identified as compatible by the matchmaker. This packet represents the 'mating flight' as described in the stack shown in figure 2 and is effectively a wrapper around the bid and ask packets that have resulted in a match. The packet header contains the identifiers of the three parties involved in the creation of this potential deal: the first two being those making the bid and the ask and the third being the owner of the node where the deal was made. The locally unique bid and ask identifiers are also added to the deal notification packet. The combination of these 5 different identifier values represents a unique deal identifier. The network address of the MAGOG daemon on the node where the match was made must also be included in the packet header. The match packet then contains the XML content of the bid and ask packets that have been matched. From these packets, the location of the nodes that generated the original bid and ask packets is extracted and a copy of the match packet is sent to each of them, a copy is also stored locally at the node where the match was made. The nodes that receive the match packet must then decide whether to go ahead with the match as a deal. This may be determined through manual user intervention or by an automated agent acting on a user's behalf.

5.3.4 Deal Notification Packets - DealBees

DealBees or deal notification packets are generated when a match packet is accepted by a node that wishes to go ahead and strike a deal. If a match is accepted, it is wrapped in a deal notification packet and sent to a Grid Clearing House (GCH). This wrapping of the match packet is necessary for security purposes and provides a container in which the sender can attach a hash of the match packet for verification by the GCH. Since a potential consumer or provider may receive multiple matches to bid or ask packets and will be likely to be able to accept only one or a small number of these matches depending on the capacity requested or offered, both parties in a deal need to know rapidly that the other party has also accepted the deal. When a GCH receives a deal notification packet it verifies the identity of the sender and if genuine, holds the packet in a buffer for a short period of time awaiting the matching deal notification packet from the other party in the deal. If both packets are received within the time window the deal can proceed and an acceptance of the deal is sent to both parties by the GCH. If only one deal notification is received by the GCH and the time window expires, the GCH sends a deal void message to both parties in the potential deal.

5.3.5 Payment Packets - ChekBees

A ChekBee or payment packet is sent to an Internet Payment Service to notify that a deal has been made and to register the cost of that deal and identify the other party involved. A ChekBee contains the unique deal identifier and payment information for the deal. When a deal is to be made, each party needs to verify the identity and credit rating of the other. The payment element is more important in the case of the provider who wishes to ensure that the consumer has the ability to pay for the resource they have made a deal for. However, the IPS also uses these packets to keep track of an entity's credit and knows to expect a credit or debit for the agreed amount at the next clearance of funds carried out by the GCH. The IPS verifies the identity of the party submitting the ChekBee and then checks the other party in the deal. The IPS then produces a response packet stating whether the identity and credit checks have been successful. This response must be ok before use of the requested or offered resources can begin. Figure 7 shows an example of an XML Chek packet.

```
<?xml version="1.0" encoding="UTF-8"?>
<Chek
  xmlns="http://www.internetcentre.imperial.ac.uk/xml/2008/08/magog/chek"
  xmlns:deal="http://www.internetcentre.imperial.ac.uk/xml/2008/08/magog/deal"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation=
       "http://www.internetcentre.imperial.ac.uk/xml/2008/08/magog/chek
       ChekSchema.xsd">

  <!-- Import Resource schema -->
  <xsd:import
      namespace="http://www.internetcentre.imperial.ac.uk/xml/2008/08/magog/deal"
      schemaLocation="DealSchema.xsd"/>

  <deal:DealID>
    <deal:BidderID>
        4B66AF00-791E-4A96-B050-00E6CDFF8F21
    </deal:BidderID>
    <deal:ProviderID>
        E8AEA05B-6D41-491D-8E26-2B2A301677EB
    </deal:ProviderID>
    <deal:MatchID>
        1BE258C9-F78C-498E-A0C9-0B63D79FFC7D
    </deal:MatchID>
    <deal:BidID>36623552554987</deal:BidID>
    <deal:AskID>21476267234233</deal:AskID>
  </deal:DealID>
  <PaymentDetails>
    <PurchaserAccount ipsid="F546BD-AA5546">
        10007654
    </PurchaserAccount>
    <SellerAccount ipsid="F546BD-AA5546">10001992</SellerAccount>
    <TransactionCost currency="GBP">2.76</TransactionCost>
  </PaymentDetails>
</Chek>
```

Figure 7: **An example XML chek packet.**

5.3.6 Transaction Complete Packet - EndBees

When a deal has concluded and use of the contracted resources has been completed, an EndBee or transaction complete packet is sent by both entities involved in the deal to the GCH. This packet contains the unique deal identifier and a value that specifies whether the deal was completed successfully or not. If a problem occurred, the audit trail of all communications leading to the deal is accessible and retained pending requests from one or other party involved in the deal to resolve the issue.

5.4 Resolution Pool and Matchmaker

The resolution pool represents the notice board in layer three of the stack. This is a pool of bid and ask message packets that are stored locally by the node awaiting resolution. Incoming messages are placed into the resolution pool that is an in-memory set of objects. Two processes operate on this pool at regular intervals, the *matchmaker* and the *packet expiry checker*. The packet expiry checker runs through all packets in the resolution pool and checks their Time Of Expiry (TOE). If a packet has passed its TOE it is removed from the resolution pool.

The resolution pool has a maximum capacity. This is because the matchmaking process can be computationally intensive when a large number of packets are involved. The maximum capacity of the local resolution pool can be altered to tune the performance of a node so that CPU utilisation is kept low and does not affect the other tasks being carried out on that node. If the pool becomes full, additional packets that are received may either be passed on without being added to the local resolution pool or the oldest packet in the resolution pool may be removed and replaced with the most recently received packet. This decision is made independently by each node and is part of a node's local configuration.

The matchmaking process uses Condor ClassAds (Raman et al. 1998). A matchmaker is present at each node and compares pairs of ClassAds to identify the optimal matching between a set of messages present in a node's resolution pool.

6 Security

The nature of the MAGOG system means that it is necessary to ensure that all communications between nodes, the clearing houses and the payment services are secure and identifiable. Their validity must not be questionable and non-repudiation is required. The environment must ensure that parties involved in transactions cannot refute the validity of contracts made between them. Attempts to modify messages within the network such as those represented in the architecture by GridBees, DealBees and ChekBees must be identifiable, rendering the messages invalid.

This need for high security within the system results in a conflict with the need for low barriers of entry that will stimulate the arrival of new providers and consumers into the system and hence improve the liquidity of the market. The clearing houses and payment services must provide the utmost security in order to give confidence to market participants that only valid deals will be carried out and that any attempt to intercept or tamper with messages will be detected.

6.1 Public Key Infrastructures

The adoption of a Public Key Infrastructure (PKI) based on X.509 certificates would require every user of the middleware to be given their own digital identity represented by a public/private key pair. This would enable entities to both encrypt and digitally sign data allowing secure data transmission of messages between nodes and identity verification so that the receiver of a message can determine, beyond all reasonable doubt, the identity of the sender.

While this approach has the potential to provide the necessary security, the use of a PKI presents issues in a network such as the GOG that is designed to be open and easily accessible. A PKI requires certificates to be issued by a Certificate Authority (CA). Chains of trust can be generated by a CA devolving trust to a lower level authority and allowing them to issue certificates to others. CAs are responsible for verifying the identity of entities that they issue certificates to and the security of the entire infrastructure is dependent on these identity checks and the security of private keys that are issued to participants in the network. If a private key is compromised or a certificate is issued to a rogue individual by a trusted CA allowing them to participate in the network in a trusted manner, the security of the entire environment may be compromised. Further, security provided through a PKI in MAGOG is deemed to raise the barriers of entry to an unreasonable level. It would require that every user obtains a keypair from a trusted CA and that they then manage to keep their private key completely secure, something that individuals cannot always be guaranteed to do successfully.

A PKI is considered acceptable for the initial testing of the infrastructure in a restricted environment where there is a single Certificate Authority and control over the key generation and distribution process and the individuals that gain access to the environment. However, a significant element of the ongoing further work is the identification of a more appropriate security infrastructure that can withstand the requirement for high levels of security in an open Internet environment.

6.2 GCH and IPS Security Management

The areas of the network where security is most critical are the Grid Clearing Houses (GCHs) and the Internet Payment Services (IPSs). The design of the security infrastructure therefore aims to push as much of the burden of security management onto these entities who should be professionally managed and trusted by third parties making them well placed to manage security and verify user's identities. GCHs handle the DealBees and EndBees relating to a transaction while IPSs handle ChekBees. Nodes participating in the network will have a relationship with one or more GCHs and IPSs. An approach being considered for the security infrastructure is a hybrid system where GCHs and IPSs generate and distribute conversation keys to every client to support the local encryption of messages sent by clients to them. Since GCHs and IPSs will be much fewer in number than users of the system and are considered to be in a better position to manage a public/private key pair, they will each have their own X.509 certificate to identify themselves and will make their public key available for users to verify that they are communicating with a valid GCH/IPS. The reliance placed on the GCHs and IPSs to manage keys securely is not considered to be a problem due to their handling of payments and legal information that already requires the utmost security.

Bid and AskBees will not be encrypted. Although this doesn't allow each of the partners in a deal to verify the other's identity, this identity verification is handled by the GCH and IPS. Rogue bidders that constantly offer goods that they aren't able to provide may reach the point of trying to make deals but if they regularly make false deals, this will be picked up by the GCH to whom they must identify themselves. Such bidders cannot be prevented from flooding the network with false bids and asks but this would still be the case if these entities had to have a certificate that could be used to verify their identity. There are benefits in being able to verify the identity of an entity making bids and asks since genuine traders could ignore these messages preventing the generation of fraudulent deals in the first place, however, the necessary security and identity management to support this would again require significantly increased barriers to entry that would be expected to be detrimental to the success of the MAGOG environment.

There are clearly drawbacks to avoiding a full public-key infrastructure for the MAGOG network, if such a network could be implemented and managed reliably on such a large scale. However, these drawbacks are felt to be far outweighed by maintaining an easily accessible open network that has low barriers to entry for all. Further, the management of a full PKI given such a large number of potential nodes is felt to be impractical as is the need to rely on all individuals to maintain the utmost security of their keys. The benefits and drawbacks of maintaining a completely open network in order to stimulate the number of network participants are to be evaluated in detail when the work reaches the stage of the first deployments of the infrastructure on Internet-connected resources.

7 Conclusion and Future Work

In this paper an overview has been presented of MAGOG, a middleware designed to enable the activation of a Global Open Grid infrastructure. Design of the middleware components is at an early stage and the results of preliminary simulations have been presented. The simulation results are encouraging, showing that the key economic properties that are necessary to support the system do in fact emerge. The technical specification of the middleware stack that must be operational on all nodes participating in the GOG network is also presented. We have described the initial design work carried out to define properties of the stack and investigated suitable implementation technologies and approaches.

MAGOG, while relatively simple in its high-level design and triad of underlying principles, will be complex to implement and this initial design work is the first stage in the process of developing a full technical specification for implementing the first version of the system's services. Future work includes continuing simulation work which will involve testing a variety of network optimisations and determining how the network operates under different P2P models and investigating how the network performs as the numbers of nodes increase. The development of the service specification is also ongoing and it is intended that this will be followed by implementation of an initial version of the software stack in order to begin testing the system with real computational resources on the PlanetLab network.

References

A.-L.Barabasi & R.Albert (1999), 'Emergence of scaling in random networks', *Science* **286**, 173.

Abramson, D., Buyya, R. & Giddy, J. (2002), 'A computational economy for grid computing and its implementation in the nimrod-g resource broker', *Future Generation Computer Systems (FGCS)* **18**(8), 1061–1074.

Arthur, W. B. (1994), 'Inductive Reasoning and Bounded Rationality (The El Farol Problem)', *American Economic Review (Papers and Proceedings)* **84**, 406–411.

Broberg, J., Venugopal, S. & Buyya, R. (2007), 'Market-oriented grids and utility computing: The state-of-the-art and future directions', *Journal of Grid Computing* **6**(3), 255–276.

Bubendorfer, K. (2002), NOMAD: Towards an Architecture for Mobility in Large Scale Distributed Systems, PhD thesis, Victoria University Wellington.

Bubendorfer, K., Welch, I. & Chard, B. (2006), Trustworthy auctions for grid-style economies, *in* 'Sixth IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2006)', IEEE Computer Society, Singapore, pp. 386–390.

Buyya, R., Abramson, D. & Giddy, J. (2000), Economy driven resource management architecture for computational power grids, *in* 'International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA2000)', Las Vegas, USA.

Buyya, R., Abramson, D., Giddy, J. & Stockinger, H. (2002), 'Economic models for resource management and scheduling in grid computing', *Concurrency and Computation: Practice and Experience* **14**(15), 1507–1542.

Buyya, R., Abramson, D. & Venugopal, S. (2005), 'The grid economy', *Proceedings of the IEEE* **93**(3), 698–714.

Csàrdi, G. & Nepusz, T. (2006), 'The igraph software package for complex network research', *InterJournal Complex Systems* p. 1695.

Eymann, T. & et al. (2005), 'Catallaxy-based grid markets', *Multiagent Grid Systems* **1**(4), 297–307.

Ghoshal, G. & Newman, M. E. J. (2007), 'Growing distributed networks with arbitrary degree distributions', *The European Physical Journal B* **59**, 75. **URL:** *doi:10.1140/epjb/e2007-00208-2*

Harder, U. & Martinez, F. (2008), Simulation of a peer to peer market for Grid Computing, *in* 'The 15th International Conference on Analytical and Stochastic Modelling Techniques and Applications, ASMTA 2008', Vol. 5055 of *Lecture Notes in Computer Science*, pp. 234–248.

Huberman, B. A. & Glance, N. S. (1993), 'Evolutionary Games and Computer Simulations', *Proc. Natl. Acad. Sci. USA* **90**, 7716–7718.

Milgram, S. (1967), 'The small world problem', *Psychology Today* **2**, 60–67.

Oram, A., ed. (2001), *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*, O'Reilly and Associates, chapter 14 (Performance) by Theodore Hong.

Raman, R., Livny, M. & Solomon, M. (1998), Matchmaking: Distributed resource management for high throughput computing, *in* 'the Seventh IEEE International Symposium on High Performance Distributed Computing (HPDC)', Chicago, IL, USA, pp. 140–147.

Ratnasamy, S., Francis, P., Handley, M., Karp, R. & Shenker, S. (2001), A scalable content-addressable network, *in* 'ACM Special Interest Group on Data Communication (SIGCOMM) 2001', San Diego, CA, USA, pp. 161–172.

Richardson, C. (2007), Growing the Global Open Grid: Design Brief and Middleware Architecture, Technical report, The Internet Centre, Imperial College London.

Rowstron, A. & Druschel, P. (2001), Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems, *in* 'IFIP/ACM International Conference on Distributed Systems Platforms (Middleware 2001)', Heidelberg, Germany, pp. 329–350.

Simon, H. (1957), *Models of Man*, chapter A Behavioral Model of Rational Choice.

Stoica, I., Morris, R., Karger, D., Kaashoek, F. & Balakrishnan, H. (2001), Chord: A scalable peer-to-peer lookup service for internet applications, *in* 'ACM Special Interest Group on Data Communication (SIGCOMM) 2001', San Diego, CA, USA, pp. 149–160.

The FastTrack Protocol (n.d.), `http://developer.berlios.de/projects/gift-fasttrack`.

The Gnutella Protocol Specification v0.4 (n.d.), `http://www9.limewire.com/developer/gnutella_protocol_0.4.pdf`.

Tsoumakos, D. & Roussopoulos, N. (2006), Analysis and comparison of p2p search methods, *in* 'proceedings of the 1st International Conference on Scalable Information Systems (InfoScale 06)', ACM, New York, NY, USA.

Wang, F., Moreno, Y. & Sun, Y. (2006), 'Structure of peer-to-peer social networks', *Physical Review E* **73**, 036123.
URL: *doi:10.1103/PhysRevE.73.036123*

Watts, D. J. & Strogatz, S. H. (1998), 'Collective dynamics of 'small-world' networks', *Nature* **393**(6684), 409–410.

Zhao, B. Y., Huang, L., Stribling, J., Rhea, S. C., Joseph, A. D. & Kubiatowicz, J. D. (2004), 'Tapestry: a resilient global-scale overlay for service deployment', *IEEE Journal on Selected Areas in Communications* **22**(1), 41–53.