Simulating Software Agent Colonies in Large Scale Distributed Artificial Intelligence (DAI) Networks

C.W. Lee, J. Ryoo, S. W. Tak, D.H. Lee, C.G. Oh, J.W. Kim, J. Stach, and E. K. Park
Computer Science Telecommunications Program
University of Missouri-Kansas City
5100 Rockhill Road, Kansas City MO, 64110
{stach, ekpark}@cstp.umkc.edu

ABSTRACT

Wide area networks have a large number of computers capable of providing a wide range of services. Due to their low cost and high performance, autonomous mobile Agents are being used to access information sources in distributed, wide-area, networks. Architectures such as active networks also employ intelligent, mobile network processes. However, there are no analytic models for the performance of such systems and as a consequence, simulation is required to test the efficacy of their performance and policies. Assessment of performance and policy requires visualization techniques that depict the state of the Agent colony and its demand on network resources. We sketch the design for a large scale, DAI simulation and analyze its implementation with respect to time and resource requirements. An Agent colony is simulated using C^{++} over CSIM. C^{++} over CSIM achieves the clarity and logic of Agent based simulation for network services, as well as ease of modeling and reuse of model entities. We also present a Java based builder to create the simulation objects under Object Orientation, and a Visualization post processor to alleviate the inherent difficulty of visualizing a Distributed Artificial Intelligence (DAI) system. Decomposition is applied to partition the search domain into an Agent-oriented view and a Network-oriented view. The individual views are then abstracted to provide summary views to the user, who is given control over the degree of abstraction.

Keywords
Intelligent Agent, Distributed Artificial Intelligence, Visualization, Distortion-Oriented Visualization, CSIM

1. Introduction

Distributed computing systems not only provide the facility for utilizing remote resources, but increase throughput by concurrent processing. In order to increase throughput, resource utilization and communication costs must be balanced. To balance utilization, we allow software Agents to migrate to desirable locations in the network to perform computation according to some attributed preference structure. To minimize communication costs, process migration must be a rational function of the mobile process. Since the software Agents perform non-deterministic user computations, Agent migration is rooted in the Task Assignment problem of distributed processing. Approaches to task assignment fall into three categories: graph theoretic, mathematical programming and heuristic methods. Graph theoretic methods use a process graph to represent a task, and apply a minimum-cut algorithm to the graph to get a process assignment with minimum inter-processor communication. The mathematical programming approach formulates process assignment as an optimization

problem and solves it by mathematical programming techniques. The heuristic method provides fast but sub optimal algorithms for process assignment that are useful for applications where an optimal solution cannot be obtained in real time[1]. Our approach to task assignment is a variant of the graph theoretic approach proposed by Stone[2]. By careful definition of characteristics of the network and the workflow of information Agents, we address the problem of distributing autonomous, mobile Agents. Assessing the performance of these Agents as a society requires special simulation methods due to the large number of simulation entities. There are several important features of any large scale simulation: abstraction, emulation, scenario generation, visualization, and extensibility[3, 4, 5]. $CSIM^{++}$ is a process-oriented, object-based language appropriate to the simulation of mobile Agents, but does not include scenario generation methods or visualization tools necessary for performance analysis. We therefore augmented $CSIM^{++}$ with a Java based Builder program to set the network parameters and create simulation entities. Figure 1 is an overview of our simulation package.

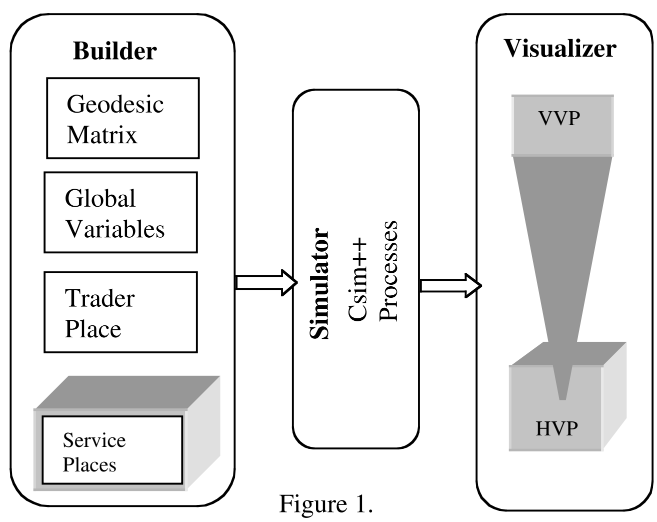

Figure 1.
Overview of Simulator

Visualization techniques for viewing the behavior of a colony of mobile Agents in a wide area network environment present significant challenges because of the number of network entities hosting the colony. The most common technique for visualizing wide area networks involves node and link diagrams. This approach has problems when the size of the network is large. Eick[6] points out that display clutter, node positioning, and perceptual tension are major limitations of traditional node and link diagrams. The problem of display clutter is exacerbated when software Agents are present in the network. Therefore, scalability of visualization is heavily dependent upon the decision of how to represent network entities. We directly address that problem in this paper.

The visualization decision on how to represent Agents and entities must avoid cognitive overload. Miller has shown that the number of items that a human can store in short term memory is limited[7]. In our work, multiple orthogonal views are used to ease cognitive overload. Two views employed in our visualization are the Agent-oriented view and network-oriented view. In each view, only the major aspects of a particular element or context are displayed, with the remainder of the view de-emphasized and simplified. The details available to the two views are abstracted to a level where they can be represented commonly, through some orthogonal relationship. For example, both network and nodal congestion are related at some degree of abstraction by traffic intensity. At this level or view, equal emphasis can be given to the constituents of performance when viewed as a single metric. Nonetheless, innovative methods of display are required to display conditions over an arbitrarily large number of elements. Consequently, we developed new methods consistent with the established guidelines for visualizing abstract data[8]. While partitioning the visualization task in terms of multiple views significantly decreases the complexity of information to be visualized, the amount of data to be represented in each view is still overwhelming. We therefore employ summary views as a navigation framework for other more concrete views, extending from a higher category[8]. In our methods, data point reduction is achieved using a technique we developed called Horizontal View Partitioning (HVP). A further division of view by data type is achieved using traditional Vertical View Partitioning (VVP)[8]. VVP is implemented using distortion-oriented visualization and filtering.

2. Simulation Requirements

Certain knowledge about the network and task flow is required in order for an Agent to select a location for service execution in a network. Some notion of topology is required, but not all its details, as well as some sort of task signature for the non-deterministic workflow the Agent must execute on behalf of a user. A geodesic is defined as the shortest path between two nodes in a network. The n *x* n dimensions of geodesic matrix correspond to the number of hops on the shortest path between nodes in the network. The diameter of the network is its largest geodesic. The network in our test system accommodates 20 nodes with bi-directional links. The maximum link speed is 1.5 Mbps. In the initial release of the system, link speeds are assumed constant throughout the network. Link queuing delay is not considered as it does not dominate the performance problem. Transmission time however, is considered, because it can be substantial depending on the payload of the Agent and the link speed. Nodal entities, modeled as formal objects, consist of Agents, Service Planners and Traders. Service Planners reside at each node and compute an Agent migration based upon network conditions and the Agent's attributed preference structure. A single Trader object advertises the services offered at each Service Place (node). Service Places may offer a maximum of ten services to a maximum population of 10^3 software Agents. The Agent preference structure is applied to the Directory advertisements to allow software Agents to select a Service Place based on both functional and qualitative attributes[9]. The qualitative attributes include Service Quality, Cost of Service and Time to Service. The functional requirements match the next required service to a set of nodes offering that service. As the software Agent arrives at a node, it enqueues to the Service Planner and may wait an unlimited time for execution. Based on the link speed, arrival rate, service rate, network configuration and Agent payload, local congestion will be dynamically created at network nodes when Agents enter the network system and begin to migrate. Local congestion, whose unit is the number of Agents enqueued to the Service Place, is an important factor in selecting the next move. Each service (task) instance has attributes such as payload and service time. The objective of the simulation is to provide a pareto optimal choice of nodes to accommodate the process signature of an Agent such that the amount of time an Agent spends in the network system is minimal, subject to its preference structure.

2.1. Encoding An Agent s Workflow

The signature of the software Agent s workflow is encoded in basic process algebra. The workflow signature is a series of terms corresponding to services joined by operators for sequencing, simple alternation, or concurrency. The operators are (+) alternation, (;) sequencing and (||) concurrency. The signature a;b;(c||d+e);f defines a sequence executing Service a followed by Service b, followed by Service c and Service d concurrently, or followed by Service e, concluding with the final Service f. The choice of (c||d) or (e) is considered to be non deterministic, i.e. determined at the completion of the execution of b. When the alternations of series are mapped to Service Places, a migration path through the network is formed. In our initial model, service signatures for the non-trace portion of the Agent population are generated randomly. It is not intended that these signatures be meaningful or correspond to a real world processing sequence. Their purpose is to generate dynamic system behavior. Service times for Services are generated randomly from an exponential distribution using the global attribute Mean_Service_Time as its mean. For those Agents of interest in the population (trace set), the signature is individually specified by the modeler, as is the Agent s position in the birth sequence for the entire population. These signatures can correspond to real world workflows. Post processing visualization must display network performance, population statistics and the statistics of the Agents in the trace set. Individual Agent performance is a consequence of dynamic interaction with the software Agent population encountered during a signature execution.

2.2. The Move/Stay Decision

The feasibility of Agent migration (the move or stay decision) is based upon comparing a dynamic lower bound and the minimum migration time to a Service Place offering the desired service, assuming that service is also offered at the Agent's current location. If the Agent does not migrate, it must enqueue again to the Service Place at its current node, thereby incurring a delay equal to the local congestion on that node, but avoiding the time penalty of migration. If the Agent migrates, it must incur a penalty (geodesic*payload/ link speed) of the transmission time to the closest node offering the service plus the minimum local congestion of any node offering the service in the network. If the local congestion at the Agent s node is less than this lower bound and the desired service is offered at the current Service Place, there is no reason to incur the computational expense of the Service Planner. The Agent simply enqueues again to its local Service Place, requesting the desired service. Otherwise, the Agent requests the Service Planner to compute the next optimal migration for its signature in a manner that minimizes its total, predicted life in the network, subject to its preference structure. If the decision is made to move and if the next required service is offered on more than one node, a minmax algorithm is run to select the pareto optimal node to visit based on Time to Service(TTS), Quality of Service(QOS) and Cost of Service(COS). Our method extends Stones' graph theoretic approach that performs a minimum cut over a graph whose interior edges are weighted by Inter Process Communication Costs (IPC) and whose exterior edges are weighted with task Execution costs (E). In distributed processing, this model is referred to as (E+IPC). In our research, we represent the interior nodes as services and the exterior nodes as Service Places offering those services. A graph of the interior nodes is constructed according to the Agent's service signature. a;b specifies a single, unidirectional edge between services a and b. a;b+c specifies outgoing edges from a to b and a to c respectively. a; b||c specifies an outgoing edge from a to supernode (bc). The edges between these services carry the expected payload of their transitions. In the case of b||c, the edge will carry the maximum payload transfer of ((a,b),(a,c)). Each exterior edge maps an interior service node to the Service Places supporting that service. The weight of the service-Service Place edge is the cost computed from congestion, quality and cost attributes of the Service Place according to a multi-attribute programming solution. Each Service Place in the set mapped to the next service being planned is considered as the root of a graph. A minimum cut algorithm is applied which results in a set of services being assigned to some Service Place. More than one service may be assigned each Service Place when the minimum cut is applied. When the service being planned is assigned, it is removed from the graph so that it does not have multiple assignments. When all the services in the rooted signature have been assigned, a minimal migration for the signature is returned from the Service Planner to the Agent. The Agent migrates to the selected node, enqueues to the Service Place, executes the service, enqueues to the Service Planner process in that node and repeats the process described above.

2.3 Simulation Objects

Major simulation objects are the Geodesic matrix, Global Variables object, Directory object and Service Place objects. These define the network and network environment in which the Agent colony resides. The Agent colony is auto-populated with objects of random signature in the simulator, except for a specific trace population whose individual and collective performance is to be assessed in the dynamic system created by auto-population. The trace population objects, which are the subject set of the simulation, are defined by the user and generated as individual, named objects. Global parameters include a number of network attributes: number of Service Places, number of Traders, number Agents, mean inter-arrival time, mean service time, mean payload size, grain size for the data sampling interval, link speed, maximum signature size, maximum task alternatives, random seed and names of Agents in the trace population. The Geodesic object provides a symmetric matrix of shortest paths between Service Places. The Trader object advertises the attributes of the Service Places (nodes) in the network including available services, local congestion, QOS and COS. Local congestion is updated dynamically during the simulation according to a Trader update parameter. The update policy in the initial implementation is a fixed interval for all Service Places. There is no random start.

3. Visualization Requirements

The objects of information visualization in networking are fairly well identified in papers such as [6, 10, 11, 12, 13]. Significant visualization data items include network nodes, links, and their attributes. While the design aspects of visualization in DAI systems have been explored [14], few researchers have worked on visualizing network DAI systems. In our work, we try to build on established network visualization techniques in an attempt to present the dynamic operations of a DAI system. Initially, we tried to superimpose the Agent colony activities over conventional network visualization. That result was not satisfactory. As an alternative, we developed a powerful data abstraction technique referred to as HVP. In visualizing data items, HVP increases the abstraction level of de-emphasized data, while presenting concrete details of the data emphasized by a specific view. Data items are partitioned into Agent-oriented and Network-oriented views according to what view is dominant at the moment.

3.1 Data Items in the Agent-Oriented View

The focus of the Agent-oriented view is the status (or condition) and activity information of a single Agent or a group of Agents. Given the fact that Agents are mobile, and that the network provides a distributed environment, visualizing the migration pattern of an Agent or multiple Agents is critical in this view. Considering that there could be thousands of Agents active in a network, it is not feasible to display the migration traces of every mobile process in the Agent-oriented view. The display space would be so cluttered that no meaningful presentation of migrating is possible. It is more viable to show an individual Agent moving from one network node to another leaving an apparent trace. Here, the information with respect to the network is minimal since the view concentrates on the Agent. Except for topological information, network data such as node and link statistics are abstracted away. Some data items that are important to show are related to the current status of an Agent. Mobility provides one criteria of Agent status. At anytime, Agents are either migrating or situated. Waiting and running are the possible states when an Agent is situated. The running state can be further categorized into planning or processing. A second category of Agent status is its Agent's progress toward goal, i.e. how much of its signature is completed. The health of an Agent is the last criterion of Agent status visualization. Agent health can be defined in many different ways, but one possible approach is assessing an Agent's behavioral integrity by keeping track of Agent generated events to detect misbehavior or access violation[15]. Once the criteria for Agent status are established, each category can be separately encoded into disparate visual elements. In our prototype when Agents are migrating, an Agent symbol is shown above the node portion of network topology. Also, the Agent symbol transforms itself into different shapes to represent its different states. Our implementation uses a cartoon figure to represent the current status of an Agent. The icon is arbitrary. Finally, the behavioral health of an Agent is visualized by changing the color of an Agent symbol. Figure 2 is a snapshot of the prototype HVP Agent-oriented view. Some additional information such as an Agent s name, passage of time, and migration path are displayed at the left lower portion of the figure. The start button at the bottom initiates the visualization process.

3.2 HVP in DAI Visualization

As mentioned above, the Agent colony resides in a dynamic system environment described by the attributes of the network. Individual Agent worlds are specified by the service offerings, service attributes and local congestion of the node at which they are situated. In the Network-oriented view, the information related to Agents is suppressed. They merely represent arrivals and routed traffic. Agent-related data are implicit and presented only through statistics provided by the network. For instance, the collective migration pattern of Agents can be indirectly obtained from examining the average Agent age, local congestion, and nodal utilization. The major data items in the Network-oriented view are the attributes of network nodes and links. The possible current states of nodes and links are operational or failed. Operational node and link states can be further analyzed. However, not all these statistical values are relevant to DAI visualization. Typically, the data items demonstrating the effect of the Agents in a DAI system are performance-centric. Performance-centric network status data include the amount of traffic between two nodes, local congestion and utilization of a network node to name a few.

Knowing what data item to visualize does not fully address the question of what techniques to use for DAI visualization. When collecting performance-centric network statistics, one needs to understand that the data is corrupted by data sources that have little to do with the Agent activities. It cannot always be guaranteed that Agents are the dominant force affecting the performance statistics of the network. Therefore, one essential function of a DAI visualization system is a capability to filter the net effect generated by the Agents of interest from the raw data of network sources.

Once the needed data is available, one can address the problem of how to represent the Network-oriented view data in the most efficient way. Many different, well-established, network visualization approaches can be adopted. Since the objective of a Network-oriented view is to observe what kind of effect rational software Agents have on their environment (the network), it is necessary for the visual interface to provide users with a means to isolate and display affected data items according to their source. Another requirement for an effective DAI visualization is the temporal projection of data. Most network-related data in a DAI system have semantic interpretation only when displayed in the context of time. Timelines are a linear visualization of events[16], and this technique can be applied to the visualization of the Network-oriented view of a DAI system. In fact, for the purpose of implementing the Network-oriented view, plotting data points with interpolation is one of the most straightforward and intuitive ways of visualization. An example of such implementation is shown in Figure 3. In this figure, network node utilization is displayed in a relational manner that abstracts the details of their utilization from its form. We see in figure 3 that the load of the network has maintained an approximate balance but we cannot tell if individual nodes are under utilized or congested. Initially, local congestion increases dramatically when Agents enter the network. Some of the nodes, particularly SP1, display markedly higher utilization than others. We can infer that the rational migration policy of individual Agents has the effect of leveling utilization over the network nodes because after what appears to be an initial congested period, utilization becomes approximately equal across all nodes displayed. Other statistics are also accessible in a temporal context by using the drop-down box at the bottom. The Network-oriented view can be further partitioned to accommodate the need for analyzing related data items collectively and for the inspection of a single data.

3.3 VVP In DAI Visualization

Figure 4 gives the detailed performance of SP1, one of the congested nodes in figure 3, over the same time frame using VVP or drill down. In this figure, the actual utilization for the congested node is displayed at each time grain. From figure 4 we see that SP1was not congested, merely higher utilized than other nodes in the region. Examining a series of these graphs will allow the analyst to determine how well the load has been distributed and if the congestion points are time coincident. More importantly, VVP allows the analyst to determine whether the relatively higher utilization of SP1 is a consequence of arrivals (i.e. its service offering) or a relative number of arrivals whose mean service time is greater than other less utilized nodes.

Reducing the amount of data that a user needs to comprehend in a given period of time decreases cognitive overload. While HVP helps solve the human cognition problem in a DAI visualization, it does little when users are presented with a single, horizontally partitioned view, containing too much information to display. Display space for visualization is a limited resource. The maximum possible number of data elements to be displayed in a visualization is confined by the number of pixels available in a given display space. If the number of data items is larger than the number of pixels available for representing a certain category of data, one has to develop a method to abstract the total population of data items without distorting the possible interpretations of the original data set.

Limited display space in visualizing large information systems has prompted extensive efforts to overcome the challenge. Problems researchers face in an attempt to address the spatial restriction in visualization are summarized in [17] as follows:

- In locating a given item of information (navigation)
- In interpreting an item, and
- In relating it to other items if the item cannot be seen in its full context.

Information drill-down is a widely used technique to manage these problems. The technique can be either distortion-oriented or non-distortion-oriented. Non-distortion-oriented techniques display a portion of the whole data set at a time, while distortion-oriented techniques allow a user to examine a local area in detail on a section of the same screen.

Among the views introduced above, the Agent-oriented view produces increasing ineffectiveness in visualization when the number of network nodes that can be potentially visited by an Agent is large. One plausible solution to this scalability problem is abstracting a group of nodes into a single node according to their proximity. When an Agent visits any node belonging to the group, the move is considered to be to the representative super node. If the analyst wants to inspect the region in more detail, she or he can evoke a view displaying the traces made by an Agent within the region represented by the super node. This zooming-in approach can be applied recursively, and is consistent with the non-distortion oriented, drill-down technique. Unlike the Agent-oriented view, the Network-oriented view focuses on visualizing time-line based data such as local congestion, utilization, and average Agent age in a network node. Therefore, capability to navigate data space in a certain context (time in our case) is critical. This is why we choose a distortion-oriented method for visualizing the status information of a DAI-based system. There are a variety of distortion-oriented data presentation techniques available[13]. Poly-focal display, bifocal display, fish-eve view, and perspective wall, are the major ones. Most of these techniques adopt sophisticated mathematical transformation functions. For the visualization of DAI system status, there is no need for such a complicated transformation functions since the visual abstraction of non-graphical data generated from a DAI monitoring system is temporal and is best captured by simple 2D or 3D graphs. However, the fact that data collection in such a system is achieved by sampling implies a non-trivial mechanism to interpolate the graph interval where sampling is not available. In addition, the summary view providing the context to the distortion-oriented presentation needs to be informative enough to guide users to the data sections of their interest. Figure 5 shows a schematic diagram describing the distortion-oriented information drill-down technique employed in our work. The technique is referred to as Vertical View Partitioning (VVP). VVP is composed of three major views: final summary view, intermediate summary view, and raw sampled data view. A summary view is what the analyst sees when the display window of a DAI monitoring system is first opened. As its name suggests, this view provides the visualization of the whole data set in a format that fits the display area of the system. Generating this view is of great importance because other views are initially hidden, obligating the final summary view to represent the rest of the views. When an analyst spots the graph segment of interest in the final summary view, they have the option to activate a slide bar. Once the slide bar is activated, an intermediate view appears underneath the summary graph. The section of the final summary graph bounded by two sliding bars is magnified and stretched into an intermediate summary view. Depending on the amount of sampled data, it is possible that no intermediate view exists, and only the raw sampled data view exists. In this case, the final summary view and the raw sampled data view are equivalent. In our work, the only cases considered are ones with a massive amount of data, and it is assumed that there is at least one intermediate summary view accompanied by a final summary view and a raw sampled data view.

4. Sketch of The Simulation Solution

The simulation and visualization requirements for assessment of DAI policies have been presented for mobile, autonomous, software Agents. The selection of the appropriate simulation paradigm and platform should conform to those requirements while providing flexibility and extensibility in the simulator itself as the distributed computing problem evolves. We examined the three different approaches to discrete event simulation: event-oriented simulation, activity-oriented simulation, and process-oriented simulation[18,19]. In the event-oriented approach, the simulation programmer defines events and writes routines that are invoked as each type of event occurs. Simulated time may pass between events. In the activity-oriented approach, the programmer defines activities that are started when certain conditions are satisfied. In the process-oriented approach, the programmer defines processes that use the resources of the system. Each process can be in one of three states: active (currently being processed), holding (waiting for an interval of simulated time to pass) or waiting (in a queue) for an event to occur. Simulated time passes only when processes are in the hold state[18]. The event approach is easy to understand and computationally efficient but is more difficult to implement than the activity approach. On the other hand while the activity approach is relatively easy to understand, it suffers from poor execution efficiency. The process-oriented approach is less common and requires more planning to implement properly, though it is generally thought to be efficient. In CSIM^{++}, simulation is a process-oriented, discrete-event model. In process-oriented simulation, the primary unit of execution is a process. Execution of a process is initiated by another process. The main program is really process one. All concurrently active processes execute in a quasi-parallel fashion. With certain restrictions, each process appears to be executing in parallel with other active processes, when in fact they are really executing one-at-a-time on a single processor. The environment simulated is a parallel execution environment. Processes can wait for event to occur and thereby cause simulated time to pass[20]. Process-oriented simulation is a convenient tool for developing simulation models of computer and communication systems. It is possible to use a process-oriented simulator, such as CSIM over C^{++}, as an execution environment for parallel programs. Process-oriented simulation is a useful technique for estimating program behavior on novel architectures[20]. [19] compared process orientation and event orientation, and found the run time performance of process-orientation to be superior to equivalent event-oriented models. Therefore, process-oriented simulation may be the best approach for mobile Agent simulation. A CSIM18 model is a C or C++ program that implements a process-oriented, discrete-event simulation model of a system[20]. However, our description of network entities and Agents is consistent with Object Orientation and as such is best achievable using C^{++} over CSIM. Object-Oriented designs reduce the risk of building complex software systems because they are developed to evolve incrementally from smaller systems in which they have been tested for reliability and stability[21].

4.1 Builder

One of the key features of our simulation system is the ability to configure a scenario easily. This is particularly important for studies of complex system behaviors such as the response to the change of local condition or cost of service in network nodes. This type of question is typically difficult to answer through analytical studies and requires significant experimentation. The Builder facilitates the generation of input configurations to the simulation run and allows users to generate scenarios easily and correctly.

The Builder forces the user to create simulation objects in a prescribed order. In this way, the Builder can check the integrity of each object created as well as the integrity of the simulation object set. The Builder steps the user through the creation a number of components: global parameters, geodesic matrix, Trader, payload factors, and mobile Agents for tracing. Global parameters include the traffic pattern, topology, and Agent population size. The geodesic matrix provides the degree of connectivity and distances among nodes (Processors). The topology is created based on the set of global parameters and the geodesic matrix. The Trader stores the characteristics of the Processors such as type and number of services the Processor provides. The Trader is updated dynamically during the simulation according to its interval parameter.

4.2 Traffic Generation, Internal Structures, Processes, and Custom functions

We use the exponential distribution for the inter-arrival times and service times of mobile Agents. An arrival time list is generated for the number of Agents corresponding to the population size specified by the user under the Builder. The simulation of mobile Agents requires a set of attributes associated with each Agent object (i.e. mobile process). A mobile Agent is an autonomous software process that can migrate from one node to another node by its own authority. Mobile Agents perform their tasks autonomously and asynchronously. To simulate this behavior, the CSIM process instance has a number of attributes such as Agent_id, payload size, arrival time, signature size, and a set of weights reflecting the Agent s preference for certain Service Place attributes: time to service(TTS), cost of service(COS), quality of service(QOS). The Agent process must also be tagged with an arrival node name, exit node name, service time for each service in its signature, current age, life expectancy in the system, network exit time etc. These internal attributes are not configured by the user, but appended to the process by the Builder for use in the simulation. The justification is that these are not external attributes associated with the Agent objects in the problem domain. Process-oriented simulation allows simulation programmers to model systems by defining interacting processes as abstractions of active entities in the system. Each process must be able to execute in parallel with other active processes. Each process requires a private data area and active processes must be able to interact (communicate and synchronize) with other active processes[18]. In this simulation, processes find the best Service Place for their subsequent tasks by using a graph theoretic approach, in terms of the minimal cost and overall optimal system throughput. Therefore, Builder and Visualization tools previously described are the pre and post processors employed to manage the complexity and number of simulation entities.

4.3 Run times and memory consumption for tests

The numbers of Processors and mobile Agents are the major factors that affect the run time and memory usage. Memory consumption and running time are measured over a set of test cases to compare system performance among various topologies and Agent populations. As demonstrated in Figures Figure 6 to 9, the increase in memory consumption and system run time is stable enough to study distributed process behavior. As figures 6 and 7 show, memory usage and system time increase monotonically as the number of processors (Service Places) increases. This is because the overhead in the simulation is adding a single Service Planner process for each Service Place added to the network. The size of local congestion in a small number of nodes is similar to the migration delay in a large number of nodes, causing the graph to be monotonical. The traffic intensity derives from the Agent population. In Figures 5 and 6, memory usage and system running time increase linearly to the number of software Agents in the colony.

Figures 10 — 17 begin at time 20. This is because for the first 20 seconds of simulation, the system was in a warming up period and unstable. In earlier experiments, we showed that when the Agent s preference structure is heavily biased toward time to service, that the behavior of the colony is congestion avoidant. The nodes of the network load level over time as a second order effect of individual, Agent decisions. However, in this experiment, services were not ubiquitous in the network. Consequently, Agent s must incur the local congestion penalty associated with migrating to a node best matching the Agent s preference structure, regardless of local congestion. As a consequence, the behavior of the colony can be self-managed only within a range of local congestion and utilization. As Figures 10 to 13 show, the number of Agents enqueued to four representative Service Places ranged from approximately 26 — 52. Figures 14 to 17 depict CPU utilization at those same four Service Places. CPU utilization ranged from approximately 28%-68% for a colony of 100 Agents. These ranges highlight the importance of the service assignment problem in DAI networks. Assignments of services to only a few nodes can result in focused overloads and regional congestion if the majority of Agents in a colony require that service. Additionally, attention should be paid to the probability of a service being visited by an Agent and the distribution of mean service times at a single Service Place in order to balance CPU utilization. All the graphs of Figures 10 to 17 display a characteristic step function with negative slope. The step function is related to the grain size in the post processor. The height of the step is the number of arrivals since the last sample point. Each step has a different length depending on the consecutive tasks' execution times. As a number of tasks are executed, the step has negative slope at the service rate. The beginning and the end of the step correspond to the initial and final queue size of the execution interval. The difference between two steps indicates the increment of arrivals.

5. CONCLUSION AND FUTURE RESEARCH

The autonomous mobile Agent simulator is being developed at University of Missouri-Kansas City under a grant from the Department of Defense (DOD). The objective of our simulation is to evaluate optimal mobile Agent service assignment to nodes on a wide area network. This paper presents the structure and efficacy of process based simulation of such systems. Significant benefits are obtained both from object-oriented and process-oriented simulation using CSIM^{++}. The structure of the simulator is such that the results are optimal by proofs presented in literature, primarily [1,2]. However the running time of the Service Planner is $\mathbf{O}(N^3)$ which is unacceptable. Our current work centers on the development of Agent perception functions and reasoning methods as to the next Service Place to visit relative to the Agent s goal seeking behavior. A Rational approach to migration will improve run times such that the mobility decision does not dominate the execution time for the Agent s service signature. However, the perception functions and reasoning must be evaluated for efficacy against the non-dominated or pareto optimal solutions produced by the graph theoretic service assignment algorithm in the initial release of the simulator. The post processor visualization program is being designed with a view toward the management of a large colony of Agents over a wide area network. Partitioning the information space to the cognitive advantage of human analysts is an approach that improves the effectiveness of a DAI-based visualization. Furthermore, combining view-partitioning with an information drill-down technique such as distortion-oriented visualization, has not often been employed in the implementation of large scale systems. Our views with respect to HVP do not yet include the collective Agent migration view that will be required in an Agent management system.

6. REFERENCES

[1] C.C. Shen and W.H. Tsai, "A Graph Matching Approach to Optimal Task Assignment in Distributed Computing Systems Using a Minmax Criterion," IEEE Transactions on Computers, vol.C-34, No.3, March, 1985

[2] H.S. Stone, "Multi Processor Scheduling with the Aid of Network Flow Algorithms," IEEE Transactions on Software Engineering, Vol. SE-3, No.1, January, 1977

[3] W. David Kelton, "Designing Simulation Experiments," Proceeding of Winter Simulation Conference, 1999

[4] A. M.Law and W.D. Kelton, Simulation modeling and analysis, McGraw-Hill, 1991

[5] S. Bajaj et al, "Improving Simulation for Network Research," USC CS Technical report 99-702b, March, 1999

[6] S. G. Eick, Aspects of Network Visualization, IEEE Computer Graphics and Applications, March 1996, pp. 69-72.

[7] G. A. Miller, The Magical Number Seven Plus or Minus Two: Some Limits on Our Capacity to Process Information, Psychological Review, 63 (2): 81-87, 1956.

[8] G. M. Nielson, H. Hagen, H. Muller, Scientific Visualization, IEEE Computer Society Los Alamitos, California, 1997.

[9] S.S. Foster, D. Moore, B. A. Nebesh, "Autopilot: Experiences Implementing a Distributed Data-Driven Agent Architecture , Proceeding of TOOLS-26'98, august 3-7, 1998, Santa Barbara, California

[10] S. Eick and G. Wilis, Navigating Large Networks with Hierarchies, Proc. of IEEE Visualization 93, IEEE Computer Soc. Press, Los Alamitos, Calif., 1993, pp. 204-210

[11] K. Cox and S. Eick, Case Study: 3D Displays of Internet Traffic, Proc. of Information Visualization 95, IEEE Computer Soc. Press, Lost Alamitos, Calif., 1995, pp. 129-131.

[12] R. Rohrer and E. Swing, Web-Based Information Visualization, IEEE Computer Graphics and Applications, July/August, 1997, pp. 52-59.

[13] H. Liu and D. Hockney, Visualization in Network Topology Optimization, ACM Computer Science Conference Communications Proceedings, ACM, 1992, pp.131-137.

[14] RJA. Buhr, D. Amyot, M. Elammari, D. Quesnel, T. Gray, and S. Mankovski, Feature-Interaction Visualization and Resolution in an Agent Environment, Feature-Interactions in Telecommunications and Software Systems V, IOS Press, Amsterdam, Netherlands, 1998, pp. 135-149.

[15] Abeck, A. Koppel, and J. Seitz, A Management Architecture for Multi-Agent Systems, Proc. of the IEEE Third International Workshop on System Management, IEEE Computer Soc., Los Alamitos, CA, USA, 1998, pp. 133-138.

[16] G. Karam, Visualization Using Timelines, Sigsoft Software Engineering Notes, spec. issue, ACM, 1994, pp.125-37. USA.

[17] Y. Leung and M. Apperley, A Review and Taxonomy of Distortion-Oriented Presentation Technique, ACM Transactions on Computer-Human Interaction, Vol. 1, No. 2, June 1994, pp. 126-160.

[18] H. Schwetman, "CSIM: A C-BASED, Process oriented Simulation Lnaguage," Proceeding of Winter Simulation Conference, 1986

[19] K.S. Perumalla, R.M. Fujimoto, "Efficient Large-Scale Process Oriented Parallel Simulation," Proceeding of Winter Simulation Conference, 1998

[20] H. Schwetman, CSIM Reference Manual, June, 1992

[21] J.A. Joines, S.D. Roberts, "Simulation in an object-oriented world," Proceeding of Winter Simulation Conference, 1999

[22] A. Leinwand and K. Fang, Network Management-A Practical Perspective, Addison-Wesley Publishing Company, Reading Massachusetts, 1993.

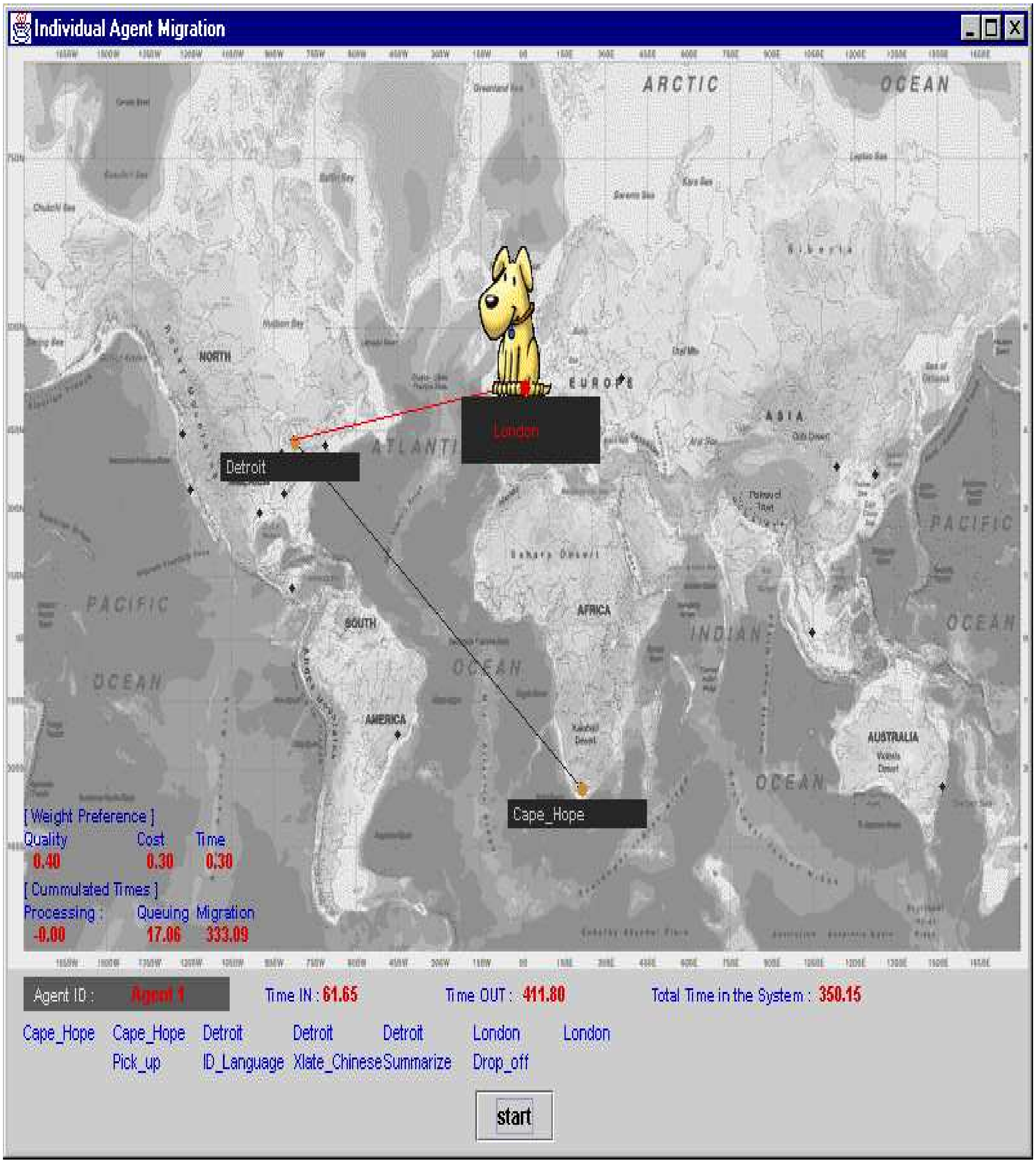

Figure 2.
HVP Agent oriented view

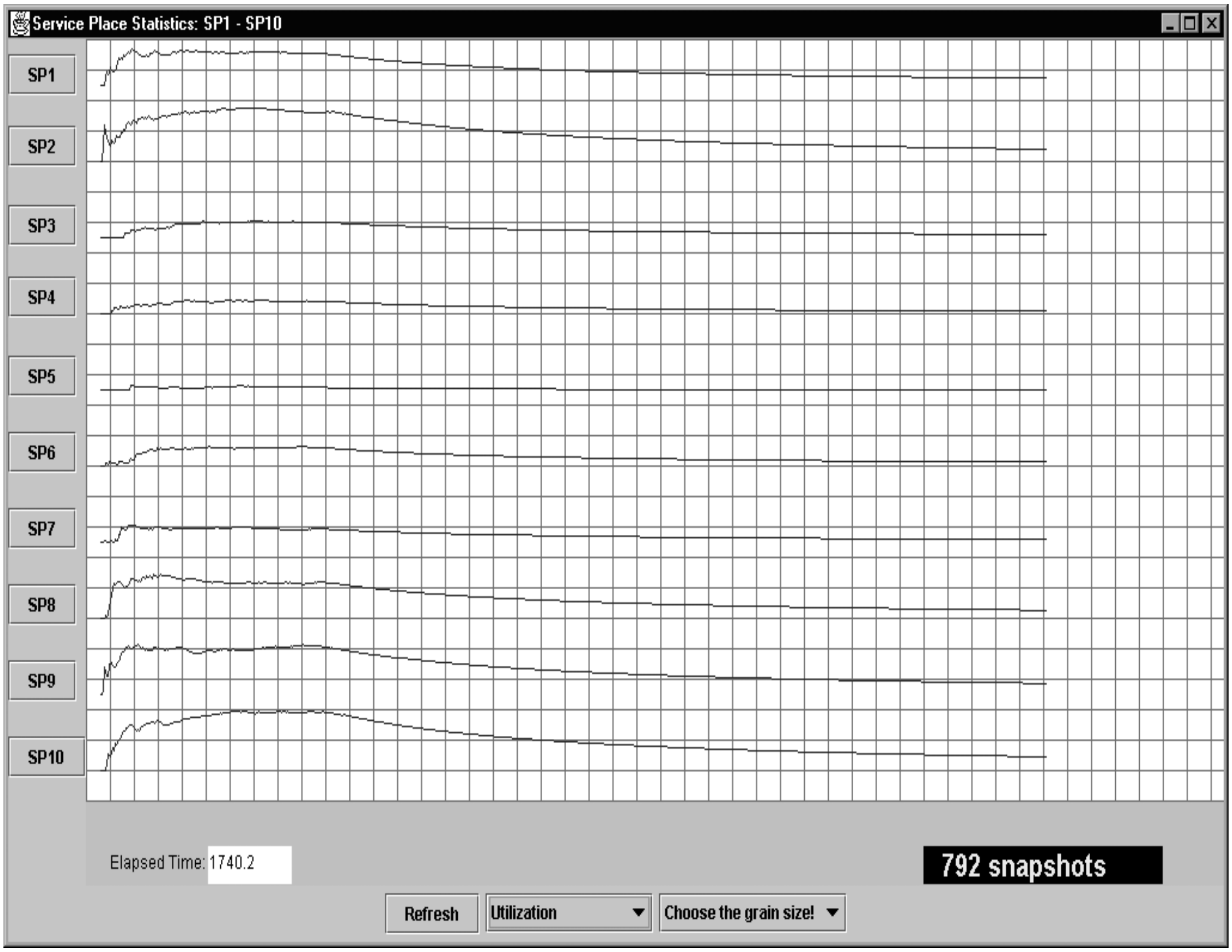

Figure 3.
HVP Network Oriented View: Comparative Service Place Utilization

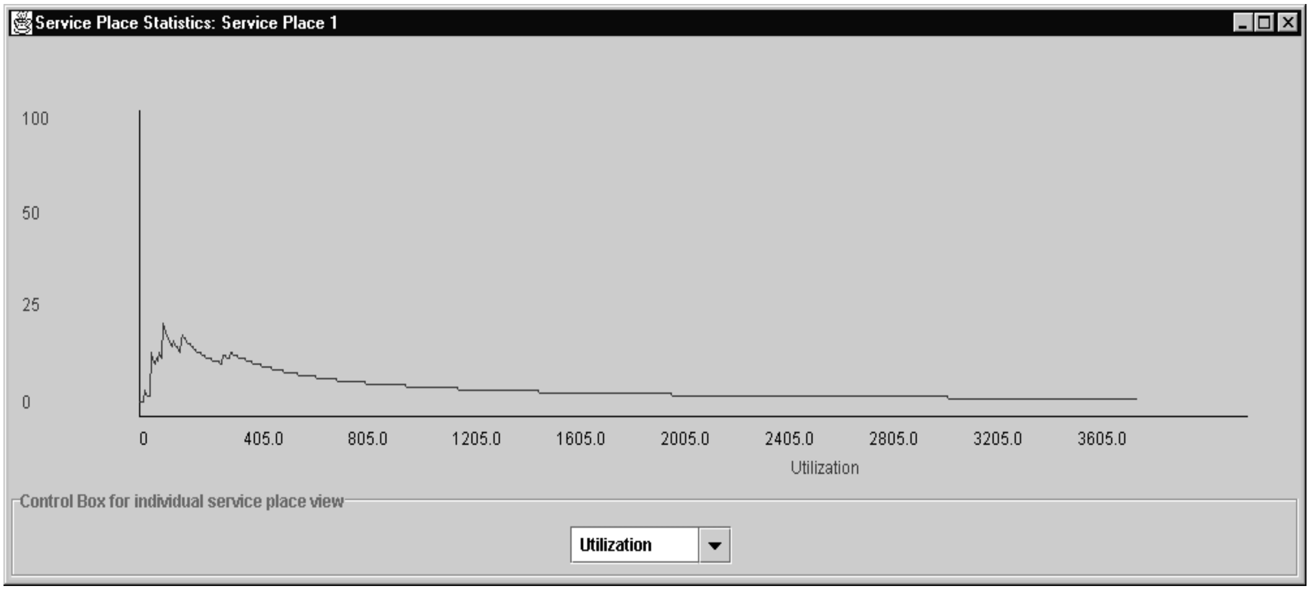

Figure 4.
VVP Network Oriented View: Specific Service Place Utilization

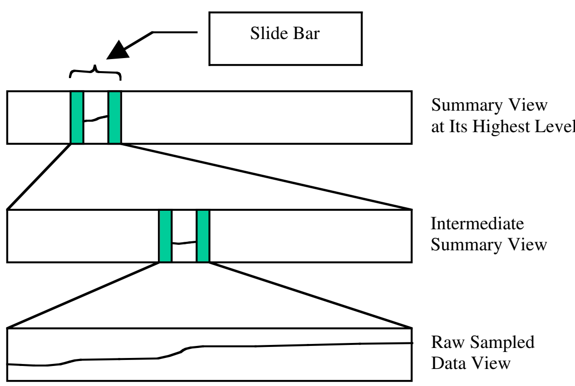

Figure 5.
Dynamic Use Of VVP Distortion Oriented View For Data Interpretation

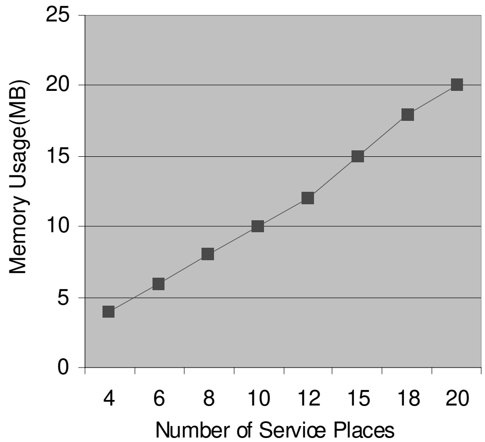

Fig. 6. Memory Usage on Number of Processors

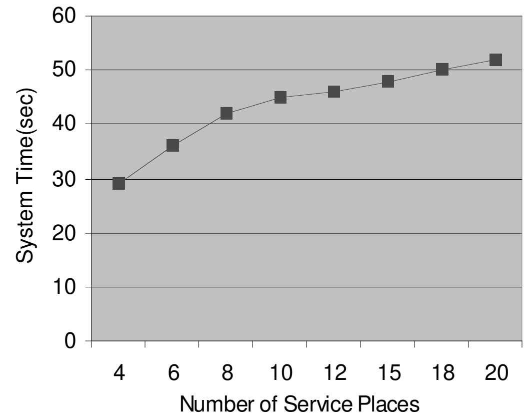

Fig. 7. System Time on Number of Processors

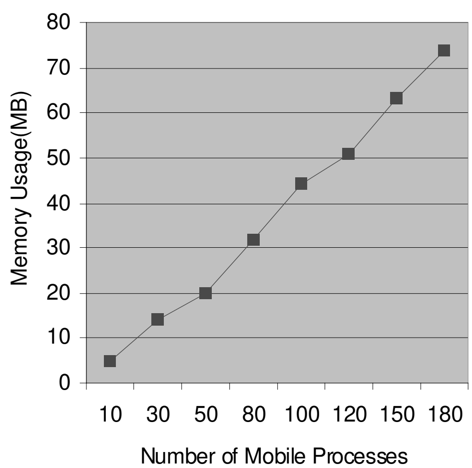

Fig. 8. Memory Usage on Number of Mobile Agents

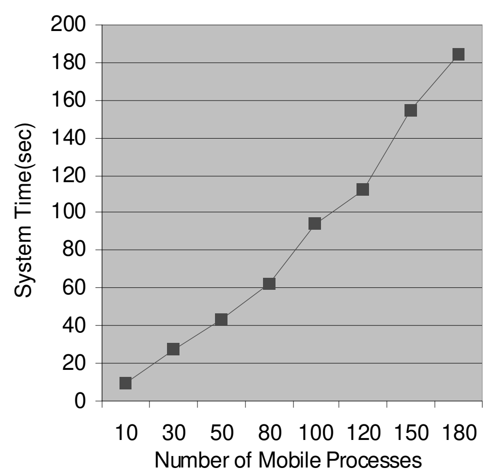

Fig. 9. System Time on Number of Mobile Agents

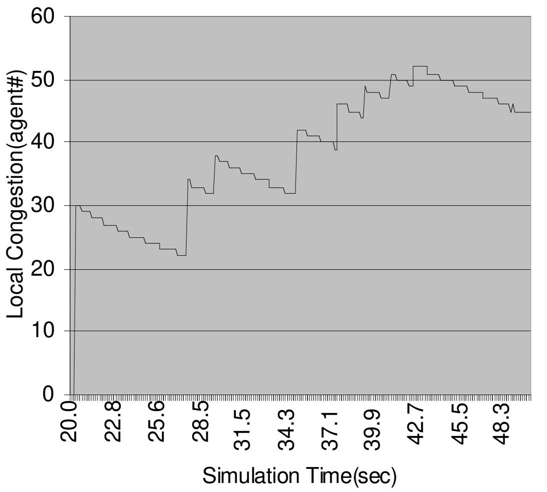

Fig. 10. Local Congestion of Processor-A

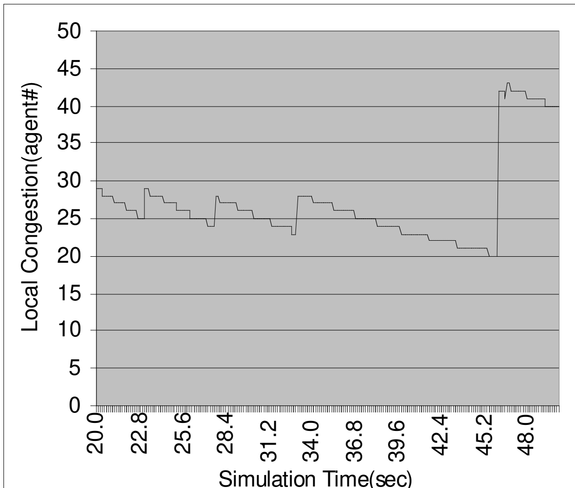

Fig. 11. Local Congestion of Processor-B

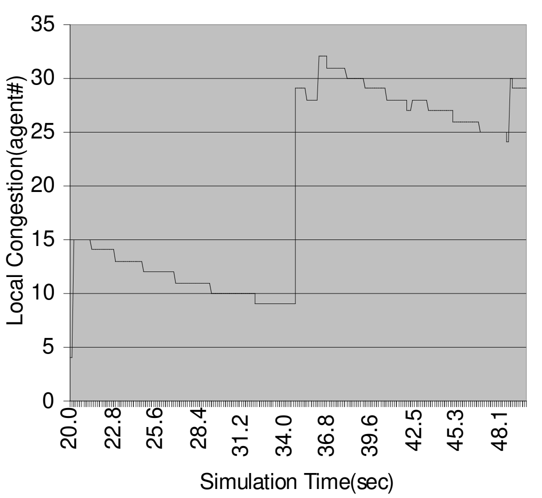

Fig. 12. Local Congestion of Processor-C

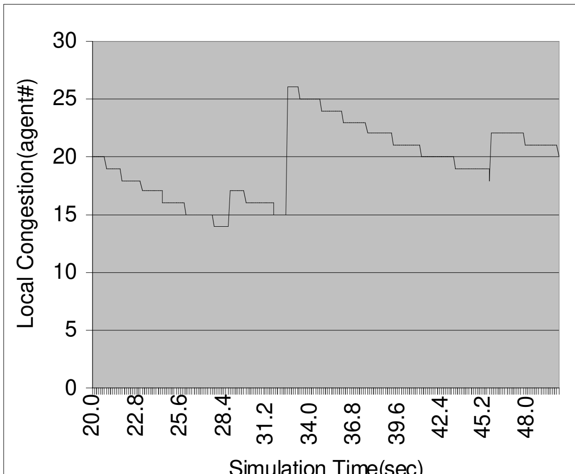

Fig. 13. Local Congestion of Processor-D

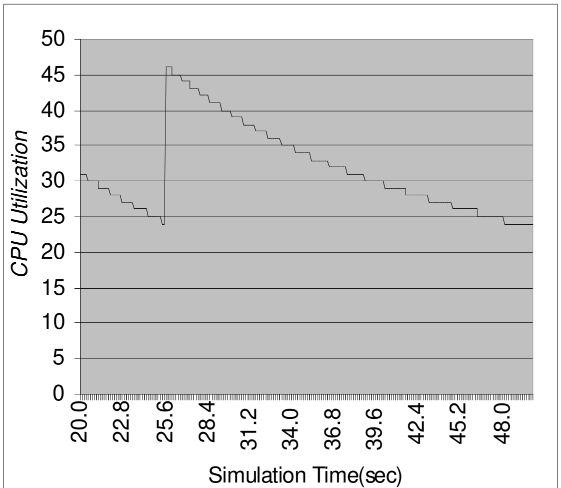

Fig 14. CPU Uitl of Processor-A

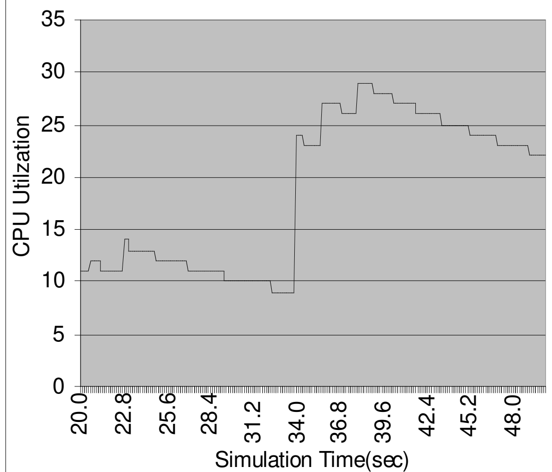

Fig. 15 CPU Util of Processor-B

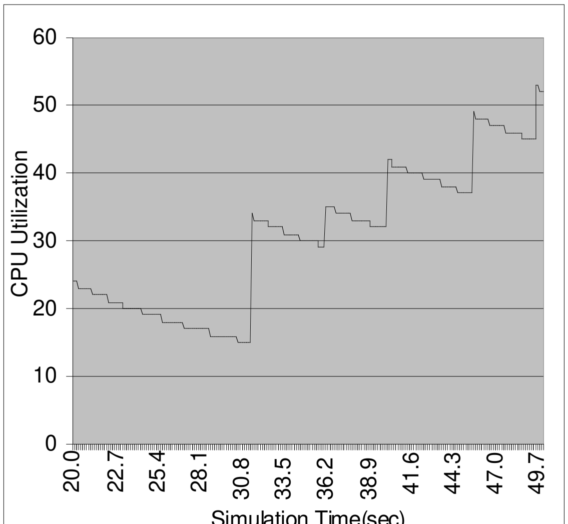

Fig. 16. CPU Util of Processor-C

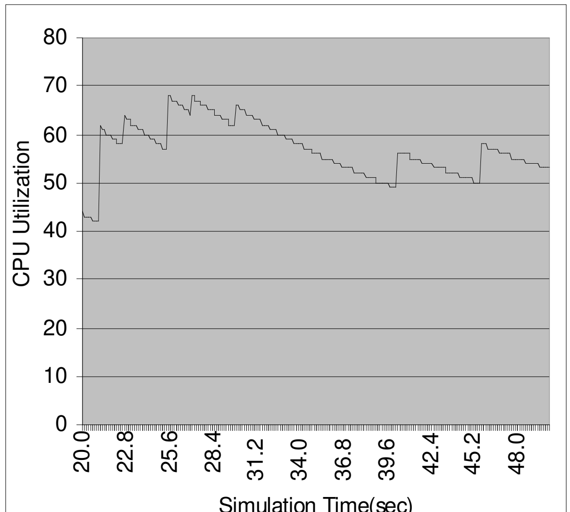

Fig 17. CPU Util of Processor-D