Auction Based Resource Negotiation in NOMAD

Kris Bubendorfer **John H. Hine**

School of Mathematical and Computing Sciences,
Victoria University of Wellington,
P.O.Box 600, Wellington 6001, New Zealand
Email: {Kris.Bubendorfer, John.Hine}@vuw.ac.nz

Abstract

The challenges faced by mobile and distributed applications include the ability to discover and negotiate for the resources required for execution. The NOMAD (Negotiated Object Mobility, Access and Deployment) system is a middleware platform that provides an infrastructure to support applications constructed of mobile code. This paper describes the NOMAD mechanisms for resource discovery and negotiation. NOMAD features a Marketplace providing a forum in which multiple resource requirements lead to contracts for the allocation of resources between applications and resource providers. NOMAD's Marketplace is also a useful model for resource allocation in distributed systems such as grid computations. Experimental results show that resources are allocated consistent with the policies of both the application and the resource provider.

Keywords: mobility, resource allocation, resource discovery, market negotiation, computational economy

1 Introduction to NOMAD

The adoption of distributed computing paradigms such as mobile devices, agents and grid computing have demonstrated the ability of one organisation's software to use resources provided by another organisation. We envision a future world in which a global market for computing resources exists. The resource providers in this market may be organisations with a temporary surplus of resources or organisations whose sole business is to provide resources for a profit. This paper presents NOMAD. NOMAD is a set of middleware services that supports distributed applications by providing resource discovery and negotiation in the context of a global market. The NOMAD design is suitable for distributed paradigms based on either mobile code or initial placement.

NOMAD consists of loosely coupled cooperating virtual machines, named Depots, that host distributed applications. Each Depot independently provides a collection of local resources ranging from CPU cycles through to specific hardware and software libraries, which are made available to applications in return for payment. Applications utilise these resources to provide services to their clients, and negotiate to alter their resource profile as their resource needs change (Figure 1).

The motivation behind NOMAD is to provide a coherent global infrastructure as a bridge between applications and the Depots on which they execute. This infrastructure includes resource discovery and negotiation, location services and a payment service. Resource discovery and negotiation includes mechanisms describing resources, performing negotiation, ensuring the integrity of negotiations and ultimately, policing contract compliance.

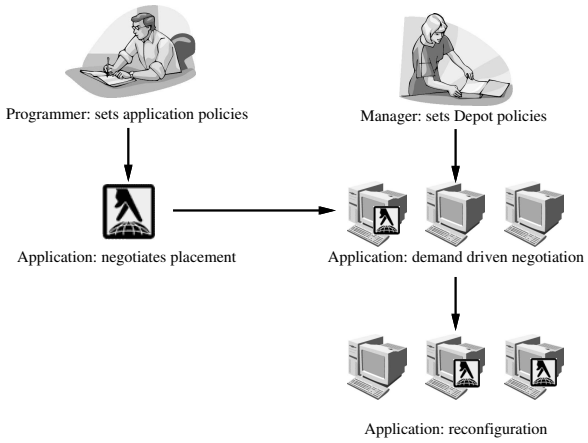


Figure 1: *High level view of NOMAD*

A NOMAD application is a logical collection of mobile code that provides one or more services (functions). Each application is uniquely identified and manages its collective parts in a cohesive fashion to best deliver its service. Applications may provide services to one or many clients[1]: a web robot, a stock ticker, a chatroom or video rendering. Services are identified publicly through a Yellow Pages Directory service and have well known value to their clients. Applications may be either long lived or relatively short lived, such as a typical personal computer application.

An application may be structured in a variety of ways to best serve its clients. Moving closer to the client or a fixed resource is a well recognised technique to improve performance (Lange & Oshima 1999, Foster, Kesselman & Tuecke 2001). Each application controls its degree of distribution by placing and replicating its clusters to achieve goals such as reduced latency to clients, redundancy and concurrent processing. Objects within an application are grouped into mobile clusters, and each cluster inherits functionality supporting mobility and resource negotiation. References outside the cluster, take place via proxies (created on the fly) so that an object in one cluster may invoke a method on an object in a different cluster, independently of either clusters' current location.

An application negotiates with the Depots via the NOMAD Marketplace for the execution resources that it requires. *The Marketplace provides both re-*

[1] An application may itself be a client of another application.

source discovery and negotiation services. For their part, Depots use policies to price their resources and to organise such things as the balance of work amongst a federation of Depots, or the quality of service ranges that individual Depots offer. A federation of Depots reflects common ownership, or administration of the member Depots.

NOMAD provides the global infrastructure that integrates Depots and applications. In particular, the provision of migration support, a location service (Bubendorfer & Hine 2003) and an impartial resource negotiation mechanism. The objective of this paper is to provide an overview of the mechanism used to negotiate the purchase of resources within the NOMAD architecture.

1.1 Related Work

A comprehensive survey of mobile agent systems is carried out in (Gray, Cybenko, Kotz, Peterson & Rus 2001). However, the majority of these systems implicitly utilise a cooperative model, in which resources are free to all and no attempt is made to manage the allocation of execution resources. This cooperative model does not reflect the reality of multiple administrative domains, ownership of machines and resources, and ignores abuse — both malicious and accidental. Examples of these systems are: Aglets (Lange & Oshima 1998), NOMADS (Suri, Bradshaw, Breedy, Groth, Hill, Jeffers & Mitrovich 2000), and Globe (van Steen, Homburg & Tanenbaum 1999).

The various grid computing models make similar assumptions. Systems such as Condor (Thain, Tannenbaum & Livny 2004), Legion (Grimshaw, Ferrari, Knabe & Humphrey 1999) and Globus (Foster et al. 2001) make resource allocation decisions based on performance information. Globus has been designed to facilitate resource sharing over multiple organisations, however it assumes that prior agreement has been reached on the sharing of resources.

Markets are particularly appropriate for resource allocation in situations where software is spread across a distributed system, serving different clients and pursuing different goals (Miller & Drexler 1988). Indeed, such systems have been used for the allocation of computational resources since 1968 (Sutherland 1968), although in this early case the auction involved the use of a whiteboard and pens to manually allocate advance bookings on a departmental PDP1.

In (Drexler & Miller 1988) and (Miller & Drexler 1988) the authors developed a computation market for the efficient auctioning of processor time to non-concurrent processes on a single processor computer system. This was also probably the first system which considered the application of the various forms of auction protocol to computer resource management. Around the same time, Enterprise (Malone, Fikes, Grant & Howard 1988) was developed for distributed computing systems. Enterprise performed initial placement of tasks to processors, using a protocol based on that of contractNet (Smith 1980). In place of currency, Enterprise used an estimate of priority based on the expected processing time. Cheating by understating processing time estimates was policed by automatically aborting processes that significantly overran their estimate. (Kurose & Simha 1989) investigated an economic iterative auction model for the distributed allocation of files and demonstrated that the attractive properties of auctions transferred into the computational domain.

Spawn (Waldspurger, Hogg, Huberman, Kephart & Stornetta 1992) was the first full attempt to create a distributed computational economy, and utilised bid escalation as described in (Drexler & Miller 1988, Miller & Drexler 1988). As with the previous systems, the only resource negotiated for was processor time, and the currency in the system was allocated to users by human system administrators. Users were prioritised by allocating different amounts of currency as in (Sutherland 1968). Each computer in Spawn executed two processes, an *auctioneer* and a *resource manager*. The resource managers each contained a small list of auctioneers on neighbouring computers which supplied updates of pricing and availability. The auctioneer continuously accepted bids on the next available slice of time. Spawn was limited to initial placement, and thus on expiry of an existing processor slice, the choice was either to abort the application or to let it continue execution. The approach used was to leave the decision to the application, that is, by offering the application first-refusal basis on the next processor slice at the current price. When an application wished to *spawn* a new child, it sent a request to the local resource manager which then matched the applications CPU requirements against those available on the monitored auctions, and bid on its behalf.

Xenoservers (Yan, Early & Anderson 2000, Reed, Pratt, Menage, Early & Stratford 1999) also promote the ideal of a computational economy with mobile objects. The planned system parallels the NOMAD architecture, with both Xenoservers and NOMAD Depots forming loci of resource and computation.

D'Agents (Gray et al. 2001) shares a market based approach with NOMAD, although a number of important differences exist between the two systems. NOMAD does not utilise access control lists as does D'Agents, but rather relies on incentive based control within the economic model. Negotiation is application initiated in NOMAD, whereas in D'Agents servers auction their spare capacity.

(Buyya, Abramson, Giddy & Stockinger 2002) explores how different economic models might be applied to the grid computing paradigm. For most cases a general approach is discussed without attention to the efficiencies of implementation. The authors have successfully applied a market based approach to a parametric sweep problem solved on a global grid.

NOMAD is unique in pursuing the creation of a market driven global infrastructure for mobile and distributed applications.

1.2 Economic Resource Management

Traditional resource management has been undertaken by the operating system. In this role, it has the conflicting goals of optimising the resource allocation for the entire system (*globally*) and for each application (*locally*). This is difficult as the operating system is neither aware of, nor privy to, the needs or goals of each application. A solution to this problem is to separate the local optimisation from the global optimisation, that is, the application must negotiate for the resources it requires from the system on which it wishes to execute (Stratford & Mortier 1999).

This is similar to the problem of allocating finite resources over an independent population — as faced by all human societies. For example, a farmer wishes to get the best price for his wheat, while the baker wishes to pay less. The solutions to this and related problems have been refined by many years of human competition and form the basis of economics (Miller & Drexler 1988). This well founded and understood technique of *resource pricing* has long been applied to the problem of allocating finite computational resources (Sutherland 1968).

The economic model is also powerful enough to provide a mechanism, with which to solve a number

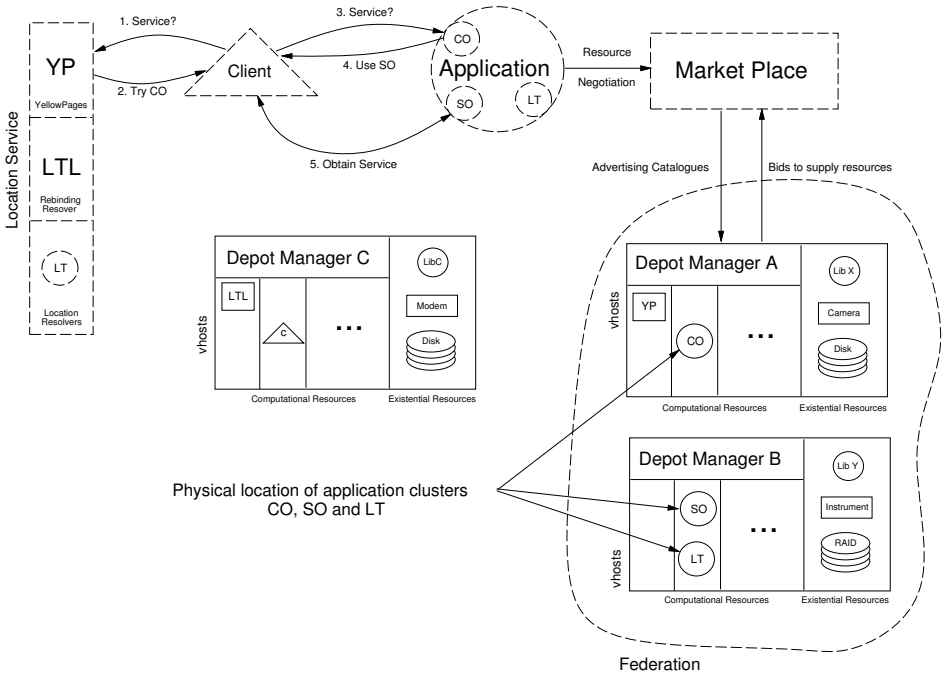


Figure 2: *Entities within the NOMAD system.*

of other problems relating to the behaviour of entities within the system. One such problem in open mobile systems is the malicious (or otherwise) wasting or over-using of resources by a visiting entity. This is a serious concern when the host and object have different administrative domains or ownership. Rather than design a system that predicts the entire range of possible actions an entity might take, and then permits or forbids those actions depending on the probable impact on the system, financial incentives (Drexler & Miller 1988) can be used to influence the behaviour of entities within the system. An example of this is the denial of service attack which, if the instigator is charged for the actual resources consumed, becomes infeasible.

The incentives that are designed into an economic system ensure that it is in an entity's *best interest* to behave in an acceptable way (Sandholm 1996*b*). Infractions cost real currency, and as a consequence badly behaving entities suffer a real financial loss and are unable to utilise additional resources once their currency is expended. This is not a complete panacea, as the system must still ensure that the entities within the system are protected from other forms of interference, and that the payment and negotiation systems are secure. Nonetheless, utilising an incentive model is a rational choice and motivates the use of the market based approach for resource management in NOMAD.

1.3 NOMAD Overview

Figure 2 gives an overview of the major components within the NOMAD architecture. Each Depot (Bubendorfer & Hine 1999) consists of a Depot Manager and a set of managed virtual hosts (vHosts). Management policies dictate how each Depot will react to specific circumstances: how it will ensure levels of service and how it interacts with Depots within and outside of its federation. The Depot Manager is also responsible for responding to negotiation, charging, and arranging the hosting of the global NOMAD infrastructure components. A vHost is a virtual machine on which application code executes. A vHost is responsible for managing physical resource use, security, fault detection, and communication.

The application in Figure 2 consists of clusters that physically reside on vHosts within Depots A and B, and provides a set of services to its client C.

From the perspective of an application, the Marketplace fulfils both the role of resource discovery and allocation. When an application requires resources, it registers its resource requirements as an auction with the Marketplace. The Marketplace distributes this information to the bidders (the Depots that are potential providers of the required resources) in a catalogue and in turn collects any bids. Bids are made by Depots to secure the right to host the application, and thereby generate revenue from the sale of its resources to applications. Clearly the application wishes to obtain the resources at the lowest possible price, whereas the Depot wishes to maximise the return on its resources. A good analogy is a private building company bidding to fulfil a government construction contract.

2 The Negotiation Service

Markets make an ideal tool for economic resource management.

> *Market price systems constitute a well understood class of mechanisms that under certain conditions provide effective decentralisation of decision making with minimal communication overhead — (Wellman 1993).*

A Market is clearly the suitable choice for NOMAD, with distributed decision making, minimal overhead, well understood theory and the abstraction of resources as currency. This permits the NOMAD system to charge for the use of resources, prevent unbounded resource attacks as with denial of service (Yan et al. 2000), and represent priority. A Market provides a forum in which buyers and sellers meet (resource discovery) and negotiate to perform optimal allocations with minimal overhead, mutual selection and decentralised decision making between buyers and sellers.

The use of different market models for distributed computing has been explored in (Buyya et al. 2002).

For NOMAD we chose an auction model. The auction model allows a resource provider to follow a range of market policies such as commodity, posted price, or dynamic pricing. For example, to follow a posted price model the provider might simply check the time of day and respond with the appropriate cost. Each request for resources elicits a response from a set of interested providers. This eliminates the need for the Marketplace to maintain potentially complex pricing policies for each resource provider.

NOMAD applications direct their own distribution, therefore they initiate negotiation for the resources they require. Depots take on a competitive provider role, bidding to supply their resources and collect revenue from the applications they host.

2.1 Selection of an Auction Protocol

For use in a computational economy, like NOMAD, an auction protocol must lead to optimal allocations with minimal overhead. In this section we look at the suitability of different types of auction protocols.

There are four main types of auction protocol identified by Vickrey (Vickrey 1961); the English, Dutch, Sealed-Bid, and what has since become known as the Vickrey auction protocol. The English auction is the conventional open forum, ascending price, multiple bid protocol. The Dutch auction is an open forum, descending price, single bid protocol. The Sealed-Bid, or tender, is a closed forum, single bid, best price protocol in which all bids are opened simultaneously. The Vickrey auction is similar to the Sealed-Bid auction, except that the winning (highest) bid then pays the amount of the second highest bid.

Messaging overhead is incurred when multicasting the current state of an auction to all bidders. This overhead is particularly costly in the case of the open forum protocols, where each (potential) bidder must be kept informed of the current price. In the English auction — each bidder needs to know every bid that has been made. This in turn raises a question of fairness, with reliable and prompt delivery of updates required, lest a bidder miss out due to network delay or loss. The second source of overhead is in the auctioneer, which must collect and process multiple bids, and select the winner. An agreement protocol is required to close multiple bid auctions. Selection of a winner can require considerable computational effort, especially in computational resource auctions, as is discussed in section 2.2.

The most significant result in auction theory is the **revenue equivalence theorem** (Vickrey 1961). This states that all four auction protocols yield the same expected return in private value[2] auctions. In addition, the strategies and payoffs associated with the Dutch and Sealed-Bid auction protocols, even for non private-value auctions, are the same. That is, the Dutch and Sealed-Bid auction protocols are strategically equivalent, in all valuation models, as only the winning bid matters and no information is revealed during the auction process.

In private value auctions the English and Vickrey auction protocols produce the same allocations (at the same prices), where the bidder who values the item most wins it, but does so with different strategies. However, the English and Vickrey auction protocols are not equivalent in non-private value auctions, as the open outcry nature of the English auction protocol provides additional information to the bidders, which can then alter their valuations. (Milgrom & Weber 1982) shows that in correlated value auctions, this enables a variation of the English[3] auction protocol to generate greater revenue than the Vickrey, Dutch and Sealed-Bid protocols when there are three or more bidders.

If a certain strategy pays a player the highest payoff, regardless of other players' strategies, then that strategy is known as a dominant strategy. Neither the Dutch nor the Sealed-Bid auction protocols have a dominant strategy. The dominant strategy for the English auction protocol is to bid a small increment over the current bid price and stop when the private value is reached. The dominant strategy for the Vickrey auction protocol is to bid the true value of the item. This strategy has two important and beneficial side effects:

- bidders reveal their values accurately allowing for globally efficient allocations, and
- bidders need not waste efforts in attempting to counter-speculate other bidders.

Counter-speculation wastes computational resources and introduces considerable complexity into the system, while not improving overall allocations.

An additional efficiency is achieved by using a protocol requiring a single bid based on a static description of the lot[4] and auction. That is, the bidder submits a single bid and does not need to be advised of changes to the state of the auction, other than the outcome. This minimises the number of messages exchanged during an auction.

The Vickrey auction protocol[5] is a single bid Pareto-optimal[6] (Rasmusen 1994) technique that is efficient in terms of messages and allocation. This suggests it is ideal for machine to machine negotiation and is the auction protocol selected for use in NOMAD.

There are however, seven limitations concerning the applicability of the Vickrey auction protocol to computational systems: the lying auctioneer, bidder collusion, possible release of sensitive information, lying in non-private value auctions, lower revenue in non-private value auctions, sub-optimal allocation and lying in interrelated auctions, uncertainty of valuation and wasteful counter-speculation. These limitations have been identified in the literature (Rasmusen 1994)(Sandholm 1996*a*)(Vickrey 1961), and are probably amongst the reasons why the protocol has been abandoned in some recent projects.

The Vickrey auction protocol provides an ideal basis for automation if the specific design of the market is able to successfully address its limitations. How the design of the NOMAD Marketplace addresses these limitations is discussed in section 2.4.

2.2 Describing Multiple Requirements

Another challenge is the focus of standard auction mechanisms on the sale of a single good. This is fine when the good is a single unit, such as a soccer ball, but not when the good in question is, say, one sock

[2] In private value auctions the valuation depends solely on the bidder's own preferences. In contrast, the valuation in a common value auction depends entirely on the other bidders' values of the item. Correlated valuations are a weighted combination of private and common values.

[3] The protocol is **open exit** in which all bidders are aware of other bidders ceasing to bid.

[4] A *lot* is an individual set of goods in an auction, which are bid on as a unit.

[5] It is worth pointing out that even small modifications to this protocol may seriously damage its dominant strategy of truthful bidding. For example, in (Drexler & Miller 1988, Miller & Drexler 1988) and Spawn (Waldspurger et al. 1992), the addition of the *escalation* mechanism reintroduces counter speculation and non-truthful bidding.

[6] A distribution of resources is Pareto-optimal if any redistribution of resources which is beneficial to one or more individuals is also detrimental to one or more others.

— which has its greatest value when part of a pair. Likewise, execution resources form an indivisible set, related and conditional upon the availability of each other. Piecewise negotiation of these individual resources will not give any usable result let alone an optimal allocation. After all, it is very difficult to execute when the memory is on a different host from the allocated CPU cycles. This is the combinatorial allocation problem (CAP) (Rothkopf, Pekeč & Harstad 1995, Parkes 2001), in which a set of components have a synergistic value that exceeds the sum of the individual parts. Because of synergistic combinations and substitution effects, bidders have preferences not just for particular items, but for collections of items.

The solution to the CAP is the Generalised Vickrey Auction (GVA) (MacKie-Mason & Varian 1994), which utilises a single auctioneer that is required to solve the CAP for all resources and all bidders. This requires however, not just a single *NP*-complete computation, but multiple *NP*-complete computations to find the optimal solution. Approximations such as iBundle (Parkes 1999) still require that the auctioneer perform multiple *NP*-complete computations.

Clearly these existing approaches are untenable in practice, and would severely limit the scale of any system in which they were utilised. Our approach is to distribute the computation of the CAP amongst the Depots that choose to bid. A NOMAD application employs a resource description graph (RDG) to describe the resources it requires. A RDG is a single directed acyclic graph which describes an auction, its set of lots and possible alternatives for each lot.

Each RDG has one or more accept states. Each accept state represents an independent lot and each lot can be won by a different Depot. An application may choose to package several lots in one RDG in the hope of achieving savings through the sharing of resources required by the different lots. A Depot may choose to bid on any subset of the lots described in a RDG. An entirety bid (Hurwicz, Schmeidler & Sonnenschein 1985), is a bid for all the lots in the RDG and allows the bidder to benefit from savings resulting from the combined bids.

Each sentence (path to an accept state) through the RDG identifies a set of resources required to fulfill a particular task. A Depot bidding on this lot is expected to provide all resources specified in one of the sentences leading to the accept state. The classes of resources that can be specified in a RDG sentence include hardware, software, network access, protection domain access, quality of service parameters and specific specialised resources. Alternate sentences ending in the same accept state define acceptable compromises. For example two sentence might represent a trade off between network bandwidth and memory that can be used for buffering or caching.

The RDG provides a general representation of resources and limits bidding to combinations of resource allocations that are deemed to be valid by the creator of the RDG. This restriction means that the Depot is faced with a smaller and better defined problem. Further, NOMAD utilises the combination of RDG and entirety bidding to replace the *NP*-complete CAP with a good approximation enabling multi-component auctions with the standard Vickrey auction protocol.

This approximation does not eliminate all *NP*-complete computations, however, it does reduce the size of the problem set and move the computations from the auctioneer to the Depot (bidder). The Depots compute their bids concurrently and the load is widely distributed.

There are two main points of caution that we have identified. Firstly, the size of the RDGs cannot be too large or the load on each Depot becomes excessive. Secondly, the edge expressions which compose the RDG must be carefully specified to minimise computation. Malicious exploitation of these limitations could form routes of attack on the negotiation system and must be rigorously policed by both the Marketplace and the Depots themselves.

The reduced *NP*-complete problem at each Depot can be entirely eliminated by further restricting the Depot to:

- making either just a single lot bid or an entirety bid for each auction, and
- waiting on the outcome of that auction before bidding again.

Providing that the size and edge specifications are enforced, the bid computation will be sufficiently tractable to make these restrictions unnecessary.

In addition, this approach avoids the difficulty of balancing the books (Parkes, Kalagnanam & Eso 2001) and the computation of the social welfare of the system (MacKie-Mason & Varian 1994), as the winner simply pays the second price. This further reduces the load on the entire system and particularly on the auctioneer. Privacy is enhanced as Depots need only reveal a single bid value to the Marketplace (auctioneer). In addition, the Marketplace need not parse and understand RDG edge expressions — this evaluation is performed by the Depots interested in bidding[7].

2.3 The NOMAD Marketplace

The NOMAD Marketplace is a distributed resource allocation system, implemented by a set of cooperating Markets. The Marketplace enables applications to negotiate and pay for execution resources. It performs the related functions of resource discovery and negotiation. The mechanism, as shown in Figure 3, is straightforward, allowing computationally bound clients to construct resource descriptions from building blocks and to negotiate efficiently.

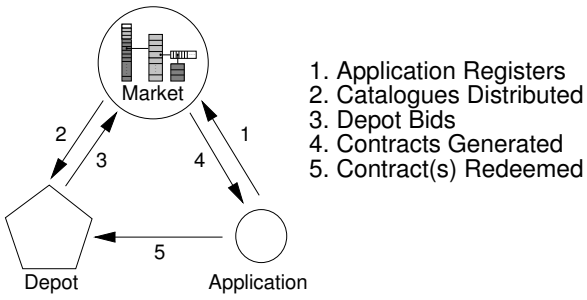


Figure 3: *The Market Protocol.*

Applications requiring resources construct a RDG describing their requirements and register an auction with a Market. The Market receiving the RDG constructs a set of catalogues consisting of all RDGs from current auctions. These catalogues are then distributed to both Depots and other Markets known to this Market. This is the resource discovery mechanism.

Negotiation in NOMAD is performed using the principles of the Vickrey auction protocol. Depots compute valuations based on the information within the catalogues and submit their bids to the Market conducting the auction. Once the auction is over, the Market identifies the winner(s) and generates signed contract certificates that are passed to the application. The application then redeems these contracts

[7] A Depot looking to bid will, by inference, have spare resources. Using these resources to evaluate potential clients is sensible.

at the appropriate Depots and obtains the negotiated resources.

The Marketplace does not seek globally optimal allocations, rather it divides the applications and Depots into geographic and categorical regions. Auctions are advertised between regions via the distribution of catalogues between regional Markets.

Bipartite auctions (in which only one Depot is eligible to bid) are simulated within the multipartite auction protocol, in such a way that ensures a target Depot will bid its true value.

2.4 Addressing the Limitations of the Vickrey Auction

The Vickrey auction is an ideal basis for automatic resource allocation, once its limitations are addressed. This section outlines how each of the limitations is satisfied within the design of the NOMAD Marketplace. Additional details are available in (Bubendorfer 2001).

The problem of the *Lying Auctioneer* can only be ultimately resolved by making the Marketplace a trusted core NOMAD service in much the same way that filesystems are trusted. Additionally, the Marketplace only charges the application a set fee rather than a commission based on sale price, removing the temptation to misrepresent the second price. *Bidder Collusion* is only possible when bidders can identify other potential bidders and form a cartel. NOMAD is an agoric system in which Depots are free to join or leave — consequently the pool of bidders is dynamic. The only entity that has access to bidder identities is the Market conducting the auction, which is trusted. As the auction is sealed, collusive bidders have no means of detecting if or how an independent bidder has bid, unlike in the English auction. The problem of *Revelation of Sensitive Information* is solved by the construction of the post-auction contracts. Specifically the Market signs the second price bid itself and the winner is not informed of the source of the second highest bid, but can trust the value as signed by the Market to ensure that it is not being cheated.

To solve the problems of *Lying in non-Private Value Auctions* and *Lower Revenue in non-Private Value Auctions*, contracts are strictly non-transferable. This property is enforced by the Marketplace mechanism. A contract and therefore the potentially resalable lot does not exist until the application redeems the contract at the successful Depot. As the contract (before redemption) is not binding on the application, and the Depot is not aware if it is the winner before the contract is presented, there is nothing to on-sell. At the time the contract is redeemed, the resources must be made immediately available to the application. It follows, that since contracts are non-transferable, all auctions in the Marketplace are private value and neither of these two limitations apply. Depots acting cooperatively within a federation may still balance their load by utilising the same algorithm that airlines use - offering financial incentives for a contract (ticket) holder to move to another prearranged Depot (flight) and redeem its contract (travel) there.

The provision of entirety bidding and the mechanisms supporting it removes the possibility of counterspeculation and lying when Depots are bidding on related lots. This resolves the issue of *Sub-optimal Allocation and Lying in Interrelated Auctions.*

Uncertainty of Valuation arises from the potential complexity computing a bidder's true valuation. This is minimised by two factors in the NOMAD context. Firstly, the resource description graph is a directed acyclic graph and therefore finite, and secondly the language of edge expressions is designed for ease of computation. Even further, only risk averse bidders are affected by local uncertainty, and as risk is proportional to stake, in the low value NOMAD auctions there are likely to be few risk averse bidders. However, even assuming there is local uncertainty, in (Bubendorfer 2001) we show that the necessary conditions underlying the proof in (Sandholm 1996*a*) are equivalent to bidder collusion and this limitation is therefore infeasible under the negotiation conditions maintained within the NOMAD Marketplace.

3 Implementation

A NOMAD prototype has been implemented in Java on top of Flexinet (Hayton & the Advanced Networked Systems Architecture Team 1999). The current version features implementations of all of the system components described in this paper. The design of the system is not bound to Java or Flexinet, and alternative platforms and language support are being investigated.

3.1 Experimental Performance

The goal of the experiments presented in this section is to demonstrate the impact of valuation policies on the allocation of resources. The experiments were carried out on a grid of identical machines, each hosting a single Depot. To demonstrate the influence of policy on distribution, we formed the Depot bid valuations on the arbitrary basis of their (x, y) position on the grid, plus the current load n. The inclusion of the current load prevents any single Depot from winning all auctions.

The first experiment uses a Depot valuation policy of $(x + y) + n$. The results in Table 1 show the test applications clustering on the Depots with low grid references and therefore correspondingly lower bids[8]. The result matrix in Table 1 shows the average load at each grid position, and the 95% confidence intervals.

Table 2 shows the effect of changing the weighting of one of the grid components, with the Depot valuation policy $(2x + y) + n$. The resulting data and graph illustrates the effect nicely with a significant preference exhibited for one side of the grid.

Depot policy is not the only influence on resource allocation, as the application is able to express constraints and preferences via its RDG. In the following experiment the test applications express a preference to be near a specific Depot. Depots use a *uniform*[9] valuation policy based on the load n and the Euclidian distance from the current Depot (x', y') to the preferred Depot (x, y):

$$n + \sqrt{(x - x')^2 + (y - y')^2}$$

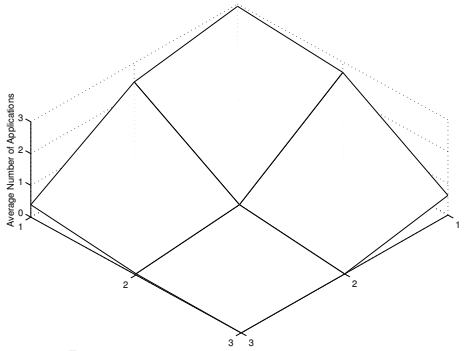
Results are given Table 3 for a five by five grid with the preferred Depot in the centre. The clear peak in the middle is confirmation that the application preferences encoded in the resource description graph directly influence distribution.

3.2 Mobile Application Prototypes

(Liu & Liu 2000) describes a prototype distributed meeting scheduler implemented as a mobile user agent. The agent travels amongst multiple machines, following a schedule, and querying each machine to obtain its owner's diary. NOMAD supports such applications, as the *ownership* of a machine is a constraint that can be specified during negotiation. A

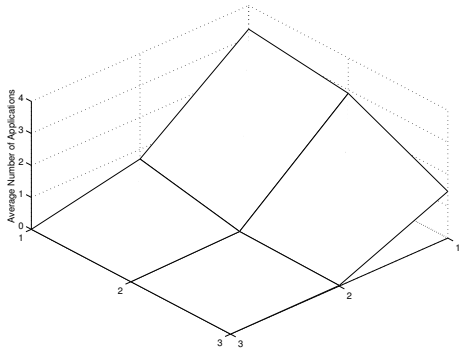
[8] The graph has been rotated so that Depot $(3, 3)$ is at the front.

[9] All Depots have the same valuation policy, unlike the earlier *non-uniform* policies.



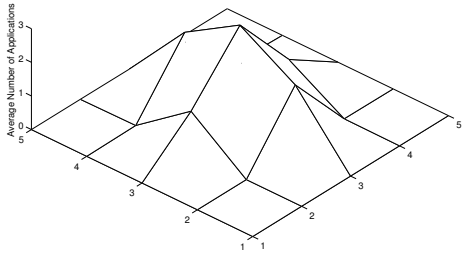
$$\bar{n}_{xy} \pm \sigma_{\bar{n}} = \begin{bmatrix} 2.93 \pm 0.18 & 2.40 \pm 0.17 & 0.39 \pm 0.06 \\ 2.67 \pm 0.18 & 0.36 \pm 0.05 & 0.04 \pm 0.01 \\ 0.62 \pm 0.07 & 0.00 \pm 0.00 & 0.00 \pm 0.00 \end{bmatrix}$$

Table 1: Depot policy: non uniform valuation.



$$\bar{n}_{xy} \pm \sigma_{\bar{n}} = \begin{bmatrix} 3.26 \pm 0.17 & 0.70 \pm 0.07 & 0.00 \pm 0.00 \\ 2.98 \pm 0.17 & 0.07 \pm 0.02 & 0.00 \pm 0.00 \\ 1.46 \pm 0.11 & 0.02 \pm 0.01 & 0.00 \pm 0.00 \end{bmatrix}$$

Table 2: Nonuniform valuation with greater grid weighting on x.



$$\bar{n}_{xy} \pm \sigma_{\bar{n}} = \begin{bmatrix} 0.00 \pm 0.00 & 0.00 \pm 0.00 & 0.00 \pm 0.00 & 0.00 \pm 0.00 & 0.00 \pm 0.00 \\ 0.00 \pm 0.00 & 0.00 \pm 0.00 & 1.92 \pm 0.13 & 0.01 \pm 0.01 & 0.00 \pm 0.00 \\ 0.00 \pm 0.00 & 1.24 \pm 0.10 & 2.90 \pm 0.18 & 0.98 \pm 0.09 & 0.00 \pm 0.00 \\ 0.00 \pm 0.00 & 0.02 \pm 0.01 & 1.89 \pm 0.14 & 0.00 \pm 0.02 & 0.00 \pm 0.00 \\ 0.00 \pm 0.00 & 0.00 \pm 0.00 & 0.00 \pm 0.00 & 0.00 \pm 0.00 & 0.00 \pm 0.00 \end{bmatrix}$$

Table 3: Application policy influences distribution.

similar application was implemented in NOMAD, although in our case it was a test of the technology rather than a useful application in its own right. The application would negotiate for resources on two Depots, and then migrate continuously between them, carrying a small payload of data each time.

This and the test applications used in the experiments from section 3.1 are not useful applications in their own right. We have also constructed a number of test applications to explore the capabilities of the NOMAD architecture. One of these applications was a whiteboard service, of which we constructed two versions to explore the flexibility of NOMAD with both client-server and distributed programming paradigms. The distributed version of the service would respond to new clients by creating a whiteboard service object near the client. From this point the client would interact with the local service object. Any updates were then coordinated amongst the peer whiteboard service objects. Both versions were constructed in less than half a day.

Of the two versions, the distributed whiteboard is the most instructive. Negotiation with a well known contact agent resulted in a new service object being created close to the location of the client. These local service agents act in concert with other client's whiteboard service agents to keep the client views weakly consistent[10]. The contact agent only performed duties involving the initial creation of the service agents for clients. This prototype could be easily extended to provide additional functionality — in the case of physically mobile clients, each service agent could shadow the movement of their client throughout the network.

These prototype applications demonstrate the strength of the NOMAD platform in the creation of distributed software. In particular the ease with which both paradigms were implemented, demonstrating the effectiveness of the supporting NOMAD framework.

4 Summary

This paper presents an overview of the design of the NOMAD Marketplace. The Marketplace utilises auctions to perform distributed resource allocation on the basis of multiple resource requirements. The combination of auction and catalogue distribution enable the discovery of computational and other types of resource by independent mobile entities within a dynamic distributed environment.

This paper outlines how the limitations of the Vickrey auction are satisfied within the design of the NOMAD Marketplace. The Marketplace moves the workload from a single central auction server and distributes it widely to the individual Depots.

The combination in NOMAD of Vickrey auction, resource description graph, Market mechanisms, and catalogue distribution, provides unique and innovative solution to the problem of global distributed resource allocation and discovery.

References

Bubendorfer, K. (2001), NOMAD: Towards an Architecture for Mobility in Large Scale Distributed Systems, PhD thesis, Victoria University of Wellington.

Bubendorfer, K. & Hine, J. H. (1999), DepotNet: Support for Distributed Applications, *in* 'Proceedings of INET'99, Internet Society's 9th Annual Networking Conference'.

Bubendorfer, K. & Hine, J. H. (2003), NOMAD: Application Participation in a Global Location Service, *in* M.-S. Chen, P. K.Chrysanthis, M. Sloman & A. Zaslavsky, eds, 'Lecture Notes in Computer Science', number 2574 *in* 'LNCS', Springer Verlag, pp. 294–306. Proceedings of the 4th International Conference on Mobile Data Management, 21-24 January, 2003, Melbourne, Australia.

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**, 1507–1542.

Drexler, K. E. & Miller, M. S. (1988), Incentive Engineering for Computational Resource Management, *in* H. B.A, ed., 'The Ecology of Computation', Elsevier Science Publishers (North-Holland), pp. 231–267.

Foster, I., Kesselman, C. & Tuecke, S. (2001), 'The anatomy of the grid: Enabling scalable virtual organizations', *International Journal of Supercomputer Applications* **15**(3).

Gray, R. S., Cybenko, G., Kotz, D., Peterson, R. A. & Rus, D. (2001), 'D'Agents: Applications and performance of a mobile-agent system', *Software—Practice and Experience* **32**(6), 543–573.

Grimshaw, A., Ferrari, A., Knabe, F. & Humphrey, M. (1999), 'Wide-area computing: Resource sharing on a large scale', *Computer* **32**(5), 29–37.

Hayton, R. & the Advanced Networked Systems Architecture Team (1999), *FlexiNet Architecture*, Citrix Systems (Cambridge) Limited, Poseidon House, Castle Park, Cambridge, CB3 0RD, United Kingdom.

Hurwicz, L., Schmeidler, D. & Sonnenschein, H., eds (1985), *Social Goals and Social Organization: Essays in Memory of Elisha Pazner*, Cambridge University Press, chapter The Economics of Competitive Bidding: A Selective Survey, Paul R. Milgrom, pp. 261–289.

Kurose, J. F. & Simha, R. (1989), 'A Microeconomic Approach to Optimal Resource Allocation in Distributed Computer Systems', *IEEE Transactions on Computers* **38**(5), 705–717.

Lange, D. B. & Oshima, M. (1998), *Programming and Deploying Java(TM) Mobile Agents with Aglets(TM)*, 1st edn, Addison-Wesley.

Lange, D. B. & Oshima, M. (1999), 'Seven Good Reasons for Mobile Agents', *Communications of the ACM* **42**(3), 88–89.

Levy, H. M. & Tempero, E. D. (1991), 'Modules, Objects and Distributed Programming: Issues in RPC and Remote Object Invocation', *Software Practice and Experience* **21**(1), 77–90.

Liu, M. & Liu, Y. (2000), A Prototype Mobile-Agent Application, *in* 'the 15th International Conference on Computers and their Applications (CATA)', New Orleans, Louisiana.

MacKie-Mason, J. K. & Varian, H. R. (1994), 'Generalized Vickrey Auctions. Working paper, University of Michigan'.

[10] Updates are not causally ordered (Levy & Tempero 1991).

Malone, T. W., Fikes, R. E., Grant, K. R. & Howard, M. T. (1988), Enterprise: A Market-like Task Scheduler for Distributed Computing Environments, *in* H. B.A, ed., 'The Ecology of Computation', Elsevier Science Publishers (North-Holland), pp. 177–205.

Milgrom, P. & Weber, R. (1982), 'A Theory of Auctions and Competitive Bidding', *Econometrica* **50**(5), 1089–1122.

Miller, M. S. & Drexler, K. E. (1988), Markets and Computation: Agoric Open Systems, *in* H. B.A, ed., 'The Ecology of Computation', Elsevier Science Publishers (North-Holland), pp. 133–176.

Parkes, D. C. (1999), iBundle: An efficient ascending price bundle auction, *in* 'Proceedings of the 1st ACM Conference on Electronic Commerce, EC-99', pp. 148–157.

Parkes, D. C. (2001), An Iterative Generalized Vickrey Auction: Strategy-Proofness without Complete Revelation, *in* 'Proceedings of the AAAI Spring Symposium on Game Theoretic and Decision Theoretic Agents', Stanford University, CA.

Parkes, D. C., Kalagnanam, J. & Eso, M. (2001), Achieving Budget-Balance with Vickrey-Based Payment Schemes in Exchanges, *in* 'Proceedings of the International Joint Conference on Artificial Intelligence', pp. 1161–1168.
*citeseer.nj.nec.com/parkes01achieving.html

Rasmusen, E. (1994), *Games and Information — An Introduction to Game Theory*, 2nd edn, Blackwell Publishers, Oxford.

Reed, D., Pratt, I., Menage, P., Early, S. & Stratford, N. (1999), Xenoservers: Accounted Execution of Untrusted Code, *in* 'Proceedings of the 7th Workshop on Hot Topics in Operating Systems (HotOS-VII)', IEEE Computer Society and IEEE Technical Committee on Operating Systems (TCOS), Rio Rico Resort and Country Club, Rio Rico, Arizona.

Rothkopf, M. H., Pekeč, A. & Harstad, R. M. (1995), Computationally Manageable Combinatorial Auctions, Technical Report 95-09, DIMACS, Center for Discrete Mathematics and Theoretical Computer Science, Rutgers, New Jersey, USA.
*citeseer.nj.nec.com/rothkopf98computationally.html

Sandholm, T. W. (1996*a*), Limitations of the Vickrey Auction in Computational Multiagent Systems , *in* 'Proceedings of the Second International Conference on Multiagent Systems (ICMAS-96)', Keihanna Plaza, Kyoto, Japan, pp. 299–306.

Sandholm, T. W. (1996*b*), Negotiation Among Self-Interested Computationally Limited Agents, PhD thesis, Department of Computer Science, University of Massachusetts.

Smith, R. G. (1980), 'The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver', *IEEE Transactions on Computers* **29**(12), 1104–1113.

Stratford, N. & Mortier, R. (1999), An Economic Approach to Adaptive Resource Management, *in* 'Proceedings of the 7th Workshop on Hot Topics in Operating Systems', IEEE Computer Society and IEEE Technical Committee on Operating Systems (TCOS), Rio Rico Resort and Country Club, Rio Rico, Arizona.

Suri, N., Bradshaw, J., Breedy, M., Groth, P., Hill, G., Jeffers, R. & Mitrovich, T. (2000), An Overview of the NOMADS Mobile Agent System, *in* 'Proceedings of ECOOP'2000', Nice, France.

Sutherland, L. E. (1968), 'A Futures Market in Computer Time', *Communications of the ACM* **11**(6), 449–451.

Thain, D., Tannenbaum, T. & Livny, M. (2004), 'Distributed computing in practice: The condor experience', *Concurrency and Computation: Practice and Experience* .

van Steen, M., Homburg, P. & Tanenbaum, A. S. (1999), 'Globe: A Wide-Area Distributed System', *IEEE Concurrency* **7**(1), 70–78.

Vickrey, W. (1961), 'Counterspeculation, Auctions, and Competitive Sealed Tenders', *The Journal of Finance* **16**(1), 8–37.

Waldspurger, C. A., Hogg, T., Huberman, B. A., Kephart, J. O. & Stornetta, W. S. (1992), 'Spawn: A Distributed Computational Economy', *IEEE Transactions on Software Engineering* **18**(2), 103–117.

Wellman, M. P. (1993), 'A Market-Oriented Programming Environment and its Application to Distributed Multicommodity Flow Problems', *Journal of Artificial Intelligence Research* **1**(1), 1–23.
*citeseer.nj.nec.com/wellman93marketoriented.html

Yan, J., Early, S. & Anderson, R. (2000), The XenoService – A Distributed Defeat for Distributed Denial of Service, Technical report, Computing Laboratory, Cambridge, UK.