Scalability and Robustness of a Network Resource Allocation System Using Market-Based Agents

Nadim Haque (n.a.haque@ecs.soton.ac.uk),
Nicholas R. Jennings (n.r.jennings@ecs.soton.ac.uk),
Luc Moreau (l.moreau@ecs.soton.ac.uk)
School of Electronics and Computer Science, University of Southampton, Southampton SO17 1BJ, UK.

Submission: Nov 1, 2005
First Revision: Apr 12, 2006
Second Revision: May 15, 2006
Accepted: May 25, 2006

Abstract. In this paper, we consider issues associated with scalability and robustness in designing a market-based multi-agent system that allocates bandwidth in a communications network. Specifically, an empirical evaluation is carried out to assess the system performance under a variety of design configurations in order to provide an insight into network deployment issues. This extends our previous work in which we developed an application that makes use of market-based software agents that compete in decentralised marketplaces to buy and sell bandwidth resources in a network that is partitioned into regions, each with a separate market server. We investigate the average call success rate and average message load per market server, as the number of markets are scaled up in a fixed size network. The same investigations are performed in the presence of single market failures. Finally, for both the failure and non-failure cases, a trade-off is found between their average call success rates and message load per server in order to find an optimum number of regions to deploy in the network.

Keywords: Agent, Communications Network, Market, Resource Allocation, Scalability, Robustness

Resource allocation is a crucial problem in effectively managing 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 (Peterson and Davie, 2003). This is a challenging problem since resources become scarce when there is a high demand for them. Specifically, in this work, we consider a circuit switched meshed network where nodes communicate with their immediate neighbours using radio links (Nicopolitidis et al., 2003). In this system, the nodes are designed to consume as little power as possible, to be as robust as possible and are targeted for rapid and cost-efficient deployment in poor countries. However, such low power consumption implies that there is limited bandwidth available in the network. To cope with this, we have developed a multi-agent system that allocates end-to-end (source-to-destination) bandwidth in such communications networks to set up calls. More specifically, based on our previous solution (Haque, Jennings and Moreau, 2005a), in which we deployed a market-based approach, in this paper we consider a network that is partitioned into a number of non-overlapping regions, each with its own market server, from where resources are allocated. Using regions and decentralised markets in this way means that there is no central point of failure from where all resources are allocated. In this context, we focus on the impact that varying region size and, hence, the number of markets has on the network and on how robust the system is in the face of failures in the network. By investigating these issues, we provide a network designer with an insight into how such a network can be deployed in practice.

In recent years, market-based approaches have been investigated and used to solve various problems in a wide range of applications. Specifically, markets have been used in computational grids (ChunLin and Layuan, 2005), peer-to-peer systems (Ratsimor et al., 2003), supply-chain management (Keller, Duguay and Precup, 2004), scheduling (Wellman et al., 2003), congestion control (Heikkinen, 2002), routing (Goemans et al., 2004), workflow automation (Hulaas, Stormer and Schnhoff, 2001) and recommender systems (Wei, Moreau and Jennings, 2005). Building on this, the solution that we have developed consists of software agents that compete in a marketplace to buy and sell network bandwidth. Here, buyer agents represent callers and seller agents represent the owners of the resources. 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. More specifically, a market-based approach was chosen for the following reasons. First, markets are effective mechanisms for allocating scarce resources in a decentralised fashion (Clearwater, 1996). 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, generally speaking, they ensure the individual that values the resources the most will obtain them.

In more detail, our system is a distributed market mechanism in which allocations of *interrelated* resource bundles are sold in multiple markets (Haque, Jennings and Moreau, 2005a). The marketplace protocol incorporates a reservation and commitment mechanism that provides a guarantee that a partial set of bandwidth resources will not be bought if a complete source-to-destination path cannot be made. Against this background, we focus in particular on analysing the system behaviour after performing scalability and robustness tests, with respect to increasing the number of regions in a fixed size network.

The remainder of this paper is structured as follows: the marketplace design and components are briefly recapped in section 1. Section 2 provides the system formalisation where the functionality is described. The system evaluation and experimental results are presented in section 3. Section 4 describes related work and, finally, section 5 concludes.

1. Marketplace Design

This section describes the design of the system. Specifically, the basic components and network model are outlined in section 1.1. Section 1.2 describes the constituent agents and, finally, section 1.3 provides a brief description of how resources are acquired in a multi-region call.

1.1. Network Model

The system consists of three types of agents: seller, buyer and auctioneer (see figure 1). Seller agents are responsible for selling node bandwidth capacity resources and buyer agents are responsible for buying these resources. The auctioneer agent accepts *asks* from seller agents and *bids* from buyer agents and conducts auctions so that resources can be allocated using a market-based protocol. As can be seen in the figure, the overall network is divided into a number of regions (3 in this case). Callers use handsets to initiate calls. 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

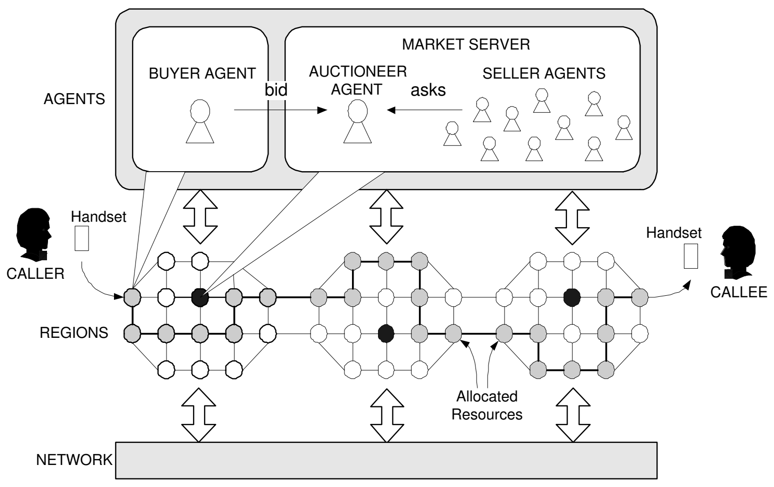

Figure 1. An overview of the system architecture. Black nodes in regions represent market servers and grey nodes represent allocated resources for a particular call from the caller to the callee.

tries to reserve a resource bundle (i.e. a set of resources in a single region) from its local market server. The resources in a bundle are interrelated which, in this context, means that they form a path within the region and are not just any resources. Buyer agents work together to collectively make a complete source-to-destination path across the regions using the bundles. If a resource bundle cannot be obtained, then a backtracking mechanism is used which allows alternative allocations to be made if currently reserved bundles cannot lead to the final destination.

As outlined in the introduction, it is desirable for resources to be bought and sold in the network from various points and not from a central location (since a single server would constitute a central point of failure). With this in mind, we partitioned the network into regions where only resources within those regions are sold (i.e. there are multiple market servers in the network, one placed in each region). Thus, if we are to look into the scalability of the number of regions within a fixed size network (i.e. the region scalability), we must consider how the network should be divided. Here we regard a network region as a group of nodes that are situated geographically close together. By geographical position, we refer to the physical placement where nodes are positioned in a way such that they can communicate with other nodes that are within their transmitter ranges. These are typically neighbouring nodes. Thus, in order for any two nodes to be able to communicate, they must be positioned geographically close together so that they are within reachable distance of each other. In this application, we consider a static network configuration as defined at deployment time. Nodes on the edge of regions can communicate with other edge nodes in neighbouring regions. In using local markets, resource information does not have to be replicated across all markets in the network. This is good since the market server that receives a bid from a buyer does not need to contact *all* other markets to make sure that the same resources are not being sold elsewhere, for each bid placed. The downside, however, is that allocations that span multiple markets need to be closely coordinated.

To model the network, each node has a fixed total bandwidth capacity that is split logically into several *equal* parts, which are the resources that are bought and sold by the market mechanism. These resources are 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. Our current work assumes that control capacity is separate from the bandwidth capacity used for relaying calls. The resources we consider are for calls and not for control messages. In this work, we do not look into the usage of control capacity and leave this investigation for future work. However, in sections 3.2.2 and 3.3.2, we do look at the number of messages received per market server, since it is on these servers that the majority of the processing takes place.

1.2. The Agents and the Markets

Auctioneer agents conduct auctions using a combinatorial reverse auction protocol (Sandholm et al., 2002) to allocate goods (units of node bandwidth) to buyers (i.e. they allocate a *combination of goods* that consist of the cheapest possible bundles). There is one auctioneer agent per region in the network, each on their respective market server nodes. Auctioneer agents execute a *winner determination protocol* that determines which resources are allocated to which parties, for each bid submitted. There are several seller agents per region, one owning each node, where they each submit an individual ask price to their local markets. 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. To minimise communication in the network, all seller agents are physically deployed on their local market server nodes.

In an auction, sellers traditionally set their own reserve prices where this price must be met in order for the resource to be allocated. Thus, given that market servers in our system use an auction-based mechanism to allocate resources, the sellers price their individual resources by setting their respective reserve price per unit, where the auctioneer on its market server is only responsible for conducting the combinatorial reserve auction. The ask prices from sellers (and bids from buyers) are sent to auctioneer agents by means of sealed bids (meaning prices are not visible or published). Sealed bids were chosen to eliminate communication between sellers since this would otherwise be costly in terms of time and the additional computational processing required. However, since no seller is aware of the price and quantity of other resources in their regions, an equilibrium price cannot be computed by sellers so bidding is carried out in a heuristic manner.

We assume that, currently, sellers all use the same linear pricing strategy. A seller agent begins with a total of λ 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 λ price units). Conversely, for each unit reclaimed by a seller, the price reduces by one price unit. The initial low price per 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 pay more for resources and seller agents can increase their profit. Sellers also reduce the price of their resources by one price unit when they have reclaimed a resource so that they can remain competitive against other seller agents.

In our work, a buyer submits a bid composed of several bundles, of which only one is required. The winner determination algorithm then attempts to allocate resources by minimising the amount spent. In more detail, for each bid submitted by a buyer, the set of winning sellers and thus, resources, 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 each bundle in a buyer 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 for each bundle: $U = \{u_1, u_2, ..., u_m\}$ where, in this case, $u_i = 1$. Sellers only ever sell one *type* of resource each (i.e. the bandwidth of a single node which is a different and unique node for each seller). They each submit an ask individually where the market eventually receives the set of asks from all sellers: $A = \{A_1, A_2, ..., A_m\}$. Each ask is a tuple $A_j = \langle \lambda_j, p_j \rangle$ where $\lambda_j \geq 0$ is the number of resource units of a node offered by the ask from the *j*th seller and p_j is the ask price per unit for that particular resource. This is described by the following:

$$\mathit{bundle} = \sum_{i=1}^{m} p_i x_i \quad s.t. \quad \sum_{i=1}^{m} \lambda_i x_i \geq u_i, \;\; i = 1, 2, ..., m$$
$$x_i \in \{0, 1\},$$

where this is computed for each *bundle* in a buyer bid and where the *bundle* with the minimum cost is allocated from the ones submitted by the buyer agent in its bid. (x takes the value of 1 if resource i is wanted, or 0 otherwise).

Here we use a relatively straightforward technique for solving the winner determination problem because the numbers of bids that need to be considered at each market server are relatively small.[1] A buyer requests one bundle of resources from a set of bundles submitted to its local market. On receipt of these bundles, the auctioneer iterates through the resources in each bundle, one at a time, in order to check if all the resources are available. Whilst doing this, the auctioneer sums up the values of the resources in each bundle to compute the total price of each bundle. When iterating through the bundles, if a resource is not available, then that bundle can be discarded since an incomplete bundle is inadequate to form a path. Then, from the bundles that are available as a set, the cheapest one is allocated to the buyer so that a sub-path can be made. Our approach is inspired by using pure market-based approaches, which focus only on price to decide which bundle to choose. This implies that the bundle allocated by an auctioneer may not be the shortest or rely on more robust nodes: price however provides an indication of resource availability. For bundles with an identical cheapest price, the shorter bundle is chosen since less resource units are consumed. If the price and bundle size are the same, then the bundle is chosen randomly. Thus, this is the procedure used to find the set of winning sellers whose resources are successfully reserved.

[1] For larger numbers, more sophisticated optimisation techniques may be needed (see (Kelly(2004)) for a more detailed discussion).

Assuming that a buyer agent's bid is successful, resources are sold at the seller agent's asking price. There is one buyer agent placed on each individual node where they await call requests, from callers, from any point in the network. We assume that all buyer agents use the same purchasing strategy. Thus, when a buyer agent receives a request for purchasing node bandwidth, it formulates its bid. It attempts to find the cheapest set of routes that lead from its current node to a destination node within its own region. Now, in this work, we assume that buyer agents select a set of bundles that minimise the length of their desired routes. The intuition here is that the buyer believes shorter routes are generally cheaper since they contain fewer resources.

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 must be enough to allow some flexibility in the bundle that a buyer could be allocated, but it should not be so high that the market algorithm has to do significant amounts of unnecessary processing. If the final destination node is within the same region, that node is the destination node. The bundles selected by a buyer agent are sent as a bid to the buyer's local market. Finally, if the buyer agent is successful in reserving resources, it is informed by the local market. Callers are assumed to pay for successfully established calls per region where the cost per region is proportional to the number of resources in the region. This is realistic because, generally speaking, the longer the distance of a call, the more resources are used and, therefore, the more the call would cost.

1.3. Acquiring Resources Across Regions

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. If

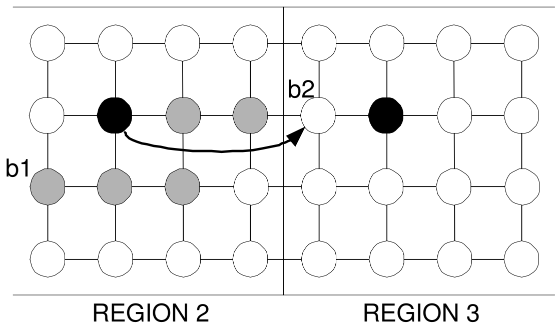

Figure 2. A market server in region 2 (black node) contacting a buyer agent, b2, in the following region for setting up a multi-region call. Grey nodes represent a reserved resource bundle for b1 in region 2.

the final destination for a call is in a different region from the one in which a buyer is located, then this buyer will attempt to find routes that lead to a node within its region that is connected to a node in a neighbouring region that leads to the final destination. Then, in a multi-region call, once 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. Thus, these boundary nodes will relay the call from the initiating source node to the final destination node, such that there is a continuous path when/if the call eventually takes place.

Figure 2 illustrates part of the reservation process. This figure shows a buyer agent, b1, which has already successfully been allocated a resource bundle (shown by the grey nodes) by its local market server within its region (region 2 in this case). This market server in region 2 then attempts to contact another buyer agent, b2, which is on a boundary node in the following region so that b2 can then bid for a resource bundle within its own region (region 3). The reservation process continues in each required region until the final node is reached and the call is set up. In general, the system allows buyers to choose which region to contact next when there is a choice of following regions in a multi-region call. In such a case, the regional route that involves the use of the fewest regions for the set up of a call is preferred (i.e. the shortest regional route).

Once the final destination has been reached, the market server in the last region sends a *commit* message to the buyer agent within its own region. This buyer agent then contacts the market server in the previous region which, in turn, informs its buyer agent and so on, until the initial region is reached. Eventually, the originating buyer agent receives the commit message and the call can be placed. Thus, there is a reservation and commitment process that takes place in the system, where payment for resources only takes place during the commit phase once all of the necessary resources have been acquired. When the call has completed, a message is sent from the initial buyer to its local market (and to all other markets and buyers involved in this call in the direction of the final region) to signal that resources can be released. The markets then resell the resources to buyers that place bids for them in the future. (The overall behaviour of the system described here is outlined, from a global perspective, in a formalisation that appears in section 2).

As part of our system, if a buyer agent in an intermediate region fails in reserving a bundle of resources, then backtracking is used to allow alternative allocations to be made. This occurs if currently reserved resource bundles cannot lead to the final destination. A buyer agent can resubmit another bid to its local market which contains bundles that lead to another destination node within its own region (i.e. to a different boundary node). This continues until a bundle has been successfully reserved or there are none available. In the latter case, the previous region is contacted to search for other routes which have previously been unexplored. Thus, agents perform a distributed search for resource bundles.

2. System Formalisation

This section provides a formal description of the system that was previously outlined in section 1. The definitions of the variables are given first. This is followed by a description of the global system behaviour, along with an algorithm that describes this behaviour.

The definitions of the system variables now follow.

- *allocation* : a bundle of bandwidth resources actually allocated to a buyer;
- *availablebundles* : sets of bandwidth resources that are currently available, as complete bundles, from the local market server;
- *calldest* : the final call destination node to which a complete resource path is to be made, for a particular call;
- *region* : the current network region.

The system behaviour is outlined in algorithm 1. A description of the main functionality of the algorithm also follows. Algorithm 1 starts with calls originating from handsets where call attempts are made (lines 1 to 3). Here, the local region for each handset is obtained (*getLocalRegion*) along with the call destination (*getCallDestination*).

The processCallAttempt procedure : This procedure (lines 5 to 23) describes the outline of the system algorithm from a global viewpoint. An

Algorithm 1 System Algorithm

```
for all $h \in H$ do                                    ▷ where $H$ is the set of all handsets
    $processCallAttempt(h.getLocalRegion(), getCallDestination())$
end for

procedure $processCallAttempt(region, calldest)$
    $allocation \leftarrow reserveResources(region, calldest)$

    if $allocation \neq null$ then
        $notifyBuyer(allocation)$

        if $isFinalDestination(region, calldest)$ then
            $notifySellers()$
            $commit()$
            $makeCall()$
            $releaseResources()$
        else
            $processCallAttempt($
                $contactNextRegion(region, calldest), calldest)$
        end if
    else
        $retryAnotherRoute(calldest)$
    end if
end procedure

procedure $reserveResources(region, calldest)$
    $availablebundles \leftarrow checkBundleAvailability(region, calldest)$
    $allocation \leftarrow allocateCheapestBundle(availablebundles)$

    if $allocation \neq null$ then
        $removeResourcesFromRepository(allocation)$
    end if

    return $allocation$
end procedure
```

attempt is made to reserve resources (*reserveResources*) in the current region where, initially, this is the region where the call originated from. If the allocation is successful, then the buyer is notified (*notifyBuyer*). A check is then done to see if the current region is the final destination region (*isFinalDestination*). If so, then all necessary resources have been reserved successfully and the sellers are informed of the sale of their resources (*notifySellers*). This occurs at the same time as the commit message being sent, via (*commit*), between market servers and buyers that were involved in setting up the call across all regions where resources were reserved. The call is then made (*makeCall*) after which the resources are released (*releaseResources*). If the current region does not contain the call destination, then the next region is contacted (*contactNextRegion*) which leads towards the call destination. This is so that the resource reservation process can continue. Finally, if an allocation was not successful, then the backtracking process (*retryAnotherRoute*) takes place where alternative routes are searched for in the current region or in the previous region. The backtracking process continues until either a complete source-to-destination path is found or until all paths searched for have been exhausted.

The reserveResources procedure : This procedure (lines 25 to 34) is executed by auctioneer agents, each on their respective market servers. It first checks to see which bundles of bandwidth resources are available as complete sets of resources (*checkBundleAvailability*) and then allocates the cheapest one (*allocateCheapestBundle*). If an allocation is made, it is removed from the local market server repository so that these same resources cannot be re-reserved for another buyer.

3. Experimental Evaluation

This section describes the experimental work that was carried out in evaluating the scalability and robustness of the system with respect to increasing number of regions. Section 3.1 describes the methodology and parameters used, while results for scalability and robustness testing are outlined in sections 3.2 and 3.3, respectively.

3.1. Experimental Methodology and Settings

Previously, we have explored the basic behaviour of our system in terms of *average call success rate* and *average call set up time* when compared against optimum and random strategies. This provided us with an insight into a fundamental measure of the percentage of successful calls and set up times, respectively, given one particular network setting. While this was promising, the aim of our current work is to extend this evaluation in order to ascertain its properties in a wider range of situations. In particular, we are interested in issues associated with region scalability and robustness. Looking at the former will give us an understanding of how the structuring of the network into regions impacts the performance for a given network size. Testing for the latter will show how well the system performs with market failures (which obviously occur in real world settings).

In our scalability analysis, we measured the average call success rate when progressively increasing the number of regions in a fixed size network (see section 3.2.1). We also look at the number of messages received per market server in the network when scaling up the number of regions (described in section 3.2.2). These are two important measurements that were used to evaluate system performance, although they are not specific to scalability. (When increasing the number of regions, we then analyse the performance of our system with respect to scalability and robustness). The average call success rate is a key measure since it provides information about what proportion of calls are successfully established. This is important since it tells us how many calls can be made and is therefore a measure of the end users' satisfaction. The average number of messages received per market server was measured in order to provide an engineering dimension so that network deployers can be given insight into how many messages would typically need to be processed per server, given different set ups. This helps them dimension their servers to support the anticipated load. Since nodes operate by battery power and are designed to consume as little power as possible, it is desirable for message processing to be distributed amongst several market servers. (As the *size* of a region increases, the load on a server would typically increase. However, as the *number* of regions increases, the multi-region handling overhead would increase too). By studying these two measures, we can observe a trade-off between attempting to satisfy end users of the system and the amount of processing per market server (see section 3.2.3). This trade-off can provide network deployers with an insight into how to dimension the network. (We acknowledge that other measures may also be used to evaluate the system and that they may be studied at a later date).

For robustness testing, a market failure was introduced in the network. The average call success rate (section 3.3.1) and messages received per market server (see section 3.3.2) were also measured for robustness testing. As with scalability testing, the average call success rate is an equally applicable measure to investigate for robustness testing, given a market failure. For the sake of consistency, we chose to investigate a *single* market failure throughout all of the robustness testing. Given that our goal is to understand the impact of regions in the presence of failures, we assume that all buyers have knowledge of the failed market server and, thus, we do not deal with failure detection. Finally, as with the scalability testing, a trade-off was looked at to find the optimum number of regions (section 3.3.3). For all experiments in sections 3.2 and 3.3, the network load was increased by varying the *call origination probability* (i.e. the probability of a call originating from any given unused handset).

For our experiments, several different network set ups were used. In each set up, the same underlying network topology was used but it was partitioned into a different number of regions with a market server in

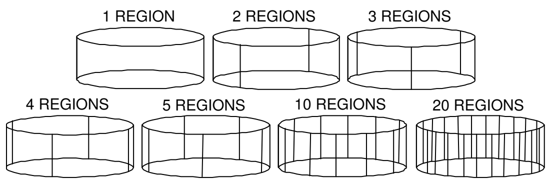

Figure 3. Set of seven tori used where each consists of an 80-node (20-by-4) network but is split into a different number of regions.

each region (market servers are placed manually within a central location in their regions where there is a high connectivity of neighbouring nodes). Thus, all of the experimental set ups use an 80-node (20-by-4) network. This was chosen because it demonstrates a topology that can be partitioned vertically with ease into several regions such that it is easier to evaluate the system for scalability. For each set up, the network was wrapped around and shaped into a torus, as shown in figure 3.[2] Also, all calls made were unidirectional (i.e. calls travel in only one direction around the torus). The combination of joining the beginning and end of the network to form a torus and allowing calls to traverse the network in one direction ensures fairness in our experiments for two reasons. Firstly, the number of types of different distance calls made within any set up are the same. For example, given the five region set up, there are exactly five different types of calls that require a single region, five different types that span across two regions, five types of triple region calls, and so on. Secondly, by using a torus, the average load in any region is the same, since there is no central region through which any extra calls traverse. This provides an even setting for testing the system.

The experimental settings we used in this evaluation were obtained from a domain expert. Specifically, each experiment was run for a total

[2] In addition to the torus topology, experiments for scalability analysis have also been run on several randomly generated network topologies where the results observed have shown similar broad trends to those outlined in this paper.

of 100,000 time steps. The duration of a call was set to 1,000 time steps. We assume that each node has 2 handsets attached to it and has a total of 10 units of bandwidth capacity available, allowing each node to relay up to 10 simultaneous calls at any one time. Calls were made to originate after every 50 time steps. Buyer agents were allowed to submit bids that contained a choice of 5 bundles from which an attempt was made to allocate the cheapest one available. Finally, the results presented in this paper are *statistically significant* with 95% confidence.

3.2. Scalability Testing

This section describes the experimental results obtained from carrying out the scalability testing outlined in section 3.1.

3.2.1. *Average Call Success Rate*

The purpose of the first experiment was to investigate the proportion of calls that could successfully be connected for each network set up, shown in figure 3, when varying the call origination probability. (The network is regarded as being heavily loaded when the call origination probability is 0.2 (20%), since this gives a typical network occupancy of 82%). As can be seen from figure 4, in general, call success rates are higher when there are fewer regions. (For the sake of clarity in figure 4, all call success rates are divided by the 1 region call success rate). Our interpretation of this is that as the number of regions increases, the multi-region handling that is performed also increases. For the multi-region set ups (i.e. when there are 2 or more regions in the network), resources need to be acquired across several regions. Here, the search for resource bundles is exhaustive across boundary nodes in intermediate regions — buyers will continue to bid until either a bundle is found or there are none available. As a result, when

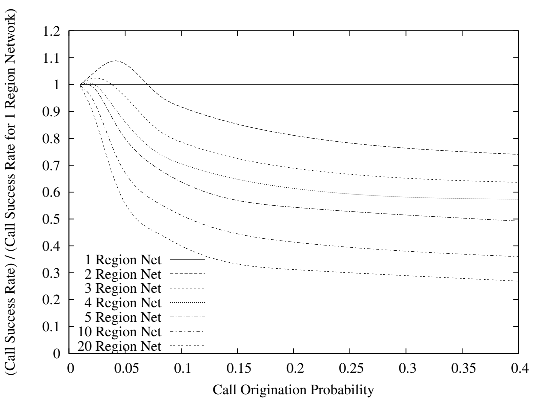

Figure 4. Average call success rate over the 1 region set up average call success rate, as the call origination probability is varied.

more searching for resource bundles takes place, resources are reserved for longer periods of time, while calls are being set up. Consequently, this reduces the chance of other calls being made.

However, another observation can be made from figure 4. Given a network load that is light to average where the call origination probability is below 0.07 (7%), we can see that the 2 region set up gives a higher call success rate than the single region case of up to 10%. Also, when the call origination probability is below 0.04 (4%), the 3 region network set up provides a higher average call success rate than the 1 region centralised case (by about 3%). This can be explained by the following. When the network is light to moderately loaded and there are sufficient resources available for a large number of calls, the additional searching that takes place (across boundary nodes) with the 2 and 3 region network set ups actually allows a higher chance for calls to be set up than with the single region case. The search for resources in the 1 region set up is similar to the search that takes place in the final region of a multi-region call (except that, on average, the required resource bundle is larger in the 1 region case because the size of the region is larger). Thus, no exhaustive searching takes place for resource bundles across boundary nodes for the

1 region case since there is only a single region and, therefore, if the initial attempt fails, the call set up fails.

However, a point is reached where actually having more regions (i.e. 4 or more) starts to give lower average call success rates than the 1 region set up, even at low call origination probabilities. Our understanding of why this occurs is that the additional multi-region handling required to process calls increases when using these network set ups. In these cases, the number of resources that are currently reserved during the search process is higher with more regions and this prevents other calls from being made. As a result, with 4 or more regions, the additional searching blocks more resources and, therefore, the average call success rate decreases. When the call origination probability increases, resource contention increases for all network set ups (i.e. node bandwidth becomes more scarce). For multi-region cases, more searching for resource bundles takes place, where resources are reserved for longer periods of time, while a call is being set up. Consequently, this reduces the chance of other calls being made. This is not an issue with the 1 region network and, therefore, it is only at this point that the single region network set up begins to perform better than all of the multi-region network set ups.

3.2.2. *Market Server Load*

Our next experiment investigates the number of messages received per market server, as the number of regions is scaled up. There are several different types of messages that are received by the market servers. These include buyer bids, seller asks, commit messages and messages that tell the market servers to release resources when the resource usage is complete. We do not differentiate between these, but rather simply sum them across all market servers and divide by the number of market servers, for each network set up, to find the average number of messages received by each

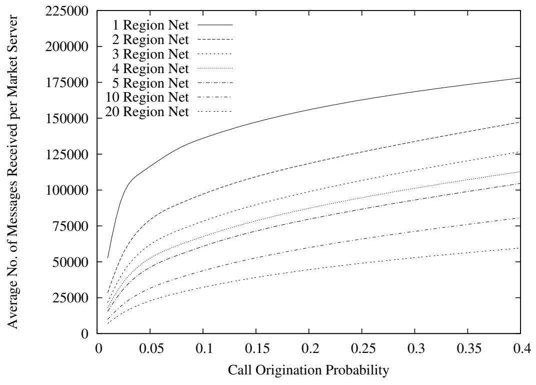

Figure 5. Average number of messages received per market server, as the call origination probability is varied.

one. Our hypothesis was that there would be a lower number of messages received by a market server, per simulation run, as the number of network regions increases. This is important because if market servers receive fewer messages then less processing needs to take place, less battery power is consumed, fewer input buffers are required and, therefore, the overall load on a server is less.

Figure 5 shows that our hypothesis is true and that when there are more market servers in the network, on average, the number of messages received per market server does indeed decrease. Using a 1 region network, with a call origination probability of 0.1, 136,000 messages are received by the single market server whereas, in contrast, with 2 regions there are 97,000 messages per server and with 20 regions, there are just 32,000 messages per server. A similar pattern is observed at all call origination probability values. The intuition for this result is that, as the number of regions increases, the number of nodes per region decreases since the overall size of the network remains fixed. Since buyers and sellers submit bids and asks respectively, only to their own local market servers, the number of such messages becomes less per market. Thus, with more regions, the number of messages received per market server is less (i.e.

the load is better distributed, which is desirable, since less processing is required per server).

Whilst the total number of messages received by *all* market servers in each network set up is higher with an increasing number of regions, this sum of messages does not increase by a disproportionate amount when, say, the number of regions doubles. For example, at a call origination probability of 0.04, there is only a 35% increase in the total number of messages received by all market servers from the 5 region network set up to the 10 region case. Also, at a lower call origination probability of 0.01, there is only an 8% increase in the total number of messages from the single region case to the 2 region network set up. However, it still remains the case that the average load per market server decreases with increasing number of regions and this is a useful result, since it is the processing *per* market server that we wish to minimise.

3.2.3. *Optimum Number of Network Regions*

Our result from figure 4 shows that, given specific call origination probabilities, the average call success rate is generally higher when the network is partitioned into a smaller number of regions. Alternatively, figure 5 showed that the number of messages received per market server decreases per market when there are more regions in the network. Thus, figure 4 (average call success rate) shows that less regions is better and figure 5 (messages received per market server) indicates that more regions gives a better performance.

Given these results, we would like to find the trade-off between the average call success rate and the number of messages received per market server. Our motivation is to help engineers in deploying such a network to find an optimum number of regions in which to partition the network. In order to achieve this, we first normalise the values for the average call

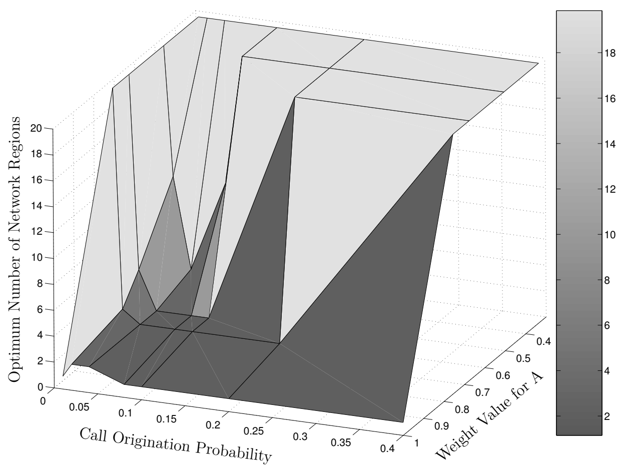

Figure 6. Optimum number of regions, as call origination probability and weight value for A are varied.

success rate and the messages received per market server, for each network set up, so that they both each range between 0 and 1, where we consider these as the utilities for each measurement (in the case of the latter, we subtract each normalised value from 1 to obtain the utility values, since receiving fewer messages per market server should give a higher utility). Then, for each of the network set ups and varying call origination probabilities, we consider the overall utility defined as the weighted sum of the utilities of the average call success rate and the number of messages received per market server: A $\times$ u(c) + B $\times$ u(m) such that u(c) is the utility of the average call success rate with weight A, u(m) is the utility of the number of messages received per market server with weight B and where A + B = 1. The values for A and B reflect the importance that a network designer would assign to the two different measures. Figure 6 shows a 3-dimensional surface plot where the optimum number of regions is plotted against the call origination probability with varying values of weights A and B. Several observations can be made from this figure.

When the network designer believes that receiving fewer messages per market server is more important than the average call success rate in the system (i.e. when B > A), the optimum number of regions to deploy is 20, across all call origination probabilities. This observation can be explained by the fact that a lower value for A means more importance is given to the number of messages received per market server, where the utility is higher when there are more regions.

When the network designer gives an equal importance to the average call success rate in the system and to receiving fewer messages per market server (i.e. when A = B = 0.5), the optimum number of regions that should be deployed decreases from 20 to 10, 3, 10 and then back to 20, given call origination probabilities of 0.01, 0.02, 0.04, 0.08 and 0.1, respectively. Thus, a minimum is seen across the optimum set of regions when A is 0.5. A similar pattern is observed when the importance given to average call success rate increases, thereby preserving the minimum. When a higher weighting is given to the average call success rate, the optimum number of regions drops further. The reason for this is that with intermediate call origination probabilities between 0.02 and 0.08, the difference in the average call success rates begins to widen with an increasing number of regions. This is also shown in figure 4 where the largest change in average call success rate occurs between these values of call origination probability. Thus, it becomes more evident that the fewer regions there are, the higher the average call success rate. In addition to this, because there is a higher weight value for A than there is for B, the optimum number of regions to deploy begins to decrease.

Finally, when exclusive importance is given by the network designer to the average call success rate (i.e. when A = 1 and B = 0), deploying a single region gives the best overall performance at all call origination probabilities except at 0.07 or below, where the 2 region network set up is best. Figure 6 shows that at these call origination probabilities, there is a maximum when the 2 region network provides the optimum solution. (Section 3.2.1 provided an explanation for why the 2 region network gave a higher average call success rate at these call origination probabilities).

3.3. Robustness Testing

This section provides the experimental results obtained from carrying out the robustness tests that were outlined in section 3.1.

3.3.1. *Average Call Success Rate with Failure*

The purpose of this experiment was to investigate how a single market failure in the network can affect the average call success rate when the number of regions is varied. Our hypothesis was that by introducing a single market server failure, the drop in the average call success rate would be higher when the number of regions is decreased, for any value of call origination probability. Figure 7 shows that this was indeed the case between 2 and 10 regions. (In figure 4, we divided the call success rates for each network set up by the 1 region case. This cannot be achieved in figure 7 because for the 1 region network, there are no calls set up since there is no market server functioning. So, for the sake of clarity in figure 7, all call success rates are divided by the 20 region call success rate. Thus, the slope of the curves in figure 7 follow a different pattern to the ones that were shown in figure 4). As the number of regions is scaled up between 2 and 10, the average call success rates increase, albeit at a progressively lower rate (i.e. the difference in the average call success rates between the 5 and 10 region set ups is very close when the call origination probability exceeds 0.12). When doubling the number of regions from 10 to 20, given a call origination probability below 0.06, the average call success rate

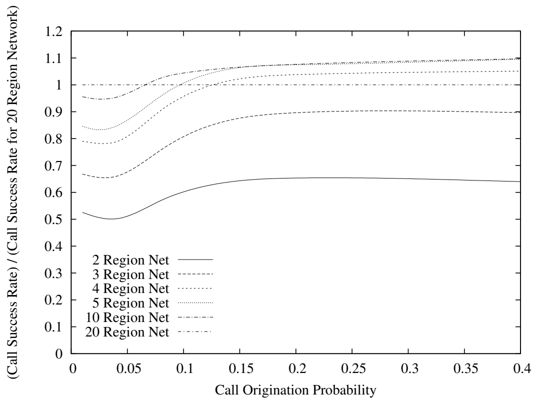

Figure 7. Average call success rate over the 20 region set up average call success rate, as the call origination probability is varied and where there is a single market failure.

increases. However, when the call origination probability exceeds 0.06, the call success rate decreases from the 10 region set up to the 20 region set up. We also notice that when the call origination probability exceeds 0.13, the 4 region set up performs better than the 20 region case and that beyond 0.1, the 5 region set up is also better than the 20 region set up.

These results can be explained by the following. When the number of regions increases and there is a *single* market server failure, the *maximum* percentage of calls that can take place increases, but at a progressively lower rate. For example, with a 2 region set up, up to 25% of calls can take place when there is a market server failure, a maximum of 40% of calls can take place with 5 regions and 45% with 10 regions. Thus, there is a steady increase in the call success rates between 2 to 10 regions, with the difference between 5 and 10 regions being marginal. We now explain how these percentages are obtained. In section 3.1, we outlined the reason why calls are unidirectional in our torus network. Regions are also assumed to be the same size in each network set up. These points ensure fairness such that the number of different distance calls originating are the same in each network. For example, on average, there should be the same number of single, double, triple, quadruple and five region calls attempted in the 5 region set up. Now given this, for the 2 region set up (regions A and B) without any market server failures, there are 4 different types of calls that can take place: A, B, AB and BA. If there is a single market server failure, for example in region A, only calls that originate, traverse and/or terminate in region B can possibly be set up. Since there is an equal probability of any of the 4 different types of call attempts being made, this means that the maximum percentage of calls that can take place in this example is 25% (1/4) since no calls can involve region A. In the 5 region set up, without any failures, there are 25 different types of calls that can exist, where there is an equal probability of each type occurring (e.g. 5 different single region calls (A, B, C, D, E), 5 double region calls (AB, BC, CD, DE, EA), and so on). Given a single market server failure, a maximum of 40% (10/25) of calls can take place. This principle applies to each network set up.

The following formula can be used to calculate the proportion of calls that can take place, given a single market server failure: $(n-1)/2n$, where n is the total number of regions originally present in a given network set up.[3] Thus, the percentages show that, given the network set ups in figure 3, the impact on the performance is affected more with a market

[3] This formula is derived in the following way for the unidirectional torus. If n is the total number of regions present in the network, then given a single market server failure, there are 0 types of calls that are n regions in length, 1 type of calls that are $(n-1)$ regions in length, 2 types of calls that are $(n-2)$ regions long, and so on, where each of these is non-negative. Therefore, we get $1 + ... + (n-2) + (n-1)$ kinds of calls that can take place. For the general case, where n is a positive integer greater than 1, this can be represented more formally as:

$$\sum_{i=1}^{n-1} (n-i) \quad = \quad \sum_{i=1}^{n-1} i \quad \text{which simplifies to} \quad ((n-1)n)/2 \ .$$

If we now wish to calculate the maximum rate of calls that can take place, we must divide by the total number of different types of calls that could be placed if there was no market server failure. Thus, we divide by n^2 since in an n region network, there are n different kinds of single region calls, n different kinds of double region calls, and so on. Therefore, we obtain the following:

$$\frac{((n-1)n)/2}{n^2} \quad = \quad (n-1)/2n \ .$$

failure, when there are fewer regions in the first instance. However, as already mentioned, the 4, 5 and 10 region set ups give better call success rates than the 20 region case at intermediate and heavier network loads. Our reading into this result suggests that when resources are more scarce, the excessive multi-region handling of the 20 region set up occupies resources for longer periods of time as compared to the 4, 5 and 10 region set ups. This is regardless of the maximum percentage of calls being higher for the 20 region case (47.5%), which is only apparent at lower call origination probabilities when resources are plentiful. Thus, figure 7 shows that at most network load levels, a network designer should ideally deploy a network which has between 5 and 10 regions if a market failure is anticipated.

3.3.2. *Market Server Load with Failure*

We now look at the load with respect to the number of messages received per market server, as the number of regions is scaled up, when there is a single market failure in the network. As discussed in section 3.2.2, we consider all of the different types of messages that are received by the market servers. Our hypothesis was that, even with a single market server failure, there would be a lower number of messages received by a market server, per simulation run, as the number of network regions increases.

Figure 8 shows that our hypothesis is true and that when the number of market servers is increased in the network then, on average, the number of messages received per market does actually decrease. There are no calls being made in the case of the 1 region network set up since the single server is assumed to have failed and therefore, no messages are received by this server. Using a 2 region network, with a call origination probability of 0.2, just under 64,000 messages are received by a market server whereas, in contrast, with 4 regions, there are 42,000 messages per server and with

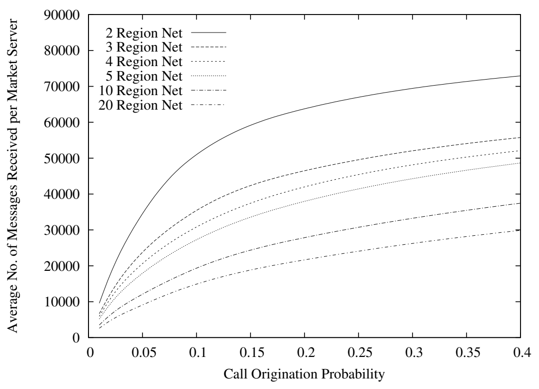

Figure 8. Average number of messages received per market server, as the call origination probability is varied and where there is a single market failure.

20 regions, there are just 21,500 messages per server. A similar pattern is seen at all call origination probability values, as the number of regions is scaled up.

Our analysis of this result indicates that this can be explained by the same trend that was described in section 3.2.2 when there is no market server failure. Inducing a single market failure in the network made no difference to the general pattern which showed that as the number of markets is increased, the average number of messages received per market server decreased. Thus, the benefits of distribution are maintained with respect to the message load per server.

3.3.3. *Optimum Number of Network Regions with Failure*

We now combine the results obtained from sections 3.3.1 and 3.3.2 in the same way as in section 3.2.3 in order to provide an insight into how many regions a network designer should use if such a network was to be deployed in practice, given a single market server failure. Figure 9 shows a 3-dimensional surface plot where the optimum number of regions is plotted against the call origination probability with varying values of weights A and B. Several observations can now be made.

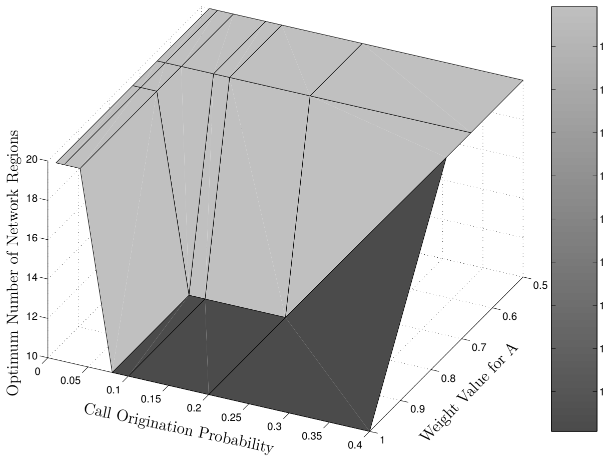

Figure 9. Optimum number of regions, as call origination probability and weight value for A are varied, where there is a single market failure.

When the network designer believes that receiving fewer messages per market server is more important than the average call success rate in the system (i.e. when B > A), the optimum number of regions to deploy is 20, across all call origination probabilities. Thus, a similar observation is made here as was noticed in section 3.2.3 when there is no market server failure. When the network designer gives an equal importance to the average call success rate in the system and to receiving fewer messages per market server (i.e. when A = B = 0.5) and also when the importance given to A is twice that given to B (i.e. when A = 0.67 and B = 0.33), the same trend continues where the optimum number of regions for deployment remains at 20, across all call origination probabilities.

A different trend is seen when the importance given to the average call success rate is increased to 3 times that of the number of messages received per market server (i.e. when A = 0.75 and B = 0.25). Here, we can see from figure 9 that the optimum number of regions is 20 when the call origination probabilities are relatively low (i.e. between 0.01 and 0.04) but that this changes when they are increased (i.e. between 0.04 to 0.2) where the optimum number of regions drops to 10 and then goes back to 20 beyond call origination probabilities of 0.2. Thus, a minimum is seen across the optimum set of regions when A is 0.75. The reason for this trend is that beyond a call origination probability of 0.06, the trend in the average call success rates changes where the 10 region network set up begins to give the best performance, only marginally beating the 5 region set up (see figure 7).

Finally, when exclusive importance is given by the network designer to the average call success rate (i.e. when A = 1 and B = 0), figure 9 shows that the 20 region case is the best network set up at low call origination probabilities (up to 0.04). However, at intermediate and higher call origination probabilities (above 0.06), figure 9 also shows that deploying the 10 region network gives the best overall performance. In fact the difference between using 10 and 5 regions is marginal when the call origination probability is above 0.12. Thus, at intermediate and higher network traffic loads, a network designer should consider using no more than 10 regions for network deployment. This trend can be explained simply by looking at figure 7 which shows that the 10 region network gives the highest average call success rate above a call origination probability of approximately 0.06.

3.3.4. *Call Distances*

During our investigation, we also discovered an additional result when looking at the 10 and 20 region set ups, in the absence and presence of failures. For these configurations, we found that the overall call success rate is marginally higher with a failure than when there is no failure. This can be explained by figure 10, which shows calls of different distances in the network for the 10 region set up, with and without failures. Short distance calls span across 1 and 3 regions, intermediate calls between 4

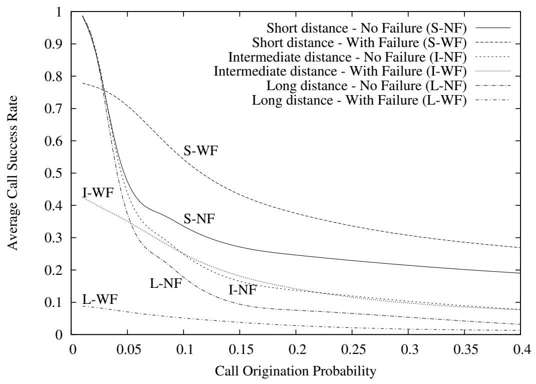

Figure 10. Average call success rate for different distance calls for the 10 region network, with and without a single market failure, as the call origination probability is varied.

and 7 regions and long distance calls cover 8 to 10 regions. The percentage of long distance calls for the failure case is less than without failures since, intuitively, our understanding suggests that the number of these calls is restricted more in the failure case (i.e. given the 10 region network set up in figure 3 with a single market server failure, no calls which span across all 10 regions can be made, for example). Thus, there are more resources available in the network for shorter distance calls to be more successful and indeed, figure 10 shows that the success rate of such calls is greater for the failure case beyond a call origination probability of 0.03. Finally, figure 10 also shows that the number of intermediate distance calls are roughly the same when the call origination probability exceeds 0.08.

4. Related Work

There are several market-based architectures that have been proposed for allocating resources in a distributed environment. In particular, (Gibney and Jennings, 1998) 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 (Wurman, Walsh and Wellman, 1998) with sealed bids and provided good utilisation of the network where the load was also balanced. However, a drawback of this system 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 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) is a framework for decentralised markets across the Internet (Rachlevsky-Reich et al., 1999). GEM has a single market which is distributed on which goods are sold. In GEM, the markets are replicated and the order for goods is distributed across these markets. 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 would induce more messages in the network than our partitioned approach (as was outlined in section 1.1).

MIDAS is an auction-based mechanism that allocates link bandwidth in a network for making paths (Courcoubetis, Dramitinos and Stamoulis, 2001). Simultaneous multi-unit Dutch auctions were used as the protocol for allocating resources. However, this protocol would be inadequate for our requirements since it is not capable of allocating several interrelated goods at the same time.

Finally, (Ezhilchelvan and Morgan, 2001) have looked at how an auction system can be distributed across several servers in a network of servers. However, their approach assumes that communication takes place using a high-bandwidth network which is an assumption that does not hold 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. Scalability testing was performed with respect to increasing the number of regions in a fixed size network. In conclusion, the results from the experiments conducted in this paper show that if light to average network loads are anticipated, then the network designer should consider deploying a decentralised network with a few regions rather than opting for a completely centralised system with a single market. We also see that the average number of messages received per market server is less as the number of regions is scaled up, and thus, allowing the processing of bids and the message load to be distributed better amongst the market servers when there are more of them.

In addition, for scalability analysis, we found the trade-off between the average call success rate and the number of messages received per market server, with respect to the optimum number of regions in which the network should be partitioned. Results showed that if the network designer assigns a higher priority to receiving fewer messages per market server than the average call success rate in the system, then the optimum number of regions to deploy is higher, at all values of call origination probability. When the level of importance for the average call success rate and the number of messages received per market server are considered to be the same by the designer, the optimum number of regions to deploy decreases progressively, at intermediate call origination probabilities, where a minimum is seen. If a network designer wishes to associate a greater importance to the average call success rate than the number of messages received per market server, the general trend that shows this minimum continues.

Robustness testing was also performed by inducing a single market server failure. This showed that at intermediate and higher values of call origination probability, the call success rate was higher as the number of regions increased until a maximum was reached, beyond which the call success rate decreased. This result was brought about by the fact that, given a market failure, there is a greater impact on the average call success rate when there are fewer regions in the network. We also saw that the number of messages received per market server decreased as the number of regions was scaled up, and that this trend follows the one where there is no failure in the network.

For robustness testing, we also looked at the trade-off between the call success rate and the number of messages received per market server. Here, we found that at most call origination probabilities and weight values, generally speaking, deploying more regions in the network was best. We also saw a minimum across the optimum number of regions when increasing the call origination probability, when the importance for the average call success rate was considerably greater than the importance assigned to the number of messages received per market server. However, even with this minimum, results suggested that generally speaking, when a market server failure is likely to occur, then deploying a higher number of regions is better than when there are too few regions.

Finally, during the course of our investigations, we also found that the average call success rate for the 10 and 20 region network set ups is slightly higher given a single market failure than when there is no market server failure in the network at all. This was explained by the fact that more longer distance calls were restricted in the failure case, thereby allowing many other shorter distance calls to take place with the remaining resources. This is a good result if a market server failure is likely to occur and if more shorter distance calls are envisaged being made. However, on the downside, the higher overall call success rate comes at the expense of fewer longer distance calls being made.

There are several investigations that we would like to look into for future work. Firstly, we plan to look at further robustness testing by introducing multiple market failures in the network to see what additional affect this has on the call success rate. Secondly, we aim to investigate the amount of control capacity used on the nodes in the network, since this will also be in contention. Performing such an investigation would provide a network designer with insight into how much control capacity nodes in the network should be deployed with in the first instance. Thirdly, we aim to impose a restriction on the number of messages that a market server can receive and process within a given amount of time. The purpose of this will be to see how well the system performs, with varying number of markets, when such a limit is imposed. Finally, we would also like to look into issues related to resource pricing in more detail in order to investigate the gains made by buyer and seller agents as a result of calls being set up.

Acknowledgements

This paper is a significantly revised and extended version of the work published in the First International Workshop on Incentive Based Computing *(IBC'05)* (Haque, Jennings and Moreau, 2005b). Also, the research in this paper is part of the EPSRC funded Mohican Project (Reference no: GR/R32697/01). Finally, we would like to acknowledge the contribution of Steve Braithwaite who provided us with domain expertise.

References

L. ChunLin and L. Layuan. A two level market model for resource allocation optimization in computational grid. In *CF '05: Proceedings of the 2nd conference on Computing frontiers*, pages 66–71, New York, NY, USA, 2005. ACM Press.

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

C. Courcoubetis, M. Dramitinos, and G. D. Stamoulis. An auction mechanism for bandwidth allocation over paths: New results. Technical report, London, UK, June 2001.

P. D. Ezhilchelvan and G. Morgan. A dependable distributed auction system: Architecture and an implementation framework. In *International Symposium on Autonomous Decentralized Systems*, pages 3–10, 2001.

M. A. Gibney and N. R. Jennings. Dynamic resource allocation by market-based routing in telecommunications networks. In S. Albayrak and F. J. Garijo, editors, *Intelligent Agents for Telecommunication Applications — Proceedings of the Second International Workshop on Intelligent Agents for Telecommunication (IATA'98)*, volume 1437, pages 102–117. Springer-Verlag: Heidelberg, Germany, 1998.

M. Goemans, L. E. Li, V. S. Mirrokni, and M. Thottan. Market sharing games applied to content distribution in ad-hoc networks. In *MobiHoc '04: Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing*, pages 55–66, New York, NY, USA, 2004. ACM Press.

N. Haque, N. R. Jennings, and L. Moreau. Resource allocation in communication networks using market-based agents. *International Journal of Knowledge Based Systems*, 18(4-5):163–170, August 2005a.

N. Haque, N. R. Jennings, and L. Moreau. Scalability and robustness of a market-based network resource allocation system. In D. Grosu and J. Shapiro, editors, *Proceedings of the First International Workshop on Incentive Based Computing (IBC05)*, pages 30–39. University of Technology of Compiegne, France, September 2005b.

T. M. Heikkinen. On congestion pricing in a wireless network. *Wireless Networks*, 8(4):347–354, 2002.

J. Hulaas, H. Stormer, and M. Schnhoff. Anaisoft: An agent-based architecture for distributed market-based workflow management. In *Proceedings of the Software Agents and Workflows for Systems Interoperability workshop of the Sixth International Conference on CSCW in Design*, June 2001.

P. W. Keller, F. Duguay, and D. Precup. Redagent-2003: An autonomous market-based supply-chain management agent. In *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1182–1189, Washington, DC, USA, 2004. IEEE Computer Society.

T. Kelly. Generalized knapsack solvers for multi-unit combinatorial auctions: Analysis and application to computational resource allocation. In P. Faratin and J. A.

Rodríguez-Aguilar, editors, *Agent-Mediated Electronic Commerce VI, Theories for and Engineering of Distributed Mechanisms and Systems AAMAS 2004 Workshop, Amec 2004, New York, NY, USA, July 19, 2004*, volume 3435 of *Lecture Notes in Computer Science*, pages 73–86. Springer, 2006. ISBN 3-540-29737-5.

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

L. L. Peterson and B. S. Davie. *Computer Networks: A Systems Approach*. Morgan Kaufmann Publishers Inc, San Francisco, California, 2003.

B. Rachlevsky-Reich, I. Ben-Shaul, N. T. Chan, A. W. Lo, and T. Poggio. GEM: A global electronic market system. *Information Systems*, 24(6):495–518, 1999.

O. Ratsimor, T. Finin, A. Joshi, and Y. Yesha. encentive: a framework for intelligent marketing in mobile peer-to-peer environments. In *ICEC '03: Proceedings of the 5th international conference on Electronic commerce*, pages 87–94, New York, NY, USA, 2003. ACM Press.

T. Sandholm, S. Suri, A. Gilpin, and D. Levine. Winner determination in combinatorial auction generalizations. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 69–76, Bologna, Italy, July 2002.

Y. Z. Wei, L. Moreau, and N. R. Jennings. A market-based approach to recommender systems. *ACM Transactions on Information Systems (TOIS)*, 23(3):227–266, 2005.

M. P. Wellman, J. K. MacKie-Mason, D. M. Reeves, and S. Swaminathan. Exploring bidding strategies for market-based scheduling. In *EC '03: Proceedings of the 4th ACM conference on Electronic commerce*, pages 115–124, New York, NY, USA, 2003. ACM Press.

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