Resource Allocation in Communication Networks Using Market-Based Agents

Nadim Haque, Nicholas R. Jennings, Luc Moreau
School of Electronics and Computer Science, University of Southampton
Southampton, UK.
{N.A.HAQUE,N.R.JENNINGS,L.MOREAU}@ecs.soton.ac.uk

Abstract

This work describes a system that allocates end-to-end bandwidth, in a switched meshed communications network. The solution makes use of market-based software agents that compete in a number of decentralised marketplaces to buy and sell bandwidth resources. Agents perform a distributed depth first search with decentralised markets in order to allocate routes for calls. The approach relies on a resource reservation and commit mechanism in the network. Initial results show that under a light network load, the system sets up a high percentage of calls which is comparable to the optimum value and that, under all network loads, it performs significantly better than a random strategy.

1 Introduction

The work presented in this paper describes the methodology, implementation and evaluation of a multi-agent system that allocates end-to-end (source-to-destination) bandwidth in a communications network to set up calls. In particular, we consider meshed networks where nodes communicate with their immediate neighbours using radio [1]. In such networks, nodes operate on batteries and solar power and are therefore designed to consume as little power as possible, where they are connected to fixed handsets via base stations. These networks are used mainly in developing third world countries where equipment is scarce and cost is at a minimum. They are equally applicable in areas where the network infrastructure is not fixed, for example, soldiers in a desert who need to communicate their geographical positions to one another. Such low power consumption and low-cost solutions imply that such a network has limited bandwidth. This has two implications: (i) the number of messages sent between nodes must be restricted and (ii) the size of each message sent should be kept to a minimum.

Therefore, it can be seen that resource allocation is a central problem in effectively managing such networks. Specifically, this covers the process by which network elements try to meet the competing demands that applications have for network resources — primarily link bandwidth and buffer space in routers or switches [2]. This is a challenging problem since resources become scarce when there is a high demand for them. Thus, practical methods must be found for allocating the scarce resources that satisfy users adequately.

Against this background, the solution we have developed can be viewed as a computational economy where software agents compete in a marketplace to buy and sell bandwidth on a switched network. Here, buyer agents represent callers that aim to make calls in the network and seller agents represent the owners of the resources who wish to profit from leasing their bandwidth. However, a key requirement is that the bandwidth resources should not all be sold from the same central location in the network because a centralised market server would give a central point of failure. Therefore, we use decentralised market servers from where resources are bought and sold. This means that if a failure was to occur on a server node, then resources should still be available from other market servers. Also, since bandwidth from neighbouring nodes is required to form a continuous end-to-end path in the network, there is a requirement for a protocol that can allocate interrelated resources simultaneously. This ensures that either no resources or a *complete* set of resources are bought.

We decided to base our solution on agents for a number of reasons. First, their autonomous behaviour allows them to carry out their tasks in the decentralised control regime of distributed marketplaces. Second, the reactive nature of agents is needed to respond to requests quickly so that calls within the network can be made with minimum delay. Third, agents have the ability to flexibly interact which is important in our system because the agents need to bid against a variety of different opponents in an environment where the available resources vary dynamically. A market-based approach was chosen for the following reasons. First, markets are effective mechanisms for allocating scarce resources in a decentralised fashion [3]. Second, they achieve this based on the exchange of small amounts of information such as prices. Finally, they provide a natural way of viewing the resource allocation problem because they ensure the individual who values the resources the most will obtain them.

To meet our requirements, the system we have developed extends the state of the art in the following ways. It develops a novel distributed market mechanism scheme in which the allocations made consist of sets of *interrelated* resources, bundles, which are sold in multiple markets. The marketplace protocol incorporates a reservation and commitment mechanism that provides a guarantee that resources will not be bought unnecessarily.

The remainder of this paper is structured as follows: section 2 describes the design of the system and the components that it comprises. A methodology outlining the evaluation of the system and experimental results are presented in section 3. Section 4 describes the related work. Finally, a conclusion of the work is discussed in section 5 along with the envisaged future work.

2 System Design

This section describes the design of the system. Specifically, section 2.1 outlines the basic components, section 2.2 describes the network used and how it is modelled, section 2.3 details the constituent agents and section 2.4 then outlines the process of how resources are acquired.

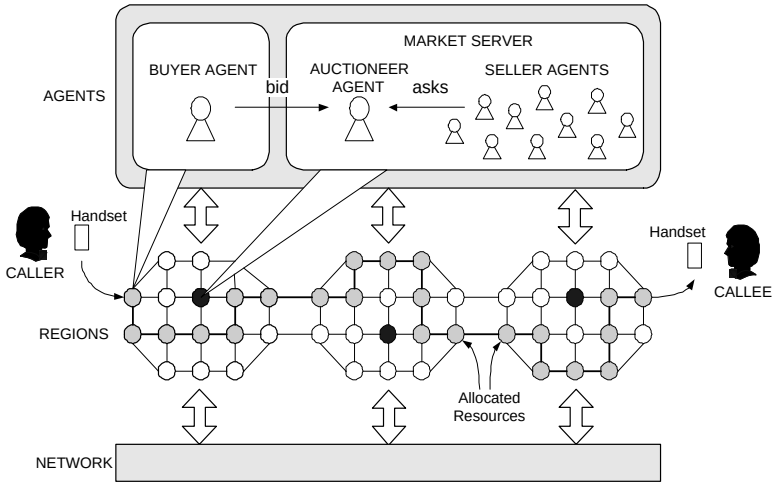


Figure 1: An overview of the system architecture. Black nodes in regions represent market servers and grey nodes represent allocated resources for a particular call from the caller to the callee.

2.1 System Architecture

The system consists of three types of agents: seller, buyer and auctioneer agents (see figure 1). Seller agents are responsible for selling the bandwidth capacity resources and buyer agents are responsible for buying these resources. The auctioneer agents accept *asks* from seller agents and *bids* from buyer agents and conduct auctions so that resources can be allocated using a market-based protocol (a description of which is given in subsection 2.3.1). As can be seen, the overall network is divided into a number of regions (section 2.2 describes what regions are and explains why each one has its own market server). Callers are not regarded as agents within the system but are used to initiate calls via the use of handsets. When a call request takes place, the destination location to where the caller wishes to make the call is passed to the buyer agent on the local node. This agent then starts the process of setting up the call. For each call attempt, a buyer agent in each required region tries to reserve a resource bundle (i.e. set of interrelated resources in a single region) from its local market server. Buyer agents work together to collectively make a complete source-to-destination path across the regions using the bundles i.e. the path is put together in a distributed way. If some resource bundles cannot be obtained for a call, then a backtracking mechanism is used which allows alternative allocations to be made if currently reserved resource bundles cannot lead to the final destination. An example of backtracking is outlined in section 2.4.

2.2 Network Structure and Modelling

As outlined in section 1, it is desirable for resources to be bought and sold in the network from various points and not from a central location. With this in mind, the structure of the network requires consideration. In particular, there are a number of ways in which a market could have been distributed. The two approaches that were considered were to have: (i) resource information replicated across several market servers, where each can sell all of the resources in the entire network, or (ii) to partition the complete resource information such that the market servers sell resources that are not for sale on any other market (i.e. to introduce local network regions that are distinct and where only resources within those regions are sold). We regard a network region as a group of nodes that are situated geographically close together where each region is created in advance of any resources being bought or sold. Nodes on the edge of regions can communicate with other edge nodes in neighbouring regions.

We chose the partitioned approach for a number of reasons. Firstly, if resource information was replicated then the recipient market server would need to contact *all* other markets to make sure that the same resources are not being sold elsewhere, for each bid submitted. This could soon flood the network with messages. This situation is avoided with regions since each market only sells the resources within that region and only the required markets are contacted. Also, the partitioned approach allows the expansion of the network where adding extra regions and markets can take place without significantly affecting any parts of the existing network.

To model the network, each node has a fixed total bandwidth capacity that is split logically into several *equal* parts, where these are the resources. This means that these parts of bandwidth can be used in relaying several calls at the same time through the nodes. Each node has a fixed number of handsets attached from where calls originate. A handset that is currently in use is assumed to be engaged and, thus, cannot be used for any other calls at the same time. Also, currently, control messages are assumed to be routed by a separate communication layer, where there is infinite capacity, for which we do not set an upper bound on the bandwidth or number of messages. We aim to relax this assumption as part of our future work (as described in section 5).

2.3 The Agents

2.3.1 The Auctioneer Agent

Auctioneer agents conduct auctions using a combinatorial reverse auction protocol [4] to allocate goods (units of node bandwidth) to buyers. With this particular protocol, the auctioneer agents try to allocate a *combination of goods* (i.e. a source-to-destination path) that consist of the cheapest possible bundles. There is one auctioneer agent present in each region in the network, each on their respective market server nodes. Market servers are placed manually within a central location in their regions where there is a high connectivity of neighbouring nodes - this is so that they can receive more messages per unit

time than if the connectivity is less. Over a period of time, auctioneer agents execute a *winner determination protocol* that determines which resources are allocated to which parties, every time they have a bid to process.

In more detail, for each bid submitted by a buyer, the set of winning sellers must be found. For each buyer, the auctioneer has a set of resources that it tries to acquire, $M = \{1, 2, ..., m\}$, as specified by the buyer in its bid. Buyers only ever bid for single units of goods for their bundles, since one unit of node bandwidth is assumed to be sufficient capacity for handling a call. They specify for which nodes these single resource units are required: $U = \{u_1, u_2, ..., u_m\}$ where, in this case, $u_i = 1$. Sellers only ever sell one *type* of resource each i.e. the bandwidth of a single node k (where k is a different and unique single node for each seller). They each submit an ask individually where the market eventually receives the set of asks from all sellers: $A = \{A_1, A_2, ..., A_n\}$. Each ask is a tuple $A_j = \langle\ \lambda_j^k,\ p_j\ \rangle$ where $\lambda_j^k \geq 0$ is the number of resource units of node k offered by the ask from the jth seller and p_j is the ask price per unit. The winner determination algorithm then attempts to allocate resources by minimising the amount spent [4]:

$$\min \sum_{j=1}^{n} p_j x_j \quad s.t. \quad \sum_{j=1}^{n} \lambda_j^i x_j \geq u_i,\ \ i = 1, 2, ..., m \qquad x_j \in \{0, 1\}$$

A bid from a buyer agent contains several bundles from which only one is required (see subsection 2.3.3). The winner determination protocol operates by exhaustively incrementing through these, finding the bundles which are available as a complete set. From these bundles, the cheapest one is allocated to the buyer agent (i.e. this algorithm is executed for each bundle in a buyer agent bid until the one with the minimum cost is found). Assuming that a buyer agent's bid is successful, resources are sold at the seller agent's asking price.

2.3.2 The Seller Agents

There are several seller agents per region, one owning each node. The implication of each seller agent owning a node is that they can attempt to compete against each other by pricing their respective resources competitively. All seller agents are physically deployed on their local market server nodes and we assume that they all use the same simple linear pricing strategy for the moment. A seller agent begins with y number of resource units initially priced at one price unit each. For each unit sold, the price increases by one price unit (i.e. when there is only one resource unit left, it should cost y price units). Conversely, for each unit reclaimed by a seller agent, the price reduces by one price unit.

The initial low price of one price unit is chosen so that sellers can sell resources more easily to begin with. As demand for resources increases, the price per unit increases so that buyer agents have to bid more for resources. Given this, seller agents can maximise their utilities by making as much profit as possible. They also reduce the price of resources by one price unit when they have reclaimed the resource so that they can lure more buyers to purchase resources from them (i.e. seller agents remain competitive against each other).

2.3.3 The Buyer Agents

Buyer agents purchase node capacity resources from seller agents within the system and are funded by callers so that resource bundles can be bought. The bundles establish a complete path from the caller's source location to the destination location that the caller wishes to contact. These resources allow calls to be made across the network. There is one buyer agent per node. They are put on individual nodes so that they can await call requests, from callers, from any point in the network. The number of buyer agents required in setting up a call is the same as the number of regions in which resources are required for a given call (i.e. different buyer agents purchase resources in their own respective regions in order to make a complete path across several regions, for multi-region calls). For a single-region call, only a single buyer within that region is required to set up that call. If the call request involves several regions, then other buyer agents are contacted to purchase resources in their regions. The process of reserving resources across several regions is described in detail in section 2.4.

The current buyer agent bidding strategy is simple and assumes that buyers have knowledge of the price of all resources within their own regions.[1] However, it must be noted that buyer agents do not know the current availability of resources, as this would be unrealistic. We assume that all buyer agents use the same purchasing strategy. Thus, when a buyer agent receives a request for purchasing node bandwidth, it then formulates its bid. In doing so, we assume that buyers have knowledge of how all of the *regions* are connected together in the network as well as in which regions all nodes are situated.[2] Therefore, once the buyer knows the final destination to where it purchases the resources, it finds the cheapest set of routes that lead from its current node to a destination node within its own region. These are then sent as a bid to the buyer's local market. If the final destination node is within the same region, then that node is the destination node. If, however, the final destination is in another region, then the buyer finds a set of routes that lead to a node within its current region that is connected to a node in a neighbouring region that leads to the region where the final destination node is. Since the buyer agents have knowledge of resource prices, they select a set of bundles that minimise the cost of their desired routes.

A buyer agent would like to obtain only one bundle from the set that it submits to its local market. Therefore, we make the assumption that buyer agents are only allowed to submit up to a certain number of bundles for each bid. The value chosen here is five because we wanted to allow some choice and flexibility in the bundle that a buyer could be allocated and yet not choose a number that is so high that the market algorithm has to do significant amounts of unnecessary processing.[3] Finally, if the buyer agent is successful in reserving resources, it is informed by the local market.

[1] More advanced buyer strategies will be investigated as part of the future work.

[2] Buyer agents do not know the entire topology of how all nodes in all regions are connected together.

[3] A future investigation will be to look into exactly how much processing is done when the number of bundles submitted is altered.

2.4 Acquiring Resources Across Regions

In a multi-region call, when a buyer agent has successfully reserved a bundle of resources, the market server in that region is responsible for contacting a buyer agent that is on the edge of the next region. The node on which this second buyer agent resides must be in reach of the last node in the bundle of resources that have been reserved in the previous region such that when the call eventually takes place, there should be a continuous path from the source node to the destination node. To this end, the reservation procedure is described next, followed by the backtracking mechanism that releases resources that are no longer required and attempts to reserve alternative bundles for a given call, when a complete path cannot be made.

2.4.1 Resource Reservation and Commitment

Figure 2 shows the actual network topology used in our experiments (see section 3). Therefore, we now use it to demonstrate how buyer agents attempt to reserve resources. The market servers in regions 0, 1, 2, 3 and 4 are assumed to be resident on nodes 3, 16, 26, 38 and 46, respectively. For this example, we assume that the source of the call is from node 0 and the destination is on node 49. When a call request arrives in region 0 on node 0, the buyer agent on that node, say b_1, sends a bid to its local market. Here, we assume that b_1 has successfully reserved the path containing resources 0-1-4-7. The market server on node 3 then contacts a buyer agent in region 1 so that it can purchase the next set of resources. It makes a random decision of selecting a buyer agent on either node 8 or node 11 since these are directly in reach of node 7 in region 0, where node 7 is the last resource in the reserved bundle. In this example we assume that node 8 is chosen on which the buyer agent, b_2, resides. Therefore, b_2 is given the responsibility of bidding for a set of resources in region 1. This process continues until the final destination is reached. Hence, there is an element of cooperation between buyers in different regions when paths are being reserved. Buyer agents reserve resources only from local markets because the complete network is split up into regions. Local markets only sell resources in the local region in which they are operating.

Once the final destination has been reached, the market server in the last region (region 4) sends a *commit* message to the buyer agent within its own region. This buyer agent then contacts the market server in the previous region (region 1) which, in turn, informs its buyer agent, b_2, about the complete path being reserved. Payment of resources takes place during this commit phase. Eventually, the originating buyer agent, b_1, receives the commit message and the call can be placed. Once the call has completed, a message is sent from b_1 in region 0 to its local market that resources need to be released. After this has been done, this message is then propagated across all used markets in the direction of the final region so that resources can be released. The markets can then resell the resources to the buyers that place bids for them in the future.

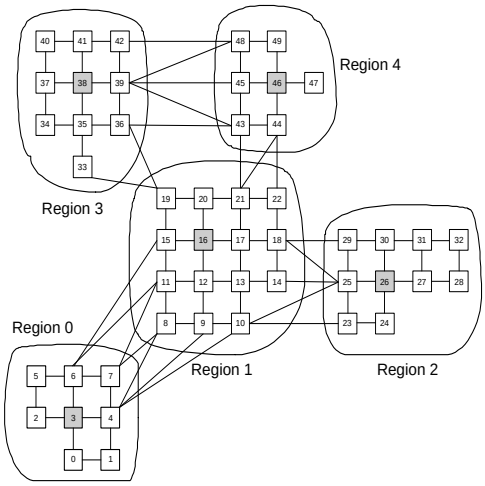


Figure 2: A 50 node network topology that has been partitioned into 5 distinct regions. The grey nodes show where the hand-picked market servers reside.

2.4.2 The Backtracking Mechanism

As part of our solution, the system uses a backtracking mechanism that allows alternative allocations to be made if currently reserved resource bundles cannot lead to the final destination. Thus, if a buyer agent in an intermediate region fails in reserving a bundle of resources, then it resubmits another bid to its local market which contains bundles that lead to another destination node within its own region. This process continues until either a bundle has been reserved or there are none left. In the latter case, the market in the previous region is informed and the previous buyer agent releases its currently reserved resource bundle and bids for another set of resources that lead to a different region.

Using figure 2 as an example, if b_2 on node 8 fails in being allocated a resource bundle from node 8 to node 21, then it can submit a second bid for a route that leads from node 8 to node 22. If this also fails, then b_2 would know that all routes that lead directly to region 4 have been exhausted. Therefore, it could try for a bundle that leads to region 3 (i.e. node 8 to node 19). If b_2 is successful in receiving such a bundle (e.g. 8-11-15-19), then the buyer agent on node 33 in region 3 can continue in setting up this call by bidding for a bundle of resources that lead from its region to region 4. In short, the agents in the system perform a distributed depth first search of the resource bundles when bids are made (a complete description of the system algorithm is given in [5]).

3 Experimental Evaluation

This section describes the experimental work that was carried out in evaluating the system. Section 3.1 describes the methodology and experimental parame-

ters and the results are outlined in sections 3.2 and 3.3.

3.1 Experimental Methodology and Settings

In order to evaluate our system, it was benchmarked against two other controls. These consist of the global optimum values, as well as a random strategy that is used for allocating resources. For both controls, as well as our algorithm, we assume that one hop in the network takes one simulation time step. When a *source-destination* pair has been selected for a call attempt in our simulation, then the same pair is used for the optimum and random strategies.

The global optimum strategy works in an entirely impractical way that gives it a number of significant advantages over our system. The optimum strategy assumes that it has global knowledge of all of the resources available at any moment in time. In more detail, at the time of a call originating, a complete global search is done to see if a path exists that leads from the source node to the destination node. If one is found, then this is deemed to be a successful allocation attempt. This test is performed on each time step during the set up period when a call attempt is made, until a solution is found. Whilst one hop in the network is assumed to take one time step, we assume that the global optimum strategy provides an instantaneous allocation, when measuring the call success rate (see section 3.2 for details concerning this experiment). If no source-to-destination path is found before a call has been set up in our system, then it is considered to have failed in the optimum strategy. With the random strategy, a randomly chosen neighbouring node is selected and a check is done to see if there is sufficient call capacity for it to accept a call. If so, it is made into the current node. If not, then the previous node must select another neighbouring node. The search process continues until either the final destination node has been reached or until there are no more neighbouring nodes to contact. If the final destination is found, then the random strategy is considered to have succeeded in its allocation attempt. To avoid cyclic routes and reserving multiple units of bandwidth on the same node, the nodes are not allowed to contact neighbours where resources have already been reserved.

The experimental settings we used in this evaluation were obtained from a domain expert. Specifically, each experiment was run for a total of 100,000 time steps. The simulation was probed after every 1,000 time steps. The duration of a call was set to 500 time steps. We assume that each node has 2 handsets attached to it. Also, each node has a total of 10 units of node bandwidth capacity available. This means that a node can handle up to 10 simultaneous calls at any one time. Calls were made to originate after every 25 time steps. The cost of calls were set at 35 price units per region. For each experiment, the *call origination probability* (traffic load in the network) was increased. Also, the number of simulation runs for each experiment was sufficient for the results to be *statistically significant* at the 95% confidence level. The network topology on which our system operates was shown in figure 2. This was chosen because it demonstrates a topology which has a central region (region 1) through which many calls would require resources in multiple regions.

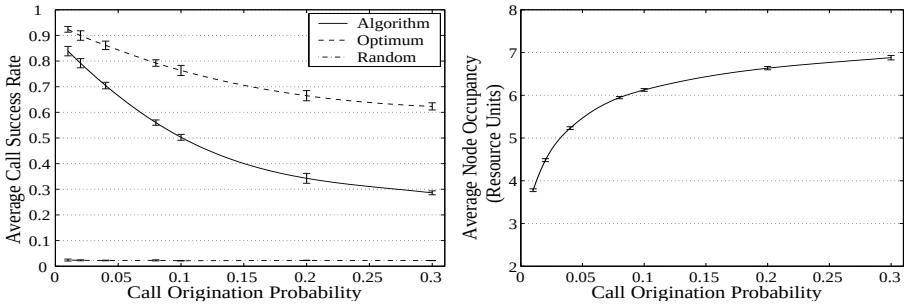
To evaluate our system, we wish to measure the average call success rate (section 3.2). This provides us with an insight into a fundamental measure of the percentage of successful calls that can be placed given different traffic loads in the network. We also look at the average time required for a call to be set up (section 3.3). For all experiments, graphs are plotted each of which show the standard deviation by using error bars.

3.2 Average Call Success Rate

The purpose of this experiment is to investigate the number of calls that could successfully be set up, on average, when varying the *call origination probability*. The hypothesis for this experiment was that if the *call origination probability* is increased, the call success rate would decrease, assuming that all other variables remain constant. As can be seen from figure 3(a), the call success rate does indeed decrease, but it does so at a steady rate. The reason for this is that as the *call origination probability* is increased, the bandwidth capacity in the nodes is used more (or occupied for longer periods of time) and therefore, bandwidth is more scarce. This is proved by figure 3(b), which shows that as the load in the network is increased, the usage of nodes is greater. Figure 3(a) also shows that our algorithm performs considerably better than the random strategy. In particular, the average call success rate does not increase with the random strategy when the load is increased because nodes are not allowed to communicate with neighbouring nodes that have already been contacted for a given call. When there are no more neighbouring nodes left, the calls are dropped. This dictates the overall poor performance of the random strategy, regardless of the load in the network.

In more detail, the results in figure 3(a) show that when the *call origination probability* was set at only 0.01 (1% load), our system successfully allocated 84% of the calls, where the global optimum was only marginally higher at 92%. This shows that the system performs comparatively well at a light load. We would expect the global optimum strategy to perform comparatively better than our algorithm because of the many advantages it is given in terms of information and processing capability (as was detailed in section 3.1). As traffic load increases, the difference in average call success rate between the optimum strategy and the system algorithm becomes larger. The reason for this is that increasing the traffic load induces more contention for resources, which has a larger effect on the algorithm than on the optimum strategy. This can be explained by the fact that our system attempts an *exhaustive* search across the network for resource bundles. Doing so means that a certain percentage of resource bundles are being reserved and are unused for periods of time and this prevents some other calls from being set up. In the case of the optimum strategy, allowing allocations instantaneously means that resources are never occupied unnecessarily for any amount of time, even when load is increased in the network. In order to try to get our system to perform as close as possible to the optimum solution, we aim to limit the amount of backtracking in the system and to make the buyer agents bid more intelligently. Consequently,

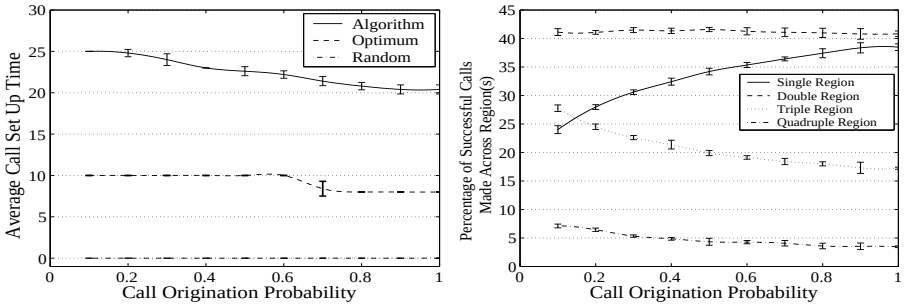
future experiments will be conducted in order to see how well the allocations are being utilised.



(a) Average call success rate against call origination probability.

(b) Average node occupancy against call origination probability (for Algorithm).

Figure 3: Graph plots for the experiment described in section 3.2.



(a) Average call set up time against call origination probability.

(b) Percentage of successful calls made across region(s) against call origination probability (for Algorithm).

Figure 4: Graph plots for the experiment described in section 3.3.

3.3 Average Call Set Up Time

The purpose of this experiment is to investigate how long it takes, on average, for a call to be set up when varying the *call origination probability*. The hypothesis was that if the *call origination probability* is increased, then the average time taken for call set up will be longer, assuming that all other variables remain constant. Figure 4(a) shows that as the *call origination probability* was increased, the average call set up time actually decreased. Specifically, figure 4(a) shows that by using the system algorithm, calls took a longer time to be set up than with using the optimum strategy. The reason for this is that by using the algorithm, call set up time takes longer because a few messages are

required between market servers and buyer agents, within and across regions. The optimum strategy does not require such messages. Using the random strategy, the average call set up time is marginally above 0 time steps. This result gives a false impression of this strategy performing well. The result can be explained by the fact that very few calls are successfully set up with the random strategy (as indicated by figure 3(a)) and that these are all short distance calls of only a few hops in length.

For our system algorithm, our intuition for calls taking shorter time when increasing load was that more shorter distance calls were being set up than longer distance calls. Figure 4(b) shows how the percentage of successful calls that were made across one or more regions was changing as load was increased. This showed that the percentage of single region calls increases when *call origination probability* is increased, double region calls stay approximately the same and that triple and quadruple region calls decrease. We also intuitively know that single region calls, on average, would take a shorter time to set up than double region calls, which in turn take less time than triple region calls, and so on. This indicates that increasing load means that the average number of regions used for a successful call decreases, which explains why the average call set up time also decreases with load.

4 Related Work

There are several market-based architectures that have been proposed for allocating resources in a distributed environment. Gibney and Jennings [6] describe a system in which agents compete for network resources in distributed markets so that calls can be routed in a telecommunications network. The system used a double auction protocol [7] with sealed bids. Results showed that as more resources were being used, the price of resources marginally increased such that eventually the buyers bought alternatives paths. This provided good utilisation of the network and also balanced the load in the network. However, a drawback was that if some resources on a path were already bought and the next desired resource could not be obtained, then the resources already bought could become redundant and a certain amount of money would be spent unnecessarily. In contrast, our reserve/commit mechanism ensures that this situation is avoided by releasing unused resources immediately and allowing payment to occur only after all necessary resources have been successfully reserved.

The *Global Electronic Market System* (GEM) [8] is a framework for decentralised markets across the Internet. GEM has a single market which is distributed on which goods are sold.[4] The general idea in GEM is that agents initially trade in local markets and when required, inter-market communication takes place between other markets. The GEM system is different from traditional independent local markets because the markets are replicated and the order for goods is distributed across these markets. Multiple markets are used in GEM to increase the probability of finding a match for a resource. If a

[4]The resources that are allocated in GEM are not necessarily *network* resources.

market is heavily loaded, then it is possible that another market can be used for obtaining resources. Looking at GEM provided an insight into one method of how servers in a market-based resource allocation system could be distributed. However, the approach taken by GEM of replicating the resource information is not suitable for our system because it induces more messages in the network than our partitioned approach (as was detailed in section 2.2).

MIDAS [9] is an auction-based mechanism that allocates link bandwidth in a network for making paths. Simultaneous multi-unit Dutch auctions were used as the protocol for allocating the resources. However, this auction protocol would be inadequate with respect to our requirements since it is not capable of allocating several interrelated goods at the same time. Finally, Ezhilchelvan and Morgan [10] have looked at how an auction system can be distributed across several servers in a network of servers. However, this approach assumes that communication takes place using a high-bandwidth network which is an assumption that cannot be made within our work.

5 Conclusions and Future Work

In this paper, a system was described that allocates end-to-end bandwidth to set up calls in a network using market-based agents. The system used a combinatorial reverse auction where bundles of interrelated resources were allocated and novel *reserve* and *commit* mechanisms were developed to cope with the partitioned nature of the distributed marketplace. Empirical evaluation showed that our system successfully set up considerably more calls than that achieved by the random strategy given all traffic loads. It also set up a comparable number of calls when put side by side with the optimum strategy given a light network load. Results also showed that the average time taken for a call to be set up is longer when the load in the network is at its lightest. This was explained by the fact that the percentage of longer distance calls decreased as load was increased and vice versa and that, intuitively, we know that shorter distance calls take less time to set up.

Whilst the optimum used to benchmark against our system was unrealistic, there are a number of ways in which our system can be improved. Firstly, we aim to develop agent strategies that are more realistic with respect to the current assumptions made. Specifically, buyer agents will need to make realistic estimates on the price of resources without knowing the actual prices a priori. In order to achieve this, various techniques such as learning and heuristic methods will be investigated for allowing buyer agents to calculate resource prices. Secondly, we aim to account for a finite number of control messages within our simulation. Currently, our system assumes that there is an infinite amount of bandwidth available for control messages. Finally, in order for our system to perform as close as possible to the optimum, we plan on limiting the amount of backtracking that will take place. It is envisaged that in doing so, as well as allowing buyers to bid more intelligently in the first instance, would mean that less resources would be reserved unnecessarily. Therefore, more resources will

be available for other calls. This should also reduce the average set up time for calls too, which is desirable.

6 Acknowledgements

The research in this paper is part of the EPSRC funded Mohican Project (Reference no: GR/R32697/01). We would also like to acknowledge the contribution of Steve Braithwaite who provided us with domain expertise.

References

[1] P. Nicopolitidis, M. S. Obaidat, G. I. Papadimitriou and A. S. Pomportsis, Wireless Networks, John Wiley & Sons Ltd, Chichester, England, 2003.

[2] L. L. Peterson and B. S. Davie, Computer Networks: A Systems Approach, Morgan Kaufmann Publishers Inc, San Francisco, California, 2000.

[3] S. H. Clearwater, Market-Based Control: A Paradigm For Distributed Resource Allocation, World Scientific Publishing Co. Pte. Ltd, Covent Garden, London, 1996.

[4] T. Sandholm, S. Suri, A. Gilpin and D. Levine, Winner determination in combinatorial auction generalizations, In AGENTS-2001 Workshop on Agent-Based Approaches to B2B, Montreal, Canada, 2001.

[5] N. Haque, Resource Allocation in Communication Networks Using Market-Based Agents, Technical Report, School of Electronics and Computer Science, University of Southampton, Southampton, UK, May 2004.

[6] M. A. Gibney and N. R. Jennings, Dynamic Resource Allocation by Market-Based Routing in Telecommunications Networks, Springer-Verlag: Heidelberg, Germany, 1998, volume 1437, pages 102–117.

[7] P. Wurman, W. Walsh and M. Wellman, Flexible double auctions for electronic commerce: Theory and implementation, Decision Support Systems, 1998, volume 24, pages 17–27.

[8] B. Rachlevsky-Reich, I. Ben-Shaul, N. Tung Chan, A. W. Lo and Tomaso Poggio, GEM: A Global Electronic Market System, Information Systems, 1999, volume 24, number 6, pages 495–518.

[9] C. Courcoubetis, M. Dramitinos and G. D. Stamoulis, An Auction Mechanism for Bandwidth Allocation Over Paths: New Results, M3I Modelling Workshop, London, UK, June 2001.

[10] P. D. Ezhilchelvan and G. Morgan, A Dependable Distributed Auction System: Architecture and an Implementation Framework, International Symposium on Autonomous Decentralized Systems, 2001, pages 3–10.