Fine Grained Resource Reservation in Open Grid Economies

Kris Bubendorfer
School of Mathematics, Statistics and Computer Science,
Victoria University of Wellington, New Zealand.
kris@mcs.vuw.ac.nz

Abstract

The CORA (Coallocative, Oversubscribing Resource Allocation) architecture is a market based resource reservation system that utilises a trustworthy Vickrey auction to make combinatorial allocations of resources. This paper provides an overview of several significant components of the CORA architecture. Firstly, CORA utilises a novel combination of techniques to improve utilisation, including oversubscription, coallocation, just-in-time reallocation and a flexible contract structure. Secondly, this paper utilises a new auction architecture that does not require the auctioneers to be trusted. The advantage is that any entity (untrusted or otherwise) can conduct a privacy preserving Vickrey auction, removing the need for a trusted and privileged auction service within the system. CORA demonstrates how a practical, efficient and trustworthy auction scheme can be implemented in a Grid Economy.

1 Introduction

Commercialisation or globalisation of large scale Grids 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. Service providers of the future could dynamically negotiate for, and create their infrastructure on Grid based utility computing and communication providers. Efficient negotiation for and allocation of resources will play an increasingly important role in the performance of these large scale computing systems. The Application Service Provider model [11, 16] is an example of a model for Grid commercialisation.

The Internet Virtual Organisation (iVO) [13], can be extended to utilise, on demand, resources leased dynamically [23] from Utility Computing Providers (UCP) [12, 18]. A further extension of the UCP model to include the leasing of communications services, results in Utility Computing Communications Providers (UC^2P). Such a UC^2P infrastructure needs to support mobility and provide flexibility in terms of granularity and duration during resource allocation. The Open Grid Service Architecture (OGSA) provides some of the needed extensions though the Open Grid Service Infrastructure, Web Service Resource Framework (OGSI, WSRF). However, unlike the traditional Grid model where fixed resources are acquired in advance of execution, a UC^2P infrastructure requires smaller, more dynamic negotiations that support mobile devices and provision for on demand services. A truly global Grid must be *open* and encompass resources from potentially untrustworthy organisations and individuals. In addition to the requirements of confidentiality, access control and integrity, an open Grid will also require a trustworthy resource allocation and management (accounting) service.

This paper provides an overview of CORA (Coallocative, Oversubscribing Resource Allocation), a market based resource reservation system that utilises a trustworthy Vickrey auction to make combinatorial allocations of resources. In particular CORA utilises oversubscription and just-in-time allocations to reduce tentative resource reservations, and a progression from soft to hard contractual agreements as the certainty of the resource allocation increases. The two phase contractual commit process neatly caters for both single and coallocative negotiations. Trust in the outcome of CORA's resource allocations is established through the use of homomorphic encryption and distributed decryption. The major advantage is that the auctioneer no longer needs to be a privileged system component, but rather an ad-hoc group of auctioneers who do not all need to be individually trusted.

2 Economic Resource Management

Auctions are proposed as an efficient solution to the challenge of distributed resource allocation in both economic [5, 8, 9] and noneconomic [21] 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 (1st 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 bid (2nd price). The second price bid mechanism results in a dominant strategy of truthful bidding in private value auctions, that is, bidding your true value will always give the best return regardless of other bidders strategies. A Vickrey auction is shown in Figure 1. In this example, the highest bid ($3) is placed by Sam, who is then awarded the auction at the price of the second best bid ($2) placed by Jim. All four auction protocols yield the same return in private value (when a good is for consumption rather than resale) auctions [29], hence the selection of an auction protocol usually depends on implementation pragmatics such as messaging requirements.

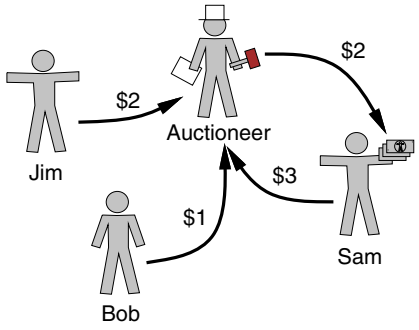

Figure 1. An example of a Vickrey auction.

3 Combinatorial Auctions

When it comes to computational auctions, the problem is that most existing resource auction systems [8] focus on a single representative resource. However, it may not be possible to achieve QoS goals using a single representative good as the basis for resource allocations. Execution resources form an indivisible set, related and conditional upon the availability of each other. Piecewise negotiation for individual resources will not often provide a usable result, let alone optimal allocation. Game theorists term this as the combinatorial allocation problem (CAP), in which a set of components have a synergistic value that exceeds the sum of the individual parts. Because of synergistic combinations and possible substitutions, bidders have preferences not just for particular items, but for collections of items. For example, in Figure 2, Jim wants apples only if he can also have oranges, Bob wants all of the goods but does not need all of them, while Sam needs all of three of the goods. The highest valuation ($4) for the auction is generated by allocating the apples and oranges to Jim and the bananas to Bob while Sam misses out. As there is no 'second-price' in a combinatorial auction the price paid by the winner is their bid less a discount. The discount is calculated by removing the winner from the auction and recomputing the result. The difference between the two values is the winner's discount. This is equivalent to the single resource Vickrey auction. The Generalised Vickrey Auction (GVA) [20] extends the original Vickrey auction protocol to address the CAP. Solving a single GVA auction is *NP*-hard [25], and for this reason there are a number of optimised variations [22, 24], and approximations [17, 19] that reduce the computation time.

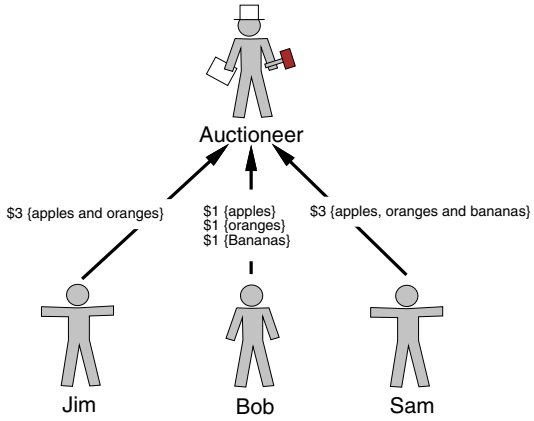

Figure 2. An example of a combinatorial auction.

4 Trust and the Auctioneer

The Vickrey auction protocol has long been a favourite for use in computational economies, for reasons such as, low messaging overhead, efficiency of allocations and lack of counterspeculation [29]. 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 [26], however, it is worth detailing a few examples as follows: both the English and Vickrey auctions suffer from self enforced bidder collusion; all auctions reveal some information, except perhaps the sealed bid auction[1]; and the Vickrey auction alone suffers from the lying auctioneer, as in all other protocols the winner pays the value of their bid.

[1]The Dutch auction reveals the winner and their bid, the Vickrey auction will reveal the winner and the price of the second bid (but not the bidder) while the English auction will reveal the valuations of all bidders (except the winner, who has not yet reached their maximum valuation). A compromised or corrupt auctioneer may reveal all the bid values in the case of both sealed bid auctions.

In the Vickrey auction for instance, a compromised auctioneer can undetectably issue false bids to inflate the value of the second bid or reduce the winner's discount. 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."* [28][2].

5 Trustworthy Auctions in CORA

The trustworthy auction scheme developed for CORA is an extension of the work done by Suzuki and Yokoo in [27, 30]. Their initial scheme [30] was a 1st price combinatorial auction (non Vickrey). This scheme was later extended to a 2nd price combinatorial auction (the Secure Generalised Vickrey Auction (SGVA)) in [27] by computing the discount in a matrix. This implementation is not very efficient, and one of the contributions of the CORA implementation is to adapt the dynamic graph programming technique from Yokoo and Suzuki's 1st price protocol [30] to also compute the discounts and subsequent prices in the CORA implementation. Unfortunately there is insufficient room in this article for a complete description of the Homomorphic SGVA [7, 27, 30] protocol and the following synopsis provides a brief overview of the approach.

In the SGVA scheme multiple auctioneers are used to ensure that a malicious auctioneer cannot subvert the outcome of an auction. Each bid value is represented in a tally-mark vector. Homomorphic encryption (addition can be performed on homomorphically encrypted values) is used to hide the bid values from the auctioneers, but still enable comparison of the encrypted values. This permits the outcome of the auction to be computed without needing the bid values to be fully decrypted. To prevent a malicious auctioneer from continuing to decrypt the vectors we use a distributed decryption protocol that requires multiple key shares to decrypt each vector element. By splitting the key shares amongst a group of auctioneers we ensure that a minority of malicious auctioneers cannot manipulate the outcome of the auction.

The SGVA protocol prevents a malicious auctioneer revealing bids or manipulating the outcome of the auction. Section 7 provides an overview of how SGVA fits within the complete CORA architecture, in particular steps 7 and 8 encompass the SGVA protocol.

In an earlier paper [7] we detailed the operation of the SGVA protocol and provided measurements of the time taken to compute the winners with variation in the number of bidders, the size of the keys and the number of resources. However, in that paper we had *not* yet managed to implement an efficient 2nd price auction. This implementation is shown in Figure 3 as the 1st price homomorphic auction.

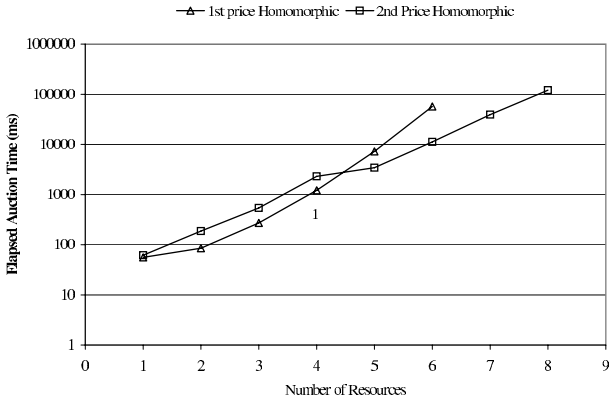

Figure 3. The net elapsed time when computing the *winners* and allocations for the 1st price combinatorial auction and the net elapsed time when computing the *winners*, *allocations* and *prices* in the 2nd price combinatorial auction.

The graph in Figure 3 also shows results for the new 2nd price algorithm in which the discount is computed using dynamic programming. It is worth emphasising that the the time taken in this example is for an auctioneer to compute the winners, allocations *and* prices in auctions consisting of 1 through 8 resources shown on the x-axis.

It is worth pointing out that the single resource auction shown in Figure 3 is in effect a standard Vickrey auction, although in this case a trustworthy one. The time measurements shown on the y-axis include the time required to perform the various encryptions and decryption of the bidders valuations. Recall that each winner [3] must be removed from the auction and the result recomputed. The difference between the revenue in the two computations is the winner's contribution to the overall revenue and is used to compute the discount applied to the winner's valuation. This is obviously an expensive operation, and the new implementation performs this in more-or-less the time it took the original algorithm just to compute the winners.

The cryptographic overhead is swamped by the combinatorial explosion as the number of resources increases. However, the results given in Figure 3 include all combinations of resources and therefore these results should be consid-

[2] 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

[3] A winner is a bidder who is awarded any allocation of resources, so there are multiple winners.

ered as a worst-case. In practice there would be many combinations that are not permissible, rather bundles of permitted combinations are usually defined by the service requesting resources. This reduces the number of combinations significantly. None-the-less, even assuming the worst case, auctions involving 6 or fewer resources are tractable.

6 Utilisation enhancing techniques

All distributed resource allocations can 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. 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 low resource utilisation. In particular an auction generally has a single winner, and multiple m losers. While the winner gains the eventual contract, 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.

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 CORA, applications issue requests for resources to a set of auctioneers, who then solicit bids that satisfy the application's constraints from resource providers. The existing techniques adopted within CORA for solving the problem of *multiplicative decrease in utilisation in auction based allocation systems* are *coallocation* [14] and *oversubscription* [15]. In addition we have developed the new techniques of *just-in-time allocations* and a *progressive contract structure* [4] within CORA.

These techniques require additional entities within the system and these entities need a greater resource horizon than an individual host, yet with less scope than say, a system scheduler. To increase this allocation horizon CORA utilises broking agents, to which hosts delegate responsibly for resource negotiation. 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. In CORA 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.

6.1 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 resource providers to satisfy a given PRC. Agent generates an HRC if and only if it is able to find a resource provider 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 resource provider listed as providing the resources in the PRC may be substituted by another resource provider 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 resource provider, more resources would have to be reserved (by both winning and losing bidders), decreasing overall utilisation. If on the otherhand, a broker indulged in the same degree of oversubscrip-

tion — then more contracts would be unsatisfiable on redemption, causing more serious and immediate difficulties to the applications.

6.2 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 resource provider(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 be 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 4 shows the first phase of the commit mechanism, section 6.3 details the second phase of the mechanism.

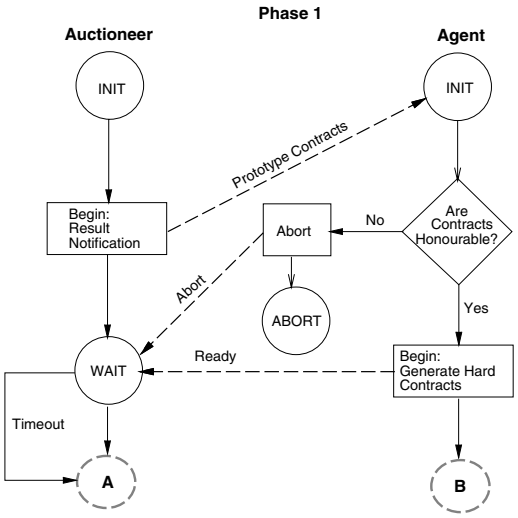

Figure 4. Phase 1 of the contract mechanism.

6.3 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 [1, 2, 3, 10]. 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 resource providers for applications requiring coallocation services. In order to distinguish ordinary allocation requests from coallocation requests, CORA adopts the *count* parameter from [10]. 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 [6], 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 5 shows the improved second phase mechanism that utilises the backstop bidders.

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

6.4 Just-in-time Allocation of Resources

Resource Providers 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 resource

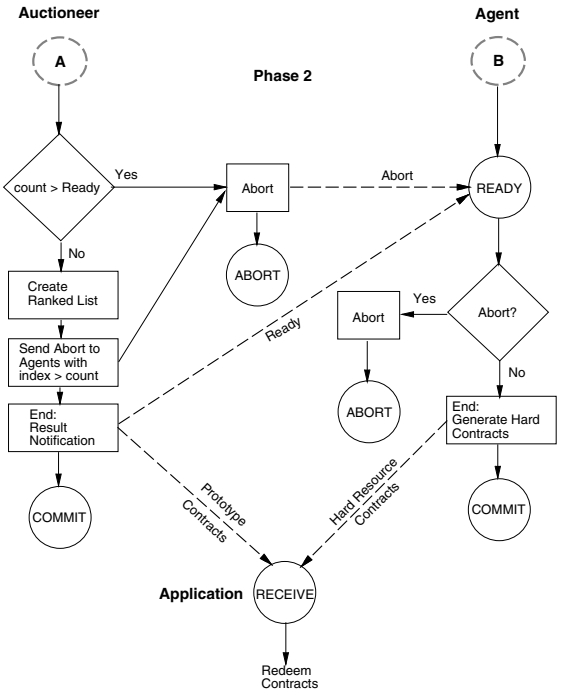

Figure 5. Phase 2, hardening the contract.

providers, 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 resource provider A. However, during the interval between when the bids were generated and the time at which the PRCs where generated by the auctioneer, resource provider 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, resource provider A is no longer able to provide the resources. If the broking agent can still satisfy the contract utilising resources from another resource provider, then it may substitute the resource provider 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, resource provider'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.

6.5 Oversubscription

CORA broking agents use the controlled oversubscription of resources to improve the resource utilisation of resource providers. 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 6.2, 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 the application will have to initiate a new resource auction.

7 CORA Architecture

Now all the components of CORA have been defined, it is time to draw those components together with an overview of the Architecture.

Resource providers (RPs) delegate all or part of their resources to a selected CORA Resource Broker (CRB) that then negotiates on their behalf. The CRB acts within the system to increase the allocation horizon beyond that of a single RP. This broader view is required to achieve oversubscription and coallocation over multiple RPs, and to facilitate the just-in-time allocations and progressive contract structure. The combination of auctioneers and CRBs provides for resource allocations outside administrative boundaries.

Figure 6 illustrates how the SVGA auctioneers have been integrated with the CORA architecture. There are two separate sequences of events, the first relates to the delegation of resource allocation to the CRBs. In step (A), an RP issues a query to the Yellow Pages service to obtain a list of matchings CRBs. In step (B) the RP assesses the CRB via the Reputation service, and finally in step (C), delegates a proportion of its resources to the chosen CRB. The second sequence describes the actual process of an auction. In step (1) an application issues a query to the Yellow Pages service to obtain a list of potential auctioneers, in step (2) the application assesses the auctioneers via the Reputation service, and in step (3) selects three candidates and requests that the auctioneers host the auction. Once a set of three auctioneers have committed to hosting the auction, the application issues a resource request to one of the auctioneers in step (4). The auctioneers in step (5) distribute the advertising catalogues to potential bidders via a clearing house (not

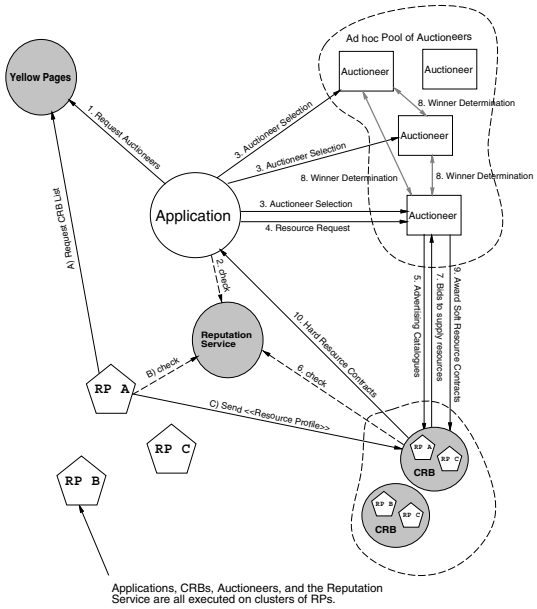

Figure 6. Overview of the CORA Architecture with SGVA Extensions.

shown in the diagram). Bidders who have registered with the clearing house receive the catalogues, and assesses the nominated auctioneers via the reputation service. If a bidder is unhappy with the application's selection of the auctioneers, it need not bid on the auction. In step (7) bidders register their bids with the auctioneers, and in step (8) the auctioneers mutually determine the winners and the subsequent allocations. In step (9) soft contracts are awarded to the successful CRBs and these are then finalised by hardening the contracts in step (10) and transferring them to the application.

8 Acknowledgments

This work exists as part of a larger research effort, and as such includes indirect contributions from a number of people. In particular Ian Welch has co-supervised research students and has co-authored related papers. Students Wayne Thompson and Stefan Krawczyk contributed code for computing the allocations of the SGVA auction.

9 Conclusions

This paper provides an overview of CORA, a market based resource reservation system that utilises a trustworthy Vickrey auction to make combinatorial allocations of resources. The CORA architecture addresses the problem of low resource utilisation due to latency in the distributed negotiation of resources. CORA utilises oversubscription and just-in-time allocations to reduce tentative resource reservations, and a progression from soft to hard contractual agreements as the certainty of the resource allocation increases. The two phase contractual commit process neatly caters for both single and coallocative negotiations. Additional soft contracts are issued to backstop resource providers to compensate for the problem of post bid unavailability of the preferred resource providers. In addition the CORA architecture removes the need for pre-existing trust. Only the winners of the auction and the prices they pay are revealed while all other bid values are kept secret.

The significance of this is that the auctioneer no longer needs to be a privileged system component, but an ad-hoc group of auctioneers who do not need to be individually trusted. This approach provides the potential for a wide distribution of load amongst many auctioneers. The use of both a threshold bid encrypted scheme and reputation based selection of auctioneers is a novel hybrid approach. The auctioneers are mutually selected by the participants in the auction. CORA demonstrates how a practical, efficient and trustworthy auction scheme can be implemented in a Grid Economy.

References

[1] S. Anand, S. Yoginath, G. von Laszewski, B. Alunkal, and X.-H. Sun. Flow-based Multistage Co-allocation Service. In *The 2003 International Conference on Communications in Computing*, Las Vegas, Nevada, USA, June 2003.

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

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

[4] K. Bubendorfer. Improving Resource Utilisation in Market Oriented Grid Management and Scheduling. In *To appear in proceedings of the 4th Australasian Symposium on Grid Computing and e-Research (AusGrid 2006)*, volume 48, Hobart, Tasmania, Australia, January 2006.

[5] K. Bubendorfer and J. H. Hine. Resource Discovery and Negotiation in the NOMAD System. In *in Proceedings of ACSC2005, The Twenty Eigth Australasian Computer Science Conference*, volume 27, Newcastle, NSW, Australia, January 2005.

[6] K. Bubendorfer, P. Komisarczuk, K. Chard, and A. Desai. Fine Grained Resource Reservation and Management in Grid Economies. In *Proceedings of the 2005 International Conference on Grid Computing and Applications*, pages 31–38, Las Vegas, Nevada, USA., June 2005.

[7] K. Bubendorfer, I. Welch, and B. Chard. Trustworthy Auctions for Grid-style Economies. In *In the proceedings of the*

6th IEEE International Symposium on Cluster Computing and the Grid (CCGrid06), volume 1, pages 386–390, Singapore, May 2006. IEEE.

[8] R. Buyya, D. Abramson, J. Giddy, and H. Stockinger. Economic models for resource management and scheduling in Grid computing. *Concurrency and Computation: Practice and Experience*, 14:1507–1542, 2002.

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

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

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

[12] P. Eerola, B. Konya, O. Smirnova, T. Ekelof, M. Ellert, J. R. Hansen, J. L. Neilsen, A. Waananen, A. Konstantantinov, and F. Ould-Saada. Building a Production Grid in Scandinavia. *IEEE Internet Computing*, 7(4):27–35, 2003.

[13] I. Foster and C. Kesselman. *"The Grid: Blueprint for a New Computing Infrastructure"*. Morgan and Kaufmann, 1999.

[14] I. Foster, C. Kesselman, and S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *Lecture Notes in Computer Science*, 2150, 2001.

[15] Y. Fu, J. Chase, B. Chun, S. Schwab, and A. Vahdat. SHARP: an architecture for secure resource peering. In *Proceedings of the 19th ACM symposium on Operating systems principles*, pages 133–148. ACM Press, 2003.

[16] S. Graupner, V. Kotov, A. Andrzejak, and H. Trinks. Service-Centric Globally Distributed Computing. *IEEE Internet Computing*, 7(4):36–43, 2003.

[17] M. M. Halldórsson. Approximations of Weighted Independent Set and Hereditary Subset Problems. In *Computing and Combinatorics, 5th Annual International Conference, COCOON 99, Lecture Notes in Computer Science 1627 Springer*, pages 261–270, 1999.

[18] P. Komisarczuk, K. Bubendorfer, and K. Chard. Enabling virtual organisations in mobile networks. In *IEE 3G2004 Conference*, pages 123–127, London, UK, October 2004.

[19] D. Lehmann, L. I. Oćallaghan, and Y. Shoham. Truth revelation in approximately efficient combinatorial auctions. *Journal of the ACM*, 49(5):577–602, 2002.

[20] J. K. MacKie-Mason and H. R. Varian. Generalized Vickrey Auctions. Working paper, University of Michigan, 1994.

[21] T. W. Malone, R. E. Fikes, K. R. Grant, and M. T. Howard. Enterprise: A Market-like Task Scheduler for Distributed Computing Environments. In H. B.A, editor, *The Ecology of Computation*, pages 177–205. Elsevier Science Publishers (North-Holland), 1988.

[22] N. Nisan and A. Ronen. Computationally feasible VCG mechanisms. In *Proceedings of the 2nd ACM conference on Electronic commerce*, pages 242–252, New York, NY, USA, 2000. ACM Press.

[23] T. J. Norman, A. Preece, S. Chalmers, N. R. Jennings, M. Luck, V. D. Dang, T. D. Nguyen, V. Deora, J. Shao, W. A. Gray, and N. J. Fiddian. Conoise: Agent-based formation of virtual organisations. In *Proceedings of AI2003, the Twentythird SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence*, pages 353–366, 2003.

[24] D. C. Parkes. 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, Mar. 2001.

[25] M. H. Rothkopf, A. Pekeč, and R. M. Harstad. Computationally Manageable Combinatorial Auctions. Technical Report 95-09, DIMACS, Center for Discrete Mathematics and Theoretical Computer Science, Rutgers, New Jersey, USA, Apr. 1995.

[26] T. Sandholm. Limitations of Vickrey Auction in Computational Multiagent Systems. In *In Proceedings of the Second International Conference on Multiagent Systems (ICMAS-96), Kyoto, Japan*, pages 299–306, December 1996.

[27] K. Suzuki and M. Yokoo. Secure generalized vickery auction using homomorphic encryption. In *Financial Cryptography, 7th International Conference, FC 2003*, volume 2742 of *Lecture Notes in Computer Science*. Springer, 2003.

[28] H. R. Varian. Economic Mechanism Design for Computerized Agents. In *Proceedings of Usenix Workshop on Electronic Commerce*, July 1995.

[29] W. Vickrey. Counterspeculation, Auctions, and Competitive Sealed Tenders. *The Journal of Finance*, 16(1):8–37, March 1961.

[30] M. Yokoo and K. Suzuki. 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*, Bologna, Italy, July 2002. ACM.