Distributed Agent Architecture for Port Automation

Tom Thurston

Department of Computer Science
University of Essex, Colchester CO4 3SQ, U.K.
Email: tthurs@essex.ac.uk

Huosheng Hu

Department of Computer Science
University of Essex, Colchester CO4 3SQ, U.K.
Email: hhu@essex.ac.uk

Abstract

In the near future, container ports will no longer be able to expand into the surrounding land and will thus be unable to meet the storage requirements due to the boom in world trade. A solution to this problem is to increase the container throughput of the port by reducing the amount of time necessary to load and unload a ship. This paper presents distributed agent architecture to achieve this task. Under such architecture, an intelligent planning algorithm is continuously optimised by the dynamic and co-operative rescheduling of yard resources such as quay cranes and container vehicles.

1. Introduction

Since the container was introduced to the world trade thirty years ago, an ever increasing number of goods are being put into these containers, and loaded onto vast ships to be carried to their respective destinations throughout the world. When a ship arrives at a port, containers destined for this port must be unloaded and then new containers, bound for other ports must be loaded before the ship can resume its course. The demands on such ports to perform the loading and unloading process with maximum efficiency will become greater as the transport companies continue to increase both the size of their fleets and the capacity of the new ships added to them.

The problem with this increase is that port authorities around the world are predicting that they will run out of space to expand their operational areas and the option of developing on surrounding land is, unsurprisingly, often hampered by opposition from local residents. Therefore, the only remaining solution will be the reduction of the amount of time that a ship needs to remain in dock. The way to do this, is to ensure that the container unload / load process is done as rapidly as possible, and this can be achieved by ensuring that the resources necessary for this procedure are operating at an optimum level.

The Load/Unload Cycle: Ship-to-shore transferral of containers is performed by Quay Cranes (QCs) such as the ones shown in Figure 1. There are typically between one and four QCs assigned to a particular ship, depending on its size. Each crane will be in charge of several columns of container storage slots along the ship. Each column has several levels (depth) and thus sophisticated computer software has been developed to ensure that the containers are positioned such that they can be removed in a pre-calculated order so that space can be made for oncoming cargo with relative ease.[1] For each column, the QC must first unload all the containers destined for this port, and then load all the containers scheduled for leaving this port. Clearly there is a strict loading schedule which must be observed.[2]

This paper is concerned specifically with the loading process, however the standard unloading process is similar, just in reverse. In most ports around the world, the loading of each container requires 3 separate processes.

i) *Retrieval of Container from Stacking Lane* -- An efficient use of yard storage space is to stack the containers in a lane. These stacks may be four or five containers high and are typically 7 containers across. Each lane has a Rubber Tyred Gantry Crane (RTG), which can retrieve or handle these containers.
ii) *Transport of Container from Stack to Quay* -- Two main categories of vehicle will perform this operation.

Fig. 1. Quay Cranes at the Cosco HIT Terminal
(Thanks to Hutchinson Port Holdings for this)

[1] Due to accidental errors this process will not always run as smoothly as is hoped.
[2] Since failure to do so will create havoc at the ships next port, which is in all probability owned by the same Company as the current one.

One is a lorry or motorized platform, which will have to be loaded by the RTG in process 1. Another is a Straddle Carrier or other vehicle capable of self-loading, which is able to collect the container from the ground. The containers loaded onto these vehicles can then be transfer to the Quayside.

iii) *Transfer of Container from Quay Side to Sea Vessel* -- When the vehicle arrives at the target QC, depending on its type, it will either unload itself and clear out the way, or wait to be unloaded by the QC. Once the QC has picked up this container, it is loaded onto the ship into its designated storage slot.

These processes will then be repeated until every container in the current storage column has been loaded. At this point the QC will maneuver to the next column and the unload/load process will recommence.

Measuring Efficiency: A means of measuring how well a container port system is running is to monitor the usage of the QCs. The system will be running in an optimum state if all the QCs assigned to any ships in port are continuously performing a load or unload operation. In order for this to happen, each QC must be presented with either a constant supply of containers ready for loading, or vehicles capable of removing unloaded containers from the quay area.

This research is to devise a distributed system for the container handling process, which maximizes QC utilization and overall container to hectare ratio, whilst using a minimal amount of yard vehicles. In the rest of the paper, related work is briefly reviewed in section 2. The system design is presented in section 3, including a new yard layout and a hybrid hierarchical architecture. Section 4 describes the initial system implementation. Finally, conclusions and future work are given in section 5.

2. Related Work

The port automation is essentially the problem of formulating a plan P^i that involves transporting the resource *R* from location *A* to location *B* such that it arrives at time *T* and does not interfere with the plans P^{i-n} through to plan P^{i-1}. The methods for scheduling these resources generally fall into one of two distinct system architectures, namely centralized and distributed.

Centralized Systems -- A great deal of centralized systems have been proposed and implemented for use in Flexible Manufacturing Systems (FMS) for scheduling AGVs to transport materials around the factory floor. The task of the central agent is often to find the optimal schedule for its subordinates by creating search trees.

Pu and Huges have tested the effects of several heuristics on a STRIP – like planner in [1] and highlighted the potential use of heuristics in FMS style scheduling. Huang et al have designed a rule-based inference system using AND / OR graphs in conjunction with an iterative deepening A* search algorithm for rule selection in [2], which allows for dynamic changes in the schedule. Moro et al present Petri net based heuristics on top of an A* based search in [3], which produces optimum results with less effort compared to traditional A* implementations.

Machine learning techniques have been successfully deployed to optimize the scheduling procedure. Erkmen et al. uses genetic algorithms to tune a fuzzy logic based scheduler in [4]. Aydin and Oztemel have recently begun to explore the benefits of reinforcement learning in their scheduling agents [5]. ECT's Sea Land Terminal in Rotterdam uses a centralized system to control a fleet of some 50 Mannesmann Dematic AG vehicles that transport containers between the Stacking Cranes in the storage yards and cranes by the quayside. Ever at al. has created a simulation model for AGVs namely TRACES [6], which was adopted at Sea Land [7][8].

Distributed Systems -- Distributed systems have the problem that numerous agents may want to use the same resource at the same time. Ramos presents an architecture and a negotiation protocol to ensure successful scheduling in a dynamic FMS [9]. The architecture includes the use of resource agents representing the resources and task agents representing the tasks, thus adding a level of hierarchy to the agent society and ensuring that resources are not double booked.

Chen et al has devised a similar system and includes a protocol for resource allocation that is presented in [10]. Ouelhadj et al also uses resource agents [11][12] and bases their negotiation protocol on Reid Smith's Contract Net, a protocol which is explained in [13][14][15]. The Contract Net allows agents bidding for a job to act as managers and subcontract various components of the job to other agents in the system. Gu et al make use of the Contract Net for this very purpose in their bidding based scheduling system for FMS [16].

Lim and Zhang have built on the above approaches and in a prime example of classic Distributed Problem Solving [17] a framework that includes extra agents such as Optimization Agents and Bottlenecks Criteria Agents. It is also worth mentioning the work of Van Dyke Parunak [18], who was probably the first to start distributing manufacturing processes (again using the Contract Net) in YAMS (Yet Another FMS). Chia et al in [19] study agent behaviors in the Dis – ARM (Distributed Airport Resource Management) test bed. They identified the existence of several important behaviors regarding resource reservation including 'poaching', whereby the local reservation of a resource by agent '*a*' prevents agent '*b*' from reserving it, despite the fact that agent '*b*' requires it more critically in the global scale of the system.

The closest architecture to the one described in this paper is by Alami et al, who in their papers [20][21][22][23], describe the Plan Merging Paradigm (PMP), which has been proven to enable multi-robot co-operation in the MARTHA project. The PMP allows AGVs using shared resources, such as crossings, to successfully integrate their local plans with the global one. However, one drawback of this system, as they point out themselves in [23], is that the reservations are made on first come first served basis. A

further drawback is that reservations are only made one cell in advance.

3. System Design

3.1. Strategy

Since land expansion can not be accomplished horizontally, the next best alternative is to build upwards so just as cities have their skyscrapers, container ports have their stacking lanes. However, as the level of the stack is increased, so too will the chances of two separate QCs simultaneously requiring containers from the same stacking lane. This would inevitably cause the container transfer points of the stacking lanes to become major sites of bottlenecks while vehicles wait to collect their containers. Furthermore, the larger the number of containers stacked on top of each other, the larger the likelihood that the container which the RTG needs to access is underneath another container. This will thus increase the container retrieval time, hence making a bad situation worse.

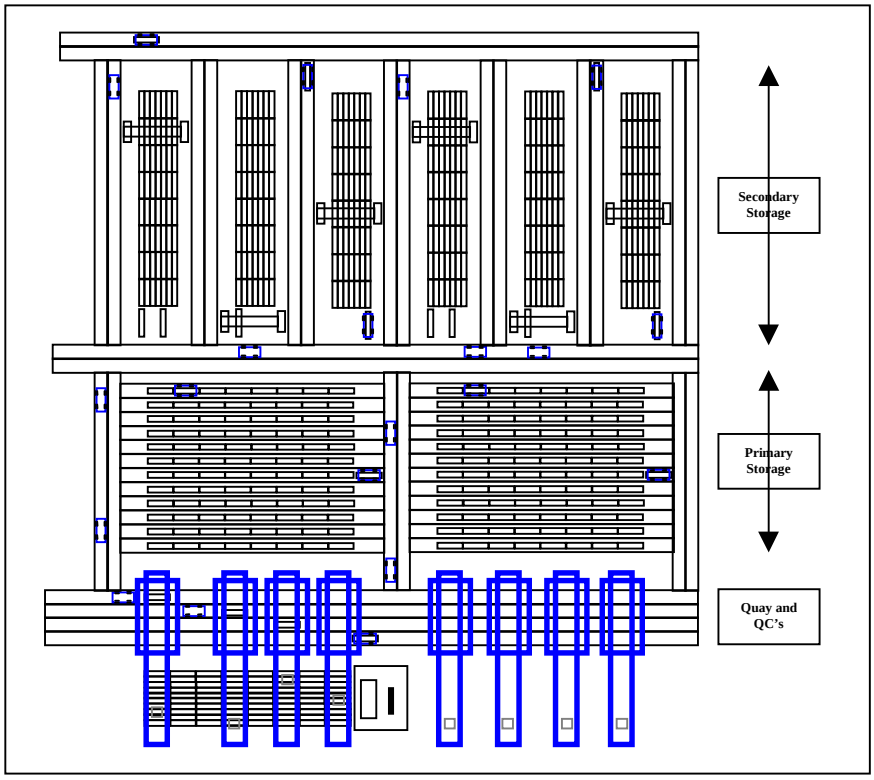

Fig. 2. A Section of a future Container Port layout which allows Straddle Carriers to perform Short Term Storage Stack to Quay Crane transport

If the loading schedules of ships are known well in advance of the ships arrival, (which they are), containers which are to be loaded in, say, the next 24 hours, can be moved from the high capacity storage areas away from the quay, and stored in a lower capacity area closer to the quayside. The advantage of lower capacity stacking lanes is that container retrieval can be faster. Indeed if the stack height and width can be reduced to 1 container, then a straddle carrier could enter such a lane and collect the container and drop it off at the target quay crane; without having to depend on the RTGs.

Figure 2 outlines how such a yard might look if storage space is divided into medium term storage, (secondary), and short-term storage, (primary). Depending on how far the loading schedule is known in advance, the system could even be expanded to include ultra high stacking lanes for long-term storage.

If such a system were to be implemented, remaining likely areas for bottleneck formation would be the commonly used shared resources, such as crossroads in the port highway infrastructure. Thus the proposed multiagent architecture seeks to enable the agents in the yard to maximize QC utilization by means of intelligent and informed planning made possible by the dynamic and co-operative rescheduling of resources.

3.2. Proposed Multiagent Architecture

As shown in Figure 3, the system will be controlled by four different types of agents, including the Quay Crane Agents (QCAs), Straddle Carrier Agents (SCAs) and the Traffic Agents (TAs), which will represent physical resources of the system. Each Quay Crane is controlled by a QCA, each Straddle Carrier is controlled by an SCA and each cell of the yard highway that contains more than one entry point, such as a crossing, is governed by a TA. The fourth type of agent is the Area Manager Agent (MA), it will oversee the initial assignment of jobs for any SCAs in the area it is in charge of.

3.3. Job Assignment

For each section of the ship, all containers destined for this port will be unloaded before any new containers are to be loaded. The loading schedule is predetermined and thus each QCA will have an ordered list of the ID numbers of the containers that it must load. The QCA sends a message to all the MAs in the system, requesting bids for the job of fetching a particular container and includes with it a Desired Time of Arrival (DTA). The DTA for container i is the Estimated Time of Arrival for container i^{-1} + the Estimated time required for loading container i^{-1}. Each MA will then perform calculations for every SCA in his area in order to determine their suitability for the requested job. This involves 3 stages.

Stage 1. Path plotting -- The MA will know the cell where an SCA will be when he has finished the job he is currently working on. It will plot the shortest path from this cell to the cell location of the target container in the

yard. Then it will plot the shortest path between this cell and the cell location of the target crane on the quayside.

Stage 2. ETA Prediction -- For each one of these paths, the MA will try to predict the arrival time at the target QCA. This will involve calculating how long it will take to cross each cell of the path, which in turn will depend on how fast the SCA is travelling when it leaves the previous cell, the level of congestion in this cell area and whether this cell is governed by a TA. If so, a tentative reservation for this resource must be made with that TA, specifying the estimated time it is required.

Stage 3. Tentative Reservations -- The TA will have a reservation list for all SCAs travelling through it. It will find a suitable window for the requested reservation and send a reply to the MA indicating how soon after the desired time, the tentative reservation is for.

Stages 2 and 3 will be repeated until the MA has a completed ETA value for each cell within the job path, and will thus know which SCA should arrive at the QCA closest to the time specified by the QCA. It then notifies the QCA of this time in its bid. Once a QCA has received bids from all its MAs it will notify them which plan contained the best bid. The contract winning MA will then pass this information onto the TAs who will delete all tentative reservations apart from those involving the winning SCA; these are necessary and will be turned into confirmed reservations.

Once this process is complete, the contract winning MA will send details of the job to the appropriate SCA who will append it to its job list. This process will then be repeated until the ship is ready to leave.

3.4. Why distribute the system like this?

The distribution is to achieve Just-In-Time production systems, whereby containers arrive at the quayside ready for loading at exactly the time they are needed.

Quay Crane Agents (QCAs) -- Each crane therefore has its own agenda, and thus it makes sense for each crane to be controlled by its own agent. In this way, malfunction of one crane should not impede the performance of a crane elsewhere in the system.

Straddle Carrier Agents (SCAs) -- They have been given with a degree of decision making. Each SCA attempts to register with a MA when it enters a new area. However, if it does not hear a reply from the new MA it can stay under the control of its old MA. If it has lost contact with its old MA, it can become its own manager! Thus in a worst case scenario, if there was a total failure in all MAs, the system would become entirely distributed and although this would mean an immense increase in wireless communications, it would nevertheless, remain fully operational.

Also, SCAs can make dynamic decisions regarding reservations. E.g. if SCA '*a*' is ahead of schedule and SCA '*b*' is behind schedule then SCA 'a' can give its reservation to SCA 'b', since this will be mutually

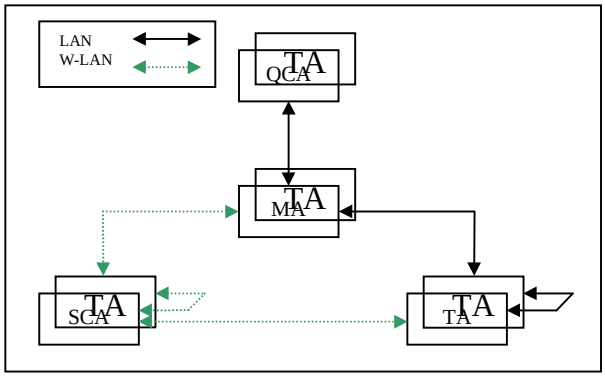

Fig. 3. Agent Infrastructure showing communications flow

beneficial. A further reason for providing the SCAs with planning abilities is to rapidly solve unforeseen circumstances. If an SCA comes across a broken down Straddle Carrier in the cell in its front, it just has to ask surrounding Straddle Carriers if they will be using the resources necessary to circumnavigate this obstacle, and if there is no objection, it can make an alteration to the journey path by perhaps travelling in the wrong lane for a short distance.

Traffic Agents (TAs) -- Clearly, dynamic re-planning by SCAs to avoid obstacles should be encouraged, but travelling in the wrong direction along the highway is not a step that should be taken lightly. In order to ensure that an altered plan will be risk free, an SCA can contact the TAs at either ends of the segment of the altered route involved to establish that no other vehicles are expected there. Since TAs know how many SCAs should be in each road segment at any particular time frame, they supply the MAs with statistics relating to congestion levels. This is important since the more agents using a resource, the more likely it will be that changes to ETAs will occur. If a route becomes really congested then a TA can suggest that the MA plot a path using a neighboring route.

Figure 4A shows a representation of how TA 'B' perceives the world. Figure 4B shows the crossing this agent governs and also that of a neighboring TA, identified as TA 'A'. It can apprise Manager Agents of the Congestion Factors for cells that form paths leading through its crossing for certain time periods. In this example, there are 2 Straddle Carriers which will be using the crossing in the time period between 10:05:00 and 10:05:30 and approaching from TA 'A'. The TA initially divides the number of vehicles using a route by the number of cells along that route. Thus, since 2/5=0.4 it can inform the MAs that the time taken to traverse any cell in the path between these two agents should be effected by a congestion factor of 0.4.

Should MAs find that these values are causing ETA's to be incorrect, they can inform that specific TA, who can then adjust the gain of the this Congestion Factor for the specific recorded congestion level, until satisfactory results are reported. However, the primary purpose of a

TA remains to ensure that 2 SCAs do not attempt to use a crossing at the same time. The SCAs will have low-level behaviours to prevent them from collisions but considering the fact that the purpose of advanced resource reservation is to enable the SCAs to travel through crossings at a maximum speed, some redundancy here is very useful.

Time	AGV ID	Previous Crossing
10:05:00	04	A
10:05:15	08	C
10:05:30	07	A

Fig. 4A. Reservation List for Traffic Agent 'B' from 10:05:00 to 10:05:30

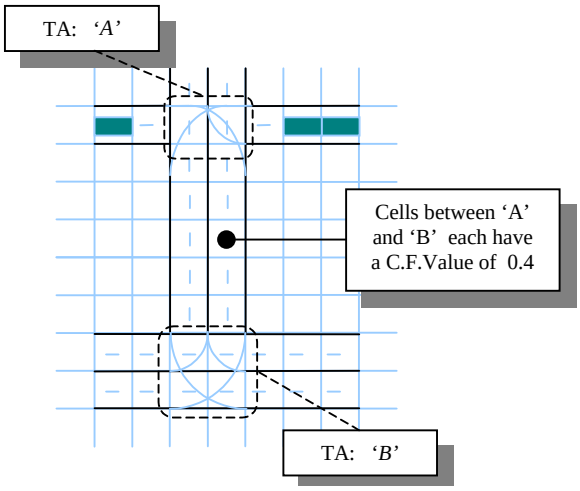

Fig. 4B. Traffic Agents Set the Congestion Factors for cells along the paths connected to them

Manager Agents (MAs) -- The purpose of the MA is to reduce the load on inter-agent wireless communications. A hundred SCAs all trying to communicate with TA's during the job bid process would really restrict the bandwidth available to SCAs communicating with each other and TAs regarding resources. By including MAs for each area of the port, initial job assignment can be entirely completed between agents using communications along a cable network with a high bandwidth. Clearly the processing load on an MA will depend on the number of SCAs in its area, which will in turn be dictated by the number of SCAs in the system. But as new vehicles are purchased, more managers can be added in order to ease the workload of the existing managers in the system. This makes the system both modular and readily expandable.

4. Implementation and Initial Results

The system has been prototyped in Java. First a yard map is created using the map editor, and then simulation runs can be created and viewed, using custom or randomly generated loading schedules. Statistics from the simulation run are then exported to Microsoft Access. All agents in the system inherit from a simple Agent Base Class we have created.

Agent Base Class -- The Agent Base Class gives each agent a unique identifier, an 'inbox' and the ability to read messages. These identifiers are used for inter-agent communication, and typically appear in the 'TO' and 'FROM' parts of messages. The IDN is assigned in order of creation, thus the 1st Straddle Carrier Agent created will have the ID (SCA, 1). The IDN 0 is reserved such that if the 'TO' field of a message contains the ID (SCA, 0), then it will be sent to all SCAs in the system.

The Agent Turn Machine -- Each agent in the system is stored in a vector. There is one vector for each agent type. The Agent Turn Machine aims to ensure that every agent has had a 'turn' every cycle, as shown in Figure 5. There is a timer that allows the user to decide the interval between each turn, and enables slow motion viewing of the real-time graphical output. The turn machine simply executes the 'takeTurn' method that each agent has inherited from the Agent Base Class, in order, from Agent 1, to Agent '*n*', for each Agent Type Vector.

```
public void agentTurnMachine() {
    .
    .
    .
    for( int i = 1; i < simData.managerVector.size(); i++) {
        (Agent) simData.managerVector.elementAt(i).takeTurn();
    }
    for( int i = 1; i < simData.carrierVector.size(); i++) {
        (Agent) simData.carrierVector.elementAt(i).takeTurn();
    }
    .
    .
    .
}
```

Fig. 5. Snippet of simple Agent Turn Machine Code

Taking a Turn -- Every time an agent takes a turn it will read the contents of its inbox and will react to each message it has received. Each agent is thus equipped with a special set of handlers that will enable it to react in an appropriate manner. The agents all have working memory to record their current mode of operation and any data relevant to their current planning status. In addition to this, SCAs and QCAs have Motion Operations to contend with.

The basic task of a MA is to keep up to date records of its agents, such as where and when they will finish their current jobs, and to process all the messages in its inbox each time he takes a turn. When a MA's mode is 'Calculating Plans', it will have to send and receive a lot of messages to and from the TAs. It will therefore take several cycles to gather all the data necessary for calculating all the ETAs for all its agents. Thus there are various sub-modes that a MA can be in whilst in this mode. Data associated with this task, primarily pointers to where abouts it was in each plan at the last cycle, are stored in working memory.

The features of a TA will cycle through various modes in relation to the job assignement process. Apart from performing duties relevant to these modes it will also process all messages received since its last cycle. Its general duty is to maintain both its Confirmed and Tentative Reservation Lists but assides from overseeing future reservations it must also ensure that Syncronisation Events are dealt with. Thus every cycle it checks that the owner of the top reservation in its list has clearance to travel through the resource it governs. When a SC has finished with a resource it sends notification to the governing TA so the TA can authorise other vehicles to use it. it was doing last cycle. Two typical examples of motion modes are *PickingUpContainer* and *WaitingForNextCellToClear*.

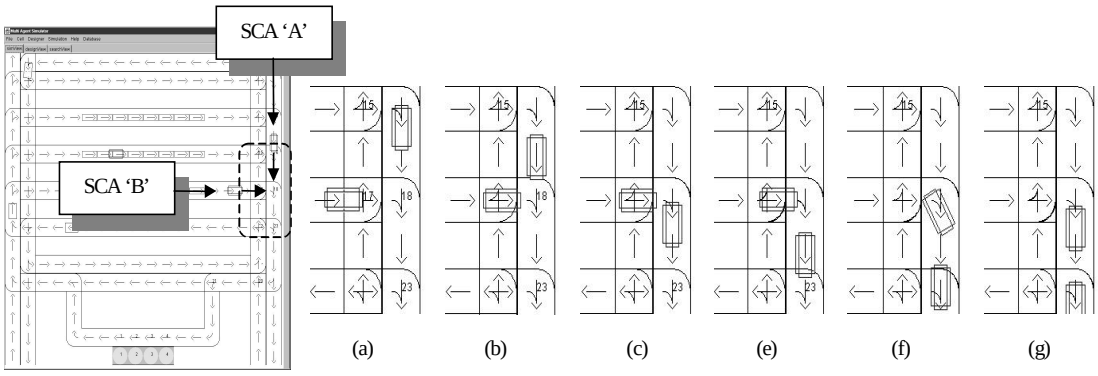

Fig. 6. Screen shots of Synchronization Events

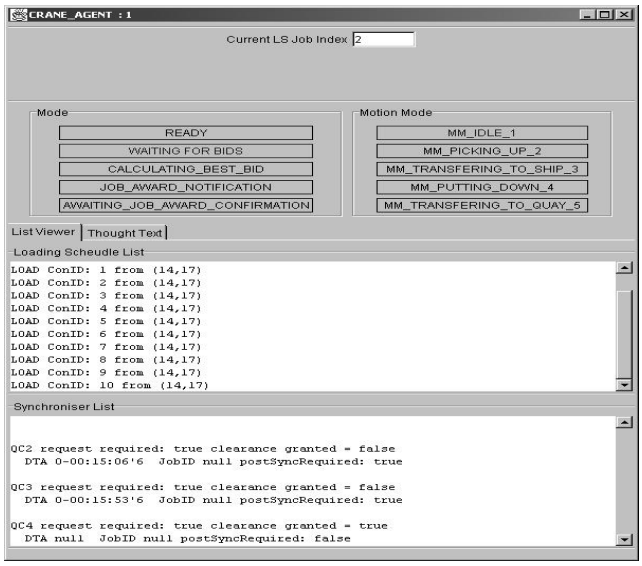

Fig. 7. The Loading Schedule Lists, Modes and QC Job Data

A QCA has a motion mode to contend with as well as a normal mode that deals with the job assignment process. The motion modes for a QCA are Idle, PickingUp Container, PuttingDownContainer, TransferToShip, and TransferToQuay. Initial testing with this implementation shows that advanced reservation of resources in conjunction with synchronisation events enables multiple SCAs and QCAs to utilise shared resources.

In addition to processing messages, SCAs have to control their own motion. This involves heading in the right direction according to their path plan and also performing functions such as picking up containers. Low-level behavious inside this agent prevent it from entering the next cell unless it is clear. Thus Motion Modes are required to help it remember what it was doing last cycle. Examples of motion modes are *PickingUpContainer* and *WaitingForNextCellToClear*.

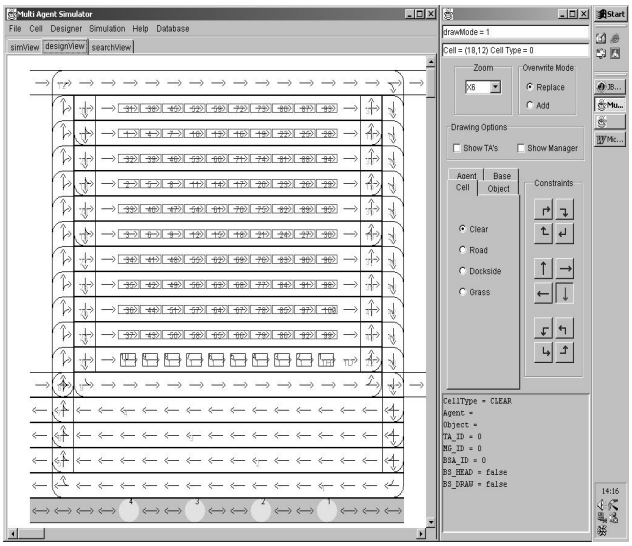

Fig. 8. A modular layout for a small yard with 4 small QCs

Figure 6 shows two SCAs heading towards the same cell. However, they do not crash because their MAs have arranged reservations for them with the TA governing this cell, well in advance of this moment. In Figure 6(b), the SCA 'A' travelling from North to South held the higher priority reservation and thus SCA 'B' travelling from West to East has had to stop. This is because it knows that before it enters this cell it must first receive clearance from the T. In Figure 6(e) SCA 'A' will have just sent a message to the TA governing the cell it has just left and cleared so that the TA can send a message to SCA 'B' who is now able to use the crossing. Alami et al has coined the term 'Synchronisation Events' for this activity [22].

Figure 7 presents a screen shot of the window showing the Loading Schedule Lists, Modes and QC Job Request Synchroniser Data. In contrast, a modular layout for a small yard with 4 small QCs (4 circles at the bottom) is shown in Figure 8.

5. Conclusions and future work

A distributed multiagent architecture for dockyard operation has been presented in the paper, which builds upon elements from both centralised and decentralised strategies. The system provides a feasible optimisation solution to the problem due to its inherent complexity.

The next stage of this research is to convert the prototype simulator into a multithreaded program in order to include representation of AGV speed control. Thus the purpose of advanced reservations will be realised in that the SCAs can slow down before entering a crossing, such that they arrive at it exactly on schedule, hence remaining continuously in motion, instead of having to stop and wait for their reservation. Once this has been achieved, data from simulation runs using a constraint satisfaction approach [24] will be compared with results from runs using a greedy reservation algorithm, intended to mimic the first come first served strategies used today.

Acknowledgements: This research is funded by EPSRC and GCS limited through a CASE studentship. Thanks to Dr Malcolm Roberts at GCS for his valuable support.

References:

[1]. P. Pu and J. Hughes, Integrating AGV Schedules in a Scheduling System for a Flexible Manufacturing Environment, Proc. of IEEE International Conference on Robotics and Automation, 1994, pp. 3149 - 3154.

[2]. T-S Huang, L-C Fu, and Y-Y Chen, Design and Analysis of a Dynamic Scheduler for a Flexible Assembly System, Proc. of IEEE International Conference on Robotics and Automation, New Mexico, April 1997, pp. 3334 - 3339.

[3]. A. R. Moro, H. Yu, G. Kelleher, Advanced Scheduling Methodologies for Flexible Manufacturing Systems using Petri Nets and Heuristic Search, Proc. of IEEE Int. Conf. on Robotics and Automation, 2000, pp. 2398 - 2403

[4] A. M. Erkmen, M. Erbudak, O. Anlugan, O. Unver, Genetically Tuned Fuzzy Scheduling for Flexible Manufacturing System, Proceedings of IEEE International Conference on Robotics and Automation, 1997.

[5] M. E. Aydin and E. Oztemel, Dynamic job-shop scheduling using reinforcement learning agents, International Journal of Robotics and Autonomous Systems, Vol. 33, 2000, pp. 169 – 178.

[6] J.J.M. Evers, D.G. Lindeijer, L. Loeve, J.A. Ottjes, "Agile Logistic Systems: the case of a container terminal", TRAIL Research School, Delft, 1998.

[7]. M.B. Duinkerken, J.J.M. Evers, J.A. Ottjes, TRACES: Traffic Control Engineering System. A case study on container terminal automation. Proc. of the Summer Computer Simulation Conf., July 1999. Chicago [SCS].

[8]. M.B. Duinkerken and J.A. Ottjes, A simulation model for automated container terminals, Proc. of the Business and Industry Simulation Symp., Washington D.C., April 2000.

[9]. C. Ramos, An Architecture and a Negotiation Protocol for the Dynamic Scheduling of Manufacturing Systems, Proc. of the IEEE International Conference on Robotics & Automation 1994, pp. 3161 - 3166

[10]. Y-Y. Chen, L-C Fu and Y-C Chen, Multi-agent Based Dynamic Scheduling for a Flexible Assembly System, Proceedings of IEEE Int. Conf. on Robotics and Automation, Leuven, Belgium, 1998, pp. 2122 - 2127

[11]. D. Ouelhadj, C. Hanachi, B. Bouzouia, A. Moualek and A Farhi, A Multi - Contract Net Protocol for Dynamic Scheduling in Flexible Manufacturing Systems, Proc. of IEEE Int. Conf. on Rob. & Auto., 1999, pp. 1114 - 1119

[12]. D. Ouelhadj, C. Hanachi and B. Bouzouia, Multi-Agent System for Dynamic Scheduling and Control in Manufacturing Cells, Proceedings of the 1998 IEEE Int. Conference on Robotics & Automation, Leuven, Belgium - May 1998, pp. 2128 - 2133

[13]. R. Davis and R. G. Smith, Negotiation as a Metaphor for Distributed Problem Solving, Artificial Intelligence, Vol. 20 No. 1, January 1983, pp. 63 - 109.

[14]. R.G. Smith, A Framework for Distributed Problem Solving'. In Proceedings of IJCAI'79, 1979

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

[16]. P. Gu, S. Balasubramanian and D. H. Norrie, Bidding-Based Process Planning and Scheduling in a multi-agent System, Computers and Engineering, No 2, Elsevier Science, 1997, pp. 477-496

[17]. M. K. Lim and Z. Zhang, A Framework for the application of autonomous agents in integrated dynamic process planning and scheduling system, Proceedings of the IASTED Conference on Robotics and Applications, Santa Barbara CA, 1999, pp. 47 - 51

[18]. H. Van Dyke Parunak, Manufacturing experiences with the contract net. In Michael N. Huns, editor, Distributed AI, Research Notes in AI, Pitman 1987, pp. 285 - 310.

[19]. M.H. Chia, D.E. Neiman and V.R. Lesser, Coordinating Asynchronous Agent Activities in a Distributed Scheduling System, Proceedings of the 3rd International Conference on Multi-Agent Systems, 1998

[20]. L. Aguilar, R. Alami, S. Fleury, M. Herrb F. Ingrand, F. Robert, Ten Autonomous Mobile Robots (and even more) in a Route Network Like Environment, Proc. IROS, Pittsburgh, USA, 1995, pp. 260 - 267

[21]. R. Alami, R. Chatila, S. Fleury, M. Ghallab, F. Ingrand, An Architecture for Autonomy, Int. Journal of Robotics Research 1998, Vol. 17, No. 4, pp. 315-337

[22]. R. Alami, S. Fleury, M. Herrb, F.Ingrand, S. Qutub, Operating a Large Fleet of Mobile Robots using the Plan-Merging Paradigm, Proceedings of the IEEE Int. Conference on Robotics and Automation, Alburquerque, New Mexico, 1997, pp. 2312 - 2317

[23]. R. Alami, S. Fleury, M. Herrb, F. Ingrand, F. Robert, Multi-robot Cooperation in the MARTHA Project, IEEE Robotics and Automation Magazine, 1996, pp. 36 – 47

[24]. E.P.K. Tsang, Foundations of constraint satisfaction, Academic Press, London, 1993