Improving Resource Utilisation in Market Oriented Grid Management and Scheduling

Kris Bubendorfer

School of Mathematics, Statistics and Computer Science
Victoria University of Wellington,
PO Box 600, Wellington 6001, New Zealand,
Ph: +64 4 463 6484, Fax: +64 4 463 5045
Email: kris@mcs.vuw.ac.nz

Abstract

Service providers of the future could dynamically negotiate for, and create their infrastructure on Grid based utility computing and communication providers. Such commercialisation of large scale gridsystems requires the provision of mechanisms to share the wide pool of Grid brokered resources such as computers, software, licences and peripherals amongst many users and organisations. Quickly and efficiently servicing resource requests is critical to the efficiency of such Grid based utility computing and communication providers. However, distributed resource negotiation is itself a contributor to lower system utilisation, as the negotiation process introduces latency and reservation uncertainty in the system. The CORA architecture is a market based resource negotiation system that utilises a Vickrey auction to make allocations of resource requests to resource providers. The architecture utilises a novel combination techniques to improve utilisation, including oversubscription, coallocation, just-in-time reallocation and a novel flexible contract structure. This paper introduces two significant improvements to the CORA architecture. Firstly, redundant contracts are generated to resolve the problem of post bid unavailability of bidders. Secondly, this paper utilises a new auction architecture that does not require the auctioneer to be trusted. The advantage is that any entity (untrusted or otherwise) can conduct a verifiable and privacy preserving Vickrey auction, removing the need for a trusted and privileged auctioneer within the system.

Keywords: resource reservation, virtual organisations, utility computing.

1 Introduction

In large scale Grid systems efficient negotiation for and allocation of resources is playing an increasingly important role in the performance of the system. Commercialisation of large scale grid systems requires the provision of mechanisms to share the wide pool of Grid brokered resources such as computers, software, and peripherals amongst many users and organisations. The Application Service Provider model (Graupner, Kotov, Andrzejak & Trinks 2003, Dimitrakos, Randal, Yuan, Gaeta, Laria, Ritrovato, Serhan, Wesner & Wulf 2003) is one such Grid commercialisation model.

The Internet Virtual Organisation (iVO) (Foster & Kesselman 1999), can be extended to utilise, on demand, resources leased dynamically from Utility Computing Providers (UCP) (Komisarczuk, Bubendorfer & Chard 2004, Eerola, Konya, Smirnova, Ekelof, Ellert, Hansen, Neilsen, Waananen, Konstantantinov & Ould-Saada 2003). A further extension of the UCP model to include the leasing of communications services, giving Utility Computing Communications Providers (UC^2P). The UC^2P infrastructure could be heavily based on developments in Grid computing.

The UC^2P infrastructure needs to encompass more resources and provide greater flexibility in terms of mobility, resource allocation and economy based resource allocation than the current Globus (Foster, Kesselman & Tuecke 2001) Grid implementations. The Open Grid Service Architecture (OGSA) has come some way to providing some of the needed extensions though the Open Grid Service Infrastructure, Web Service Resource Framework (OGSI, WSRF). However the communication services model requires extension in order to encompass the potential services that could be required in a general purpose UC^2P scenario, and its resource allocation does not provide for speedy resource allocation and optimal charging mechanisms, which are required in iVO operations.

Quickly and efficiently servicing resource requests is critical to the efficiency of a Grid based UC^2P system. However, this is itself a source of low system performance in large-scale distributed systems as the negotiation process introduces latency and reservation uncertainty in the system. Applications request the resources they require, yet by the time the application actually uses the resources, considerable time may have elapsed. In effect, the host must tentatively reserve the resources that are under negotiation for the entire negotiation process, even if the negotiation later terminates without agreement. This has the direct effect of reducing the utilisation of resources within the entire system. In a simulation of a small distributed system utilising a Vickrey auction for resource allocation, utilisation was usually less than 30% (Antliff 2003). The simulation should model UC^2P system well, as it utilised a fine grained resource model and job lengths drawn from a Poisson distribution with a mean lifetime of 10 seconds. The poor utilisation demonstrated by the simulation would be an unacceptable return on computing resources in a commercialised Grid system.

This is not a problem within the traditional Grid model, of large long term computations, where resources are acquired in advance. However, when generalising the Grid model as a basis for the deployment of virtual organisations operating within computing utilities, smaller, more dynamic negotiations that would support mobile devices and the provision of on demand services must be considered. In this

context the utilisation and negotiation latency of resources will become a significant performance bottleneck.

The major goal of the CORA architecture is to address the multiplicative decrease in utilisation within an auction based resource allocation architecture. CORA was developed within the Nomad (Bubendorfer 2001) middleware system. Nomad is a mobile agent system, that utilises a economic management model as a basis for an open system. Many of the lessons learnt developing the Nomad system are applicable to the wider Grid community. Like the Grid, Nomad is a distributed computational system, which consists of a collection of loosely coupled cooperating machines that are capable of hosting distributed applications. Many of the broad aims of Nomad are also shared by the CONOISE (Norman, Preece, Chalmers, Jennings, Luck, Dang, Nguyen, Deora, Shao, Gray & Fiddian 2003) project, however our approach and focus differ.

This paper specifically extends the coallocation and oversubscription resource allocation (CORA) Architecture, that was first presented in (Bubendorfer, Komisarczuk, Chard & Desai 2005) by utilising backstop resource providers in the auction mechanism to compensate for last minute unavailability, and introduces a new auction architecture that does not require the auctioneer to be trusted. This removal of trust is based on garbled circuits (Naor, Pinkas & Sumner 1999) that permit any entity (untrusted or otherwise) to conduct a verifiable and privacy preserving Vickrey auction.

2 CORA

Within large-scale, distributed platforms the efficient allocation of resources plays an critical role in the performance of the system. However, distributed resource allocations can also result in low resource utilisation owing to the delay involved in negotiation, the delay in taking up the agreed resources, and the tentative allocation of resources during the negotiation process. Resource oversubscription allows for better utilisation of resources in distributed systems, however, this must be done in a controlled way to ensure that the resulting allocations can be fulfilled. In the CORA (Coallocation, Oversubscribing Resource Allocation) architecture, resource providers (hosts) delegate all or part of their resources to a selected broking agent(s) that then negotiates on their behalf.

2.1 Economic Resource Management

With large-scale, distributed platforms such as Globus Grids (Foster & Kesselman 1997) , and Planet-Lab (Peterson, Anderson, Culler & Roscoe 2002), it is important to improve the performance of the system as a whole. For example, when hosts are solely responsible for managing and allocating their own resources, it results in efficient local resource allocations. However, it will not necessarily result in globally efficient resource allocations over all hosts.

For many excellent reasons auctions are touted as an efficient solution to the problem of distributed resource allocation in both economic (Bubendorfer 2001, Buyya, Abramson, Giddy & Stockinger 2002, Chien, Chang & Soo 2005) and noneconomic (Malone, Fikes, Grant & Howard 1988) resource allocation systems. There are four main types of auction protocol; the English, Dutch, Sealed-Bid, and what has since become known as the Vickrey auction protocol. The English auction is the conventional open outcry, ascending price, multiple bid protocol. The Dutch auction is an open outcry, descending price, single bid protocol. The Sealed-Bid, or tender, is a sealed 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 bidder pays amount of the second best bid[1]. All four auction protocols yield the same return in private value auctions[2], hence selection of an auction protocol usually depends on it's structure.

However, while on one hand auction protocols are an ideal mechanism for determining the optimal allocation of resources, and for determining the market price of a good, auctions compound the problem of resource utilisation. In particular an auction generally has a single winner, and multiple m losers. While the winner gains the eventual contract[3], there is no such compensation for the m losers of the auction process, and any resources r put aside during the auction will decrease the net utilisation of the system by mr. In addition, the length of time an English auction is unbounded, Dutch auctions are bounded by the bid decrement rate, while sealed bid auctions are of fixed duration.

The duration of an auction protocol limits the application of such resource allocation systems to larger longer lived entities within the system. This is reasonable considering the inherent cost of remote execution, shorter lived resource demands must remain the responsibility of the local host and scheduler.

2.2 Auction Based Resource Negotiation

Many systems utilise auction protocols for resource negotiation, without regard of the inherent shortcomings of the chosen protocol. For reasons including, low messaging overhead, efficiency of allocations and lack of counterspeculation the Vickrey auction protocol has long been a favourite for use in computational economies. However, the Vickrey and all other auction protocols, have known problems that limit the applicability of the protocols in practice. An exhaustive analysis of these protocol considerations is detailed in (Sandholm 1996), however, it is worth detailing a few examples as follow: both the English and Vickrey auctions suffer from self enforced bidder collusion; the Dutch, sealed bid and Vickrey auction all return less revenue than the open exit version of the English auction; all auctions reveal some information, except perhaps the sealed bid auction[4]; and the Vickrey auction alone suffers from the lying auctioneer, as in all other protocols the winner pays the value of their bid.

Any of the four auction protocols outlined in section 2.1 can be implemented with the CORA architecture. Overall, the advantages of the Vickrey auction in automated resource negotiations outweigh its specific disadvantages. A number of careful design choices in the implementation avoid the majority of the limitations of the protocol with only a small reduction in flexibility and optimality. In particular, all auctions are private value auctions (no reselling of contracts) (Bubendorfer 2001). The problem of a lying auctioneer is addressed in section 2.3.

[1] This mechanism results in a dominant strategy of truthful bidding, that is, bidding your true value will always give the best return regardless of other bidders strategies.

[2] The revenue equivalence theorem (Vickrey 1961).

[3] The result of a resource negotiation is a contract.

[4] The Dutch auction reveals the winner and their bid, the Vickrey auction will reveal the winner and the valuation of the second bid (but not the bidder) while the English auction will usual reveal the valuations of all bidders (except the winner). A compromised auctioneer may reveal all the bid values in the case of both sealed bid auctions.

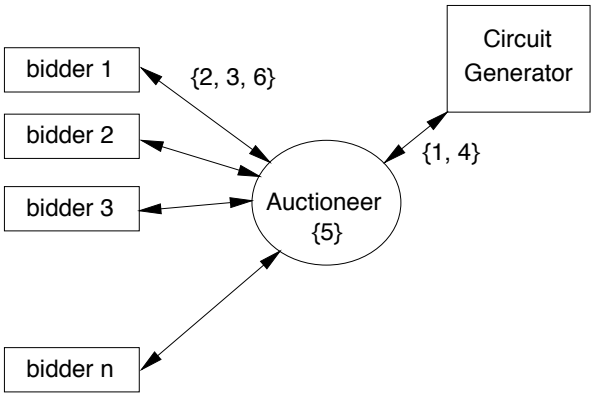

Figure 1: Overview of the garbled circuit protocol.

2.3 Trust and Accountability

When resources under negotiation are spread across multiple administrative domains, the allocations of the resource allocator (in our case auctioneer) should be verifiable, yet disclose as little private valuation information as possible - especially in a competitive economic environment. Thus the role of the auctioneer must be carefully scrutinised.

In the Vickrey auction for instance, a compromised auctioneer can undetectably issue false bids to inflate the value of the second bid, likewise, the values of past bids can be collected and either used in future auctions, or passed on to colluding bidders – *"Even if current information can be safeguarded, records of past behaviour can be extremely valuable, since historical data can be used to estimate willingness to pay."* (Varian 1995)[5].

Until recently there was little recourse but to design auction based allocation systems with an auctioneer as a trusted service. However, this approach tends to centralised designs and lacks openness, transparency and the verifiability. The ideal would be to allow any entity to run an auction, yet for all participants in that auction to maintain their privacy yet allow the outcome of an auction to be verified.

There has been considerable effort in the e-commerce community to remove or at least reduce the amount of trust vested in the auctioneer, utilising a number of cryptographic techniques including, homomorphic encryption (Cachin 1999, Yokoo & Suzuki 2002), shared polynomials (Harkavy, Tygar & Kikuchi 1998), and garbled circuits (Naor et al. 1999). The most appropriate technique for use in CORA is garbled circuits, as garbled circuits appear to be reasonably efficient (the garbled circuits themselves can be constructed offline[6]), does not require multiple distributed auctioneers, continues to preserve privacy after the auction, and the resulting allocations can be verified by any auction participant. Importantly the protocol preserves the communication pattern of the original Nomad auctioneer. Figure 1 provides a high level outline of the protocol from (Naor et al. 1999).

The process is as follows:

1. The auctioneer obtains the garbled program (can be done in advance as here, or in step 4.
2. The auctioneer publishes the details of the auction in a catalogue, this includes the closing time, circuit generator and the requirements of the auction initiator.
3. Bidders submit their encrypted bids to the auctioneer (the circuit generator can only partially decrypt the bid).
4. The auctioneer passes portions of the bids to the circuit generator which partially decrypts them and computes the garbled inputs to the circuit. The inputs are returned to the auctioneer along with a signed translation table that *decrypts* the output of the circuit.
5. The auctioneer uses the garbled inputs and the circuit to compute the output of the circuit.
6. The auctioneer publishes the result with the signed circuit generator's translation table.

The *only* information revealed at the end of the auction to the auctioneer and participants is the winner and the second best bid value. All other values remain secret, thus minimising the trust that bidders must place in the auctioneer. During the auction the auctioneer knows the bidders, but not their bid values, whereas the circuit generator is aware of the bid values but not the bidders who generated them. This is identical to the human security mechanism of requiring two keys held by separate individuals to access a secure vault. The nice result is that neither the auctioneer or circuit generator individually need to be trusted, it is sufficient to ensure that they are independent and are not involved in a collusive agreement. It is much easier to construct a well known untrusted service, than a trusted one[7].

Replay attacks (replaying a bid from a past auction in a current auction) are prevented by utilising a nonce associated with each auction. Bidders can ensure that their bids have been considered in the outcome of the auction by checking a list of hashed bid values that have been signed by the circuit generator. This prevents a compromised auctioneer from simply dropping inconvenient bids. Likewise, bidders can verify that the auctioneer computed the generated circuit by utilising the signed translation table provided by the circuit generator. The auctioneer can also verify that the circuit generator is not corrupt with the *cut-and-choose* technique.

2.4 CORA utilisation enhancing techniques

The techniques identified and adopted within CORA as significant steps towards solving the problem of *multiplicative decrease in utilisation in auction based allocation systems* are *coallocation* (Foster et al. 2001) and *oversubscription* (Fu, Chase, Chun, Schwab & Vahdat 2003). The techniques developed within CORA to move further towards the goal are *just-in-time allocations* and a *progressive contract structure*. All of these techniques require some additional entities within the system, with a greater resource horizon[8] than individual hosts, yet with a smaller scope than say, a system scheduler. To increase the allocation horizon CORA utilises broking agents, to which hosts delegate responsibly for resource negotiation.

[5]It is worth mentioning that due to the open nature of the English auction it suffers from the revelation of valuation information even without a compromised auctioneer, and past bids can be used to adjust reserve values and so on. It is not possible to hide bid values as it is these values that other bidders respond to. If the values are encrypted, the English auction logically degenerates into a single sealed bid auction

[6]The circuit is constructed independently from the bidders and auctioneer – the encrypted bids are only needed by the circuit generator to construct the translation table.

[7]bidders can satisfy themselves when the auction is published prior to bidding that the nominated circuit generator is a satisfactory choice (recorded in their list of well known circuit generators).

[8]It is very difficult to achieve oversubscription and coallocation within the resource scope of an individual machine. A greater view of the resource allocation landscape is called for.

The broking agents then interact with an auctioneer (equivalent to a Globus GARA) that manages resource allocations over administrative boundaries. In a little more detail:

- **Coallocation:** Resource allocation often requires making allocations in a coordinated fashion across virtual organisation boundaries. This form of allocation is known within the Grid community as coallocation. CORA introduced the broking agent role into the Nomad architecture, where the broking agent can act for a group of resource providers and allocate resources based on evaluation of allocation constraints over an ad-hoc resource group.
- **Oversubscription:** Controlled oversubscription of resources improves resource efficiency and availability when rights to allocated resources can be lost or left idle. CORA introduced an oversubscription mechanism by distinguishing between the granting of soft-state and hard-state resource rights to applications.
- **Flexible Contract Structures:** A progression from soft to hard state contracts as the system becomes more certain about the set of resources being allocated. That is, contracts harden as they progress through the various stages of negotiation.
- **Just in Time Allocation:** Introducing the caching of availability knowledge for an ad-hoc group of resource providers allows the Just in Time allocation of resource allocation contracts to resource providers thereby reducing the latency that is inherent in the auctioning process.

2.5 The CORA Architecture

The CORA Architectural components are the Broking Agents, the Reputation Service, and the Agent Finder, as shown in Figure 2. The Nomad components are included for completeness, but are not part of the CORA system per se. A depot is simply a resource provider, that hosts application components, a federation is a collection of depots defined by ownership or administrative domain, and the marketplace is a inter-federation market for resources. The agent pool is a collection of broking agents whose membership is defined by registration within the agent finder. The agent pool does not imply anything about the ownership, administration or physical location of the broking agents.

2.6 Self enforced bidder collusion via delegation

All auction protocols are susceptible to bidder collusion, although, only the English auction protocol and the Vickrey auction protocol self-enforce any collusive agreement (Sandholm 1996). However, with the delegation of resource negotiation from resource providers to mid-level resource brokers, a new form of bidder collusion becomes inevitable when using the Vickrey auction protocol. We have named this problem **self enforced bidder collusion via delegation**.

Consider a system in which there are broking agents. Each agent represents a number of resource providers that have delegated the allocation of their resources. In an auction it makes sense for the broking agent to determine the bids for all the providers that it represents and then only forward the best two bids[9].

[9]The other bids from this agent will not alter the outcome of the auction

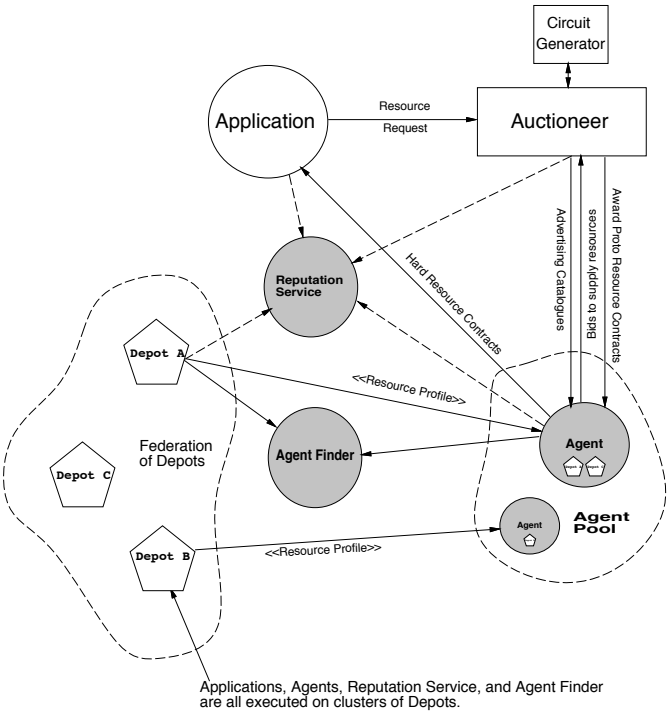

Figure 2: CORA architecture within Nomad.

The obvious ploy for the agent to increase its revenue, is to fabricate the second bid. In situations where only one agent bids on one item — price paid is that of the falsified second bid.

This changes the the dominant strategy of truthful bidding in Vickrey auctions to: *make the first bid truthfully, and make the second bid as favourable as possible*. To make matters worse, in a coallocative auction, the agent can now involve itself in speculating how many of its bids are likely to succeed, and then falsify the remaining coallocative valuations. This has *reintroduced* counterspeculation in coallocative auctions.

There are two possible partial solutions; only allow one bid per agent (reducing the number of bids available in coallocation situations), or ignore multiple bids from the winner when determining the second price (only applies when there is only one bidding agent). Neither of these is a complete solution to this problem, as both somewhat reduce the optimality of the auction process. This problem is an open research question.

2.7 Progressive Contracts

CORA introduced the notion of soft resource contracts (PRC) and hard resource contracts (HRC). A PRC represents soft-state resource rights and is generated by the auctioneer, and returned to the requesting application *and* the winning Agent as the initial result of a negotiation. Being soft-state, a PRC does not guarantee that resources are available, but rather that they may be available upon redemption. For this level of guarantee the broking agent must first *harden* the PRC into a HRC, after considering the current resource situation. The key idea is that an Agent assigns resources from its pool of depots to satisfy a given PRC. Agent generates an HRC if and only if it is able to find a depot for the resources in the PRC. The resources on which the original bids were based may no longer be available, or a better choice may have since become available. In these cases, the depot listed as providing the resources in the PRC may be substituted by another depot in the HRC.

This two level approach is inherently sensible, as a top level allocation entity such as an auctioneer, can not and should not attempt to provide resource guarantees when considering the inherent latency in negotiation. Such approaches would not scale. The primary advantage of the PRC stage is that the negotiation can be cheaply aborted at this early stage if the available resources within the system suddenly change.

Consider the situation if HRC contracts were issued by the marketplace instead of PRC. To prevent rejections of contracts on redemption at the depot, more resources would have to be reserved (by both winning and losing bidders), decreasing overall utilisation. If on the other-hand, a broker indulged in the same degree of oversubscription — then more contracts would be unsatisfiable on redemption, causing more serious and immediate difficulties to the applications.

2.8 Hardening Contracts

The progressive resource contracts work on the principle of hardening. That is, The soft contract generated during the auction process is a placeholder, and only that through the contract commit mechanism does the contract harden into an actual promise of resources. This hardening of the contract takes place in a two-phase commit mechanism. The use of this mechanism ensures that an application is not faced with a situation in which, the contracts presented by it are refused by a depot(s) unless in exceptional circumstances. The conjecture here is that, it is in an application's best interest to re-initiate an auction rather than faced with a partial rejection of contracts at redemption time.

One of the positive side effects of this two phase contract mechanism, is that it neatly caters for both single and coallocative negotiations.

Figure 3 shows the first phase of the commit mechanism, section 2.9 details the second phase of the mechanism.

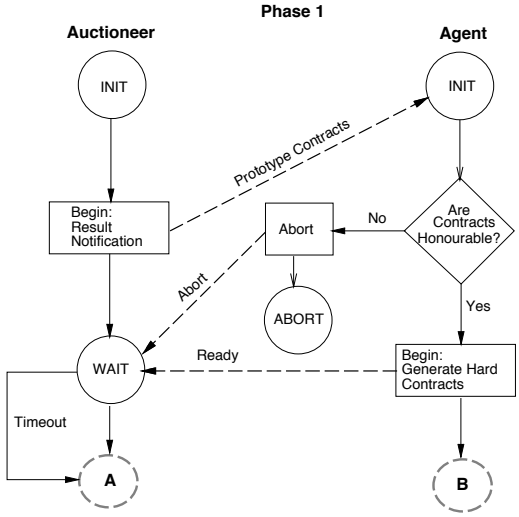

Figure 3: Phase 1 of the contract mechanism.

2.9 Coallocation

Coallocation is a technique of simultaneously allocating resources in predetermined capacities over an ad-hoc group of resource providers. This technique is widely used in Grid computing paradigm and several recent research efforts have taken various approaches to solve this (Anand, Yoginath, von Laszewski, Alunkal & Sun 2003, Czajkowski, Foster & Kesselman 1999, Azzedin & Maheswaran 2001, Azzedin, Maheswaran & Arnason 2004). Coallocation is highly desirable for many applications that demand adequate QoS and parallelism such as content distribution in multimedia and scientific applications.

In CORA, Agents can allocate resources over an ad-hoc group of depots for applications requiring coallocation services. In order to distinguish ordinary allocation requests from coallocation requests, the *count* parameter from RSL (Czajkowski, Foster, Karonis, Kesselman, Martin, Smith & Tuecke 1998)[10] is used. A coallocative negotiation is treated in the same way as a conventional single auction, except that multiple PRCs are generated.

In the original implementation published in (Bubendorfer et al. 2005), only *count* PRCs were generated reflecting the best *count* bids received by the auctioneer. The problem with this approach is that all of the bidders must be in a position to commit to the hardening of the contract. The implication of this is that each coallocative auction is *count* times more likely to fail due to the withdrawal of a bidder. The net result is the waste of considerable resources, specifically those used for; initiating the auction, distributing catalogues, evaluating bids, determining the winner, distributing PRCs, waiting on the commit phase, and the reservation of the resources on each of the bidders.

To address this problem, a new second phase mechanism has been designed that extends the principle of Just-in-Time resource allocations to the coallocative two phase contract mechanism.

Rather than simply generating *count* PRCs, the market generates additional PRCs as *backstops* to include more than *count* bidders and includes them in the hardening of the contract. Figure 4 shows the improved second phase mechanism that utilises the backstop bidders.

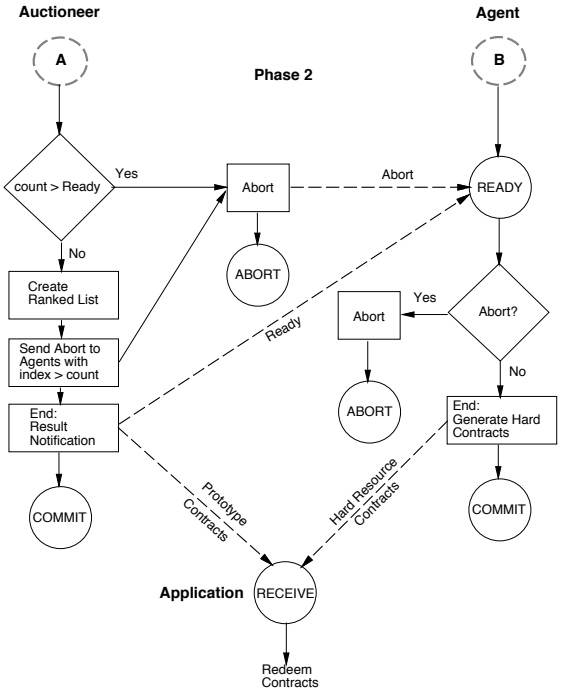

Figure 4: Modified Phase 2 to show the enhanced coallocative mechanism.

How many additional PRCs are generated is a somewhat tuneable heuristic. The current approach

[10] Created for coallocation in computational grids (Czajkowski et al. 1999).

is to create a list of additional candidates, comprised of those with bid values *less* than the average of the bid values for the best *count* bids plus Δ. To limit the number of PRCs generated the list is then truncated using a stepwise function (if *count* is less than 10, the length is 2, if *count* is greater than 10 then the list is truncated at 20% of *count*. All of these values including Δ are subject to tuning in a production system.

Providing the number of *ready* messages is not less than *count* the algorithm is now in a position to harden the contracts and commit the coallocative negotiation. The *count* best PRCs that have signalled *ready* are now sent. This extension to CORA coallocation also allows shorter timeouts, further reducing overall negotiation latency.

2.10 Just-in-time Allocation of Resources

Depots periodically communicate their resource profiles to the broking agents, which then allocate resources based on those profiles. This is effectively caching of availability knowledge for an ad-hoc group of depots, and allows an Agent to make allocation decisions just before sending the actual contracts to application. That is, in the step between PRC and HRC contracts. This technique of making allocation decisions at the last moment before hardening of contracts is effectively just-in-time allocation.

Consider the situation in which the PRC received by a broking agent is for depot A. However, during the interval between when the bids were generated and the time at which the PRCs where generated by the auctioneer, depot A's resource availability changes. This could reflect a change in the set of resources delegated to the broking agent, be a result of oversubscription, or failure. When the broking agent receives the PRC, depot A is no longer able to provide the resources. If the broking agent can still satisfy the contract utilising resources from another depot, then it may substitute the depot for another when hardening the PRC into a HRC. This overcomes many of the problems introduced by the latency in negotiation.

The significant latency is introduced by the time delay in receiving all bids from all Agents, carrying out auction process, and notifying winning Agents about results. During this time, depot's situation may change and it may no longer be able to provide promised resources. Therefore, it is essential to minimise this latency as much as possible. Just-in-time allocation helps in reducing this latency inherent in an auction process.

2.11 Oversubscription

CORA broking agents use the controlled oversubscription of resources to improve the resource utilisation depots. broking agents use the same resources in multiple bids to increase the chance of winning an auction, relying on the low probability of winning all such auctions.

This does not alter the valuation of the bids, but raises the spectre of contracts being rejected through a lack of resources. Obviously the degree of oversubscription and the probability of winning an auction have a direct bearing on the likelihood of rejection. These factors are a combination of agent policy and market environment. As discussed in Section 2.8, the resources in a PRC issued by the auctioneer are not guaranteed until the second phase of the two-phase commit mechanism, when the agent hardens the contract. In the worst case, if the agent can't find sufficient resources, then application will have to initiate a new resource auction.

2.12 Project Status

The CORA architecture is implemented and extends the Nomad prototype. While it has been tested for functionality, many of the policy components are not fully implemented. That is, the intelligence of the various entities, such as the brokers, is limited to simple behaviours that satisfied testing - but would not be useful in a functioning system.

In the future the aim is to add more scheduling and bidding intelligence to the broking agents. A simulation is also planned to measure the efficacy of the CORA architecture - however we still need to understand more about the behaviour of the brokers and the requirements of a UC^2P system before such a simulation can produce meaningful results.

3 Conclusions

This paper extends the original CORA allocation architecture in two major ways. Firstly, additional soft contracts are issued to act as backstop resource providers and thus compensate for the problem of post bid unavailability of the preferred resource providers. This new mechanism is integrated seamlessly into the second phase of the contract hardening protocol, extending the concept of just-in-time allocation. Secondly, this paper utilises a new auction architecture that does not require the auctioneer to be trusted. The major implication of this change is that the auctioneer no longer needs to be a privileged system component, but rather any entity (untrusted or otherwise) can conduct a verifiable and privacy preserving Vickrey auction. The benefits include: easing system design, provision of infrastructure and increasing confidence through verifiability, which outweigh the cost of providing the circuit generator.

References

Anand, S., Yoginath, S., von Laszewski, G., Alunkal, B. & Sun, X.-H. (2003), Flow-based Multistage Co-allocation Service, *in* 'The 2003 International Conference on Communications in Computing', Las Vegas, Nevada, USA.

Antliff, S. (2003), 'Experimental Verification of Collective Vickery Auctions in Nomad', Honours Report, Victoria University of Wellington, 2003.

Azzedin, F. & Maheswaran, M. (2001), A Co-allocation Mechanism for Multimedia Enabled Grids, *in* 'Proceedings of 13th IASTED International Conference on Parallel and Distributed Computing Systems (PDCS '01)', pp. 27–32.

Azzedin, F., Maheswaran, M. & Arnason, N. (2004), 'A Synchronous Co-Allocation Mechanism for Grid Computing Systems', *Cluster Computing* **7**(1), 39–49.

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

Bubendorfer, K., Komisarczuk, P., Chard, K. & Desai, A. (2005), Fine Grained Resource Reservation and Management in Grid Economies, *in* 'Proceedings of the 2005 International Conference on Grid Computing and Applications', Las Vegas, Nevada, USA., pp. 31–38.

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.

Cachin, C. (1999), Efficient Private Bidding and Auctions with an Oblivious Third Party, *in* 'The ACM Conference on Computer and Communications Security', pp. 120–127.

Chien, C.-H., Chang, P. H.-M. & Soo, V.-W. (2005), Market-Oriented Multiple Resource Scheduling in Grid Computing Environments, *in* 'Proceedings of Advanced Information Networking and Applications (AINA'05)', Vol. 1, Taipei,, pp. 867–872.

Czajkowski, K., Foster, I. & Kesselman, C. (1999), Resource Co-Allocation in Computational Grids, *in* 'Proceedings of the 8th IEEE International Symposium on High Performance Distributed Computing (HPDC-8)', pp. 219–228.

Czajkowski, K., Foster, I. T., Karonis, N. T., Kesselman, C., Martin, S., Smith, W. & Tuecke, S. (1998), A Resource Management Architecture for Metacomputing Systems, *in* 'Proceedings of the Workshop on Job Scheduling Strategies for Parallel Processing', Springer-Verlag, pp. 62–82.

Dimitrakos, T., Randal, D. M., Yuan, F., Gaeta, M., Laria, G., Ritrovato, P., Serhan, B., Wesner, S. & Wulf, K. (2003), An Emerging Architecture Enabling Grid Based Application Service Provision, *in* 'Seventh International Enterprise Distributed Object Computing Conference (EDOC'03)', Brisbane, Queensland, Australia, pp. 240–251.

Eerola, P., Konya, B., Smirnova, O., Ekelof, T., Ellert, M., Hansen, J. R., Neilsen, J. L., Waananen, A., Konstantantinov, A. & Ould-Saada, F. (2003), 'Building a Production Grid in Scandinavia', *IEEE Internet Computing* **7**(4), 27–35.

Foster, I. & Kesselman, C. (1997), 'Globus: A Metacomputing Infrastructure Toolkit', *The International Journal of Supercomputer Applications and High Performance Computing* **11**(2), 115–128.

Foster, I. & Kesselman, C. (1999), *"The Grid: Blueprint for a New Computing Infrastructure"*, Morgan and Kaufmann.

Foster, I., Kesselman, C. & Tuecke, S. (2001), 'The Anatomy of the Grid: Enabling Scalable Virtual Organizations', *Lecture Notes in Computer Science* **2150**.

Fu, Y., Chase, J., Chun, B., Schwab, S. & Vahdat, A. (2003), SHARP: an architecture for secure resource peering, *in* 'Proceedings of the 19th ACM symposium on Operating systems principles', ACM Press, pp. 133–148.

Graupner, S., Kotov, V., Andrzejak, A. & Trinks, H. (2003), 'Service-Centric Globally Distributed Computing', *IEEE Internet Computing* **7**(4), 36–43.

Harkavy, M., Tygar, J. D. & Kikuchi, H. (1998), Electronic Auctions with Private Bids, *in* '3rd USENIX Workshop on Electronic Commerce', pp. 61–74.

Komisarczuk, P., Bubendorfer, K. & Chard, K. (2004), Enabling virtual organisations in mobile networks, *in* 'IEE 3G2004 Conference', London, UK, pp. 123–127.

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.

Naor, M., Pinkas, B. & Sumner, R. (1999), Privacy Preserving Auctions and Mechanism Design, *in* 'the 1st ACM Conference on Electronic Commerce', ACM, pp. 129–139.

Norman, T. J., Preece, A., Chalmers, S., Jennings, N. R., Luck, M., Dang, V. D., Nguyen, T. D., Deora, V., Shao, J., Gray, W. A. & Fiddian, N. J. (2003), Conoise: Agent-based formation of virtual organisations, *in* 'Proceedings of AI2003, the Twentythird SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence', pp. 353–366.

Peterson, L., Anderson, T., Culler, D. & Roscoe, T. (2002), A Blueprint for Introducing Disruptive Technology into the Internet, *in* 'Proceedings of the 1st Workshop on Hot Topics in Networks (HotNets-I)'.

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

Varian, H. R. (1995), Economic Mechanism Design for Computerized Agents, *in* 'Proc. of Usenix Workshop on Electronic Commerce'.

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

Yokoo, M. & Suzuki, K. (2002), Secure Multi-agent Dynamic Programming based on Homomorphic Encryption and its Application to Combinatorial Auctions, *in* 'Proceedings of the first joint International Conference on Autonomous Agents and Multiagent Systems', ACM, Bologna, Italy.