Reliable MAS Performance Prediction Using Queueing Models

Nathan Gnanasambandam, Seokcheon Lee, Natarajan Gautam, Soundar R.T. Kumara
Pennsylvania State University
State College, PA 16801
{gsnathan, stonesky, ngautam, skumara}@psu.edu

Wilbur Peng, Vikram Manikonda
Intelligent Automation Inc.
Rockville, MD, 20855
{wpeng, vikram}@i-a-i.com

Marshall Brinn
BBN Technologies
10 Moulton Street, Cambridge, MA 02138
mbrinn@bbn.com

Mark Greaves
DARPA IXO
3701 North Fairfax Drive, Arlington, VA 22203-1714
mgreaves@darpa.mil

Abstract

In this paper, we model a multi-agent system (MAS) in military logistics based on the systemic specifications of the capabilities and attributes of individual agents (TechSpecs). Assuring the survivability of the MAS that implements distributed planning and execution is a significant design-time and run-time challenge. Dynamic battlefield stresses in military logistics range from heavy computational loads (information warfare) to being destructive to infrastructure. In order to sustain and recover from damages to continuously deliver performance, a mechanism that distributes knowledge about the capabilities and strategies of the system is crucial. Using a queueing model to represent the network of distributed agents, strategies are developed for a prototype military logistics system. The TechSpecs contain the capabilities of the agents, playbooks or rules, quantities to monitor, types of information flow (input/output), measures of performance (Quality of Service) and their computation methods, measurement points, defenses against stresses and configuration details (to reflect command and control structure as well as task flow). With these details, models could be dynamically developed and analyzed in real-time for fine-tuning the system. Using a Cougaar (DARPA Agent Framework) based model for initial parameter estimation and analysis, we obtain an analytical and a simulation model and extract generic results. Results indicate strong correlation between experimental and actual events in the agent society.

Keywords: Multi-agent systems, Survivability, Queueing network models, Technical specifications

1. Introduction

Multi-agent systems that implement distributed planning and execution are highly complex systems to design and model. In this research, we model a survivable multi-agent system (MAS) based on the systemic specifications (TechSpecs) of the capabilities and attributes of individual agents. The MAS under consideration is exposed to significant stresses because it operates in highly unpredictable battlefield-like environments. Even under such hostile conditions, the stated goal of this survivable MAS based logistics system is to deliver robustness, security and performance. Hence, performance prediction using suitable models is vital to being able to tune the actual performance delivered by the MAS.

Within the research domain of military logistics, we are conducting our studies using a continuous planning and execution (CPE) agent society. The CPE society is constructed using the Cougaar MAS development platform developed under DARPA's leadership [2]. From the modeling perspective, the CPE society (or otherwise) is nothing but a collection of distributed agents that lend themselves to be represented by a network of queues. With this motivation, we analytically modeled the CPE society using queueing theory. In doing so, we realized that if the TechSpecs were suitably specified, the generation of the queueing model could be

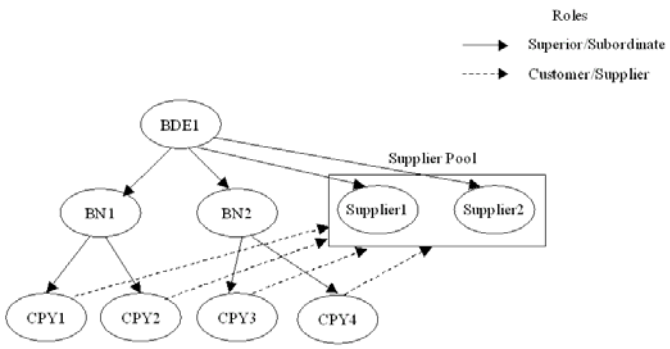

Figure 1. Agent Hierarchy in CPE Society

accomplished with lesser human intervention. The primary function of the model is to help evaluate the performance of the MAS and provide alternatives to steer the agent society towards optimal regions of operation boosting performance in a distributed environment. Therefore the main focus of this research lies in specifying the MAS in a systematic fashion so that queueing models can be derived from the specification.

1.1 Continuous Planning and Execution Society Overview

The CPE society comprises of agents and a world model. Agents in the CPE society assume a combination of command and control, and customer-supplier roles as required in a military logistics scenario. The world model is an artificial source that provides the agents with external stimuli. Figure 1 represents the superior-subordinate and the customer-supplier relations between the brigade (BDE), battalion (BN), company (CPY) and supplier (SUPP) agents as modeled in this research. Each agent in the society constantly performs one or more of the following tasks: 1) Evaluates its own perception of the world state through local sensors and remote inputs; 2) Performs planning, replanning, plan reconciliation and plan refinement; 3) Executing plans, either through local actuators or through sending messages to other agents; 4) Adapting to the environment, e.g. centralizing or decentralizing planning as computational resources permit.

1.2 Definitions

The following definitions are in order when relating to the system under consideration.

Stresses occur due to the operation of the MAS in battlefield environments where events such as permanent infrastructure damage and information attacks adversely affect overall system performance.

Based on the planning activity in CPE, we simply base our *measures of performance (MOPs)* on timeliness or freshness of a plan at the point of usage and on the quality of the plan. Based on the requirements of Ultra*log [3], a broad series of performance measures categorized according to timeliness, completeness, correctness, accountability and confidentiality is available but is outside the requirements of CPE. Some insights about these MOPs can be gained from [6]. The MOPs are the components of the *quality of service (QoS)* expected from the system.

Survivability of a distributed agent based system (or otherwise) is the extent to which the quality of service (QoS) of the system is maintained under stress [6].

Although we consider a survivable MAS, we only concern ourselves with performance analysis in this work. We assume that a global controller exists that coordinates between threads relating to performance, robustness and security. The contents of this paper are organized in the following way. In Section 2, we introduce the concept of TechSpecs based design and some of the benefits associated with this approach. We then discuss the components of the CPE society in detail and organize the TechSpecs for CPE into various categories in Section 3. The discussion on TechSpecs leads us further in the direction of how to utilize them to form models. We dicuss some models we created in Section 4. We provide two analytical methods using queueing networks to model a small example in CPE and verify our models using a simulation. Finally, in Section 5 we discuss our conclusions and some possible directions for future research.

2. The Concept of TechSpecs Based Design

Technical Specifications (or TechSpecs) refer to component-wise, static information relating to agent input/output behavior, operating requirements, control actions and their consequences for adaptivity [7]. In addition to outlining a comprehensive set of functionalities, the TechSpecs are responsible for the definition of domain MOPs, their respective computational methodologies and QoS measurement points. The construction of TechSpecs helps us proceed in the following direction:

1. Use the *specs* to ensure a close mapping between MAS functionality and an abstracted model. An apparent choice here is a queueing model because of similarities between multi-class traffic in queueing networks and the different types of *flows* in CPE.

2. Establish the parameters of the queueing model - from TechSpecs directly (eg. update rate at a node) as well as by collecting empirical data from sample runs (eg. processing times).

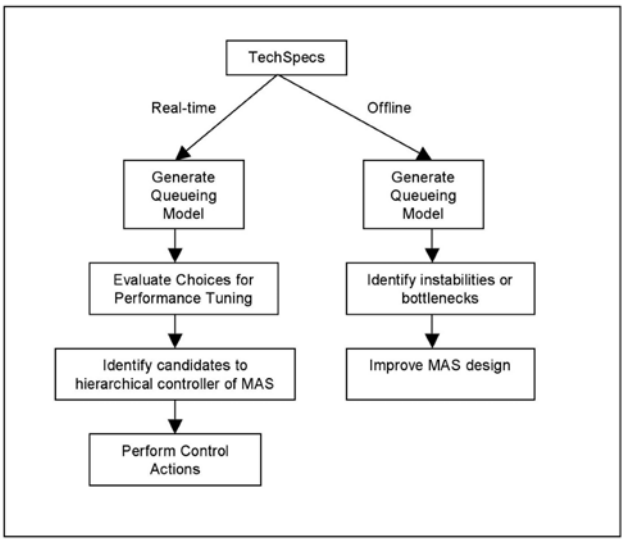

Figure 2. TechSpecs based MAS Design

3. As the queueing model provides an indication of system performance for a given configuration, use it to quickly explore options for control (choices resulting from adjusting (queueing) parameters or configurations). Once a suitable candidate is obtained, this choice is translated back into the application level knob settings (for control) to result in better QoS for the MAS.

The direction that TechSpecs motivates us to take is illustrated in Figure 2. Figure 2 indicates that we could use the *specs* in an online or offline fashion. Because the functionality is clearly defined using TechSpecs, offline analysis can be independently carried out to remove instabilities from the MAS design. Assuming automatic conversion from TechSpec to a model is feasible, TechSpecs have a real-time use as well - i.e. use the *specs* as a template to derive the model. As noted above, the candidate parameters from the queueing model (parameters that may lead to performance improvement) cannot be used directly. Reconverting these choices to actual control knob settings may be handled by a seperate global controller. We allude to this in Section 3.2.

It can be noted that the idea of TechSpecs bears analogy to the conventional control problems in electronic or hardware realms where the technical specification or rating could be leveraged to effect better design and control. This was one of the motivating factors for TechSpecs based design for MAS.

Benefits of TechSpecs

The advantage of establishing comprehensive TechSpecs is that it leads to the codification of requirements, functionalities, measurements and responses to situations. Further, it enhances the potential to aid the MAS configuration (what nodes to put agents on) both statically and dynamically. An incomplete list of potential benefits of using a TechSpecs based approach to MAS design is provided below:

- Enhancement of the MAS Design: Since TechSpecs impose the requirement of predictability, the MAS components must be built with fidelity
- Distribution of Knowledge: TechSpecs carries with it the idea of being composable. By using the TechSpecs of smaller components as building blocks we can build the TechSpecs of larger systems when the system expands.
- Concurrent Analysis: Model building can be concurrent with actual MAS design. Provides a look-ahead capability to avoid regions of instability or bottlenecks (especially from queueing analysis).

3. CPE Society TechSpecs

In this section we discuss the formulation of TechSpecs. In order to build TechSpecs the functionalities of the components of the CPE society are defined as described in Section 3.1. We then categorize the capabilities of CPE components in a manner that would lend itself to easy translation into the queueing models. We then show through examples how the mapping process between a TechSpec and a queueing model could be interpreted. This would enable us to analyze the MAS using the models we develop in Section 4.

3.1 Description of CPE Society Components

The World Model: The world model refers to the conceptual set-up that provides the agents with external stimuli. It captures a military engagement scenario using a 2-dimensional model of the world. As shown in Figure 3, CPY agents moving along the x-axis engage an unlimited supply of targets that move along the y-axis. The targets move at a fixed rate but engagement slows them down. While a probabilistic model is chosen to create targets and engaging them, a deterministic model is chosen for fuel consumption (which is dependent on the distance moved). A logistics model for resupplying the units with fuel or ammunition is based on the demand generation from maneuver plans. Currently, the world model is also implemented as an agent.

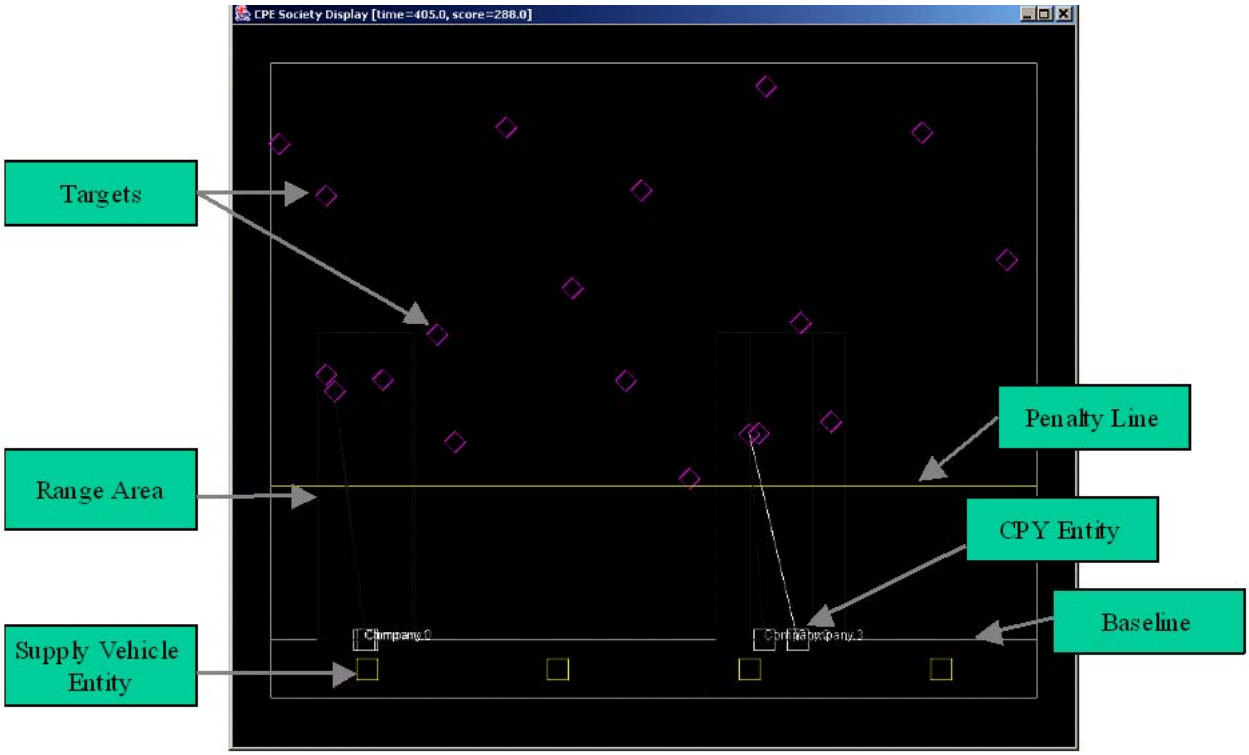

Figure 3. The World Model

CPY Agent: Each CPY unit is designated a target area for engaging in combat actions. These action require a superior agent (BN) to supply a maneuver plan to each of the CPY agents. This plan enables the CPY agent to move along the x-axis and engage the enemy by firing. Each of these agents simulate sensors and actuators. The CPY agents consume resources and subsequently forward the demand to SUPP agents. The current status is reported to superior agents to enable replanning.

BN Agent: The BN agent maintains situational awareness of all the agents under its direct command and performs (re)planning for them using a consistent set of observations that is collected continously. The BN agent has to execute a branch and bound algorithm of a specified planning depth and breadth to generate a maneuver plan for its subordinates. The BN agent serves as a medium for transferring orders from superiors to subordinates.

BDE Agent: The BDE agent is responsible for generating maneuver plans for the BN and CPY agents although this implementation does not empower the BDE with that functionality.

SUPP Agent: SUPP agents represent an abstracted set of supply and inventory and sustainment services. These agents take maneuver plans from the CPY agents and supply them with fuel or ammunition. It is currently assumed that the SUPP units have infinite inventory. Projected and actual consumption depend on the sustainment plan generated from orders and the presence of enemy targets.

3.1.1 TechSpec Organization

Right at the outset, our goal is to embed enough transperancy in the TechSpecs to allow the generation of models (queueing models). Hence, we extract the input/output behavior, state, actions and QoS for each entity within CPE and form the following categories within the TechSpecs :

- Internal State of an Agent: Corresponds to continously updated variables or data structures corresponding to the actual working of the agent.
- Inputs: Relates to distinct classes of information received or sent to or from an agent respectively.
- Outputs: Information provided to other agents.

- Actions: Determines the actions that need to be taken as a result of state changes or the dependencies introduced by input/output operations.

- Operating Modes: The fidelity or the rate at which outputs are sent may relate to the operating mode of an agent. Switching operating modes may be necessary to alter QoS requirements or as counter-measure for stress.

- QoS Measurement (QoS Measurement Points): Indicates the measure of performance that needs to be monitored or measured in order to compute the QoS at the designated measurement point. For example, when we consider queueing models, we would be interested in measuring the average waiting times at different agents to compute a quantity such as the freshness of the maneuver plan.

- Tradeoffs: While these may not pertain to every agent, some agents have the capability to trade-off a certain measure of performance to gain another. These are specified explicitly in TechSpecs.

This categorization facilitates the delineation of specific *flows* of jobs between agents. For example, consider the following *flow*: External stimuli at CPY gets converted to update tasks at CPY, delivered to BN as updates, converted to a manueuver plan at BN, delivered to CPY and then forwarded to SUPP for sustainment. From a queueing theory perspective, the update tasks that originates at CPY and end up at BN for the purpose of planning could constitute a class of traffic with CPY and BN acting as servers to process these tasks. Similarly, consider the flow where external stimuli received at CPY end up as updates at BDE through BN. This could be regarded as another class of traffic. At this point it is important to notice that classes of traffic could be derived form the input/output details embedded within TechSpecs. We decribe how we handle these flows in the queueing network formulation in Section 4.

Another example of how we could describe something in the application domain (say a QoS metric) with the queueing model is as follows. If one is interested in how fresh a maneuver plan is at its usage point (i.e. CPY), the model could describe it in terms of the queueing delays for a particular class of traffic. In our application, this very quantity happens to be a QoS metric called manuever plan freshness. In the actual MAS, this metric is calculated directly from the timestamps that are tagged to the tasks.

3.1.2 TechSpec Representation

Although an elaborate discussion of the format of TechSpec representation is outside the scope of this paper, we present some aspects of the specification directly relating to the application and some infrastructural requirements that need to be part of the specification.

Table 1. TechSpec Categories: Application Perspective

Property / Attribute	BDE	BN	CPY	SUPP
Superior	-	BDE	BN	CPY
Internal State	Overall Status	World State	Maneuver Plan Fuel Ammunition	Sustainment Plan World State
Inputs	Update	Update	Update Maneuver Plan	Maneuver Plan
Outputs	Update	Update Maneuver Plan	Update Maneuver Plan	Maneuver Plan
Actions	Update	Plan Update	Update	Plan Update
QoS Measurement Points	-	-	Maneuver Plan Freshness	Sustainment Plan Freshness
Operating Modes	-	High Medium Low	High Medium Low	-
Operating Mode Tradeoffs	-	Planning Depth versus Breadth	-	-

Table 1 represents some TechSpecs categories specific to this application. Simply speaking, this is a tabular representation of the information contained in Section 3.1 organized using the aforementioned categories. From Table 1 one can understand that an output called *update* originates from CPY agent and travels up at BN because BN is CPY's superior. Similarly, an output called *maneuver plan* would reach CPY from BN. One assumption that is being made here is that *updates* travel up the hierarchy and *plans* downward. These outputs form part of the different classes of traffic if observed from a queueing perspective. Another example would be that the *plan action* in the BN agent relates to a functionality in the MAS domain and would simply be abstracted by a *processing time* in the queueing domain.

In addition to the above specification, static requirements of the agents in terms of infrastructure are also embedded into TechSpecs. Some of these requirements for BDE, BN, CPY and BDE agents shown in Table 2.

3.2 Translating TechSpecs to the Queueing Domain

In order to translate the specs into queueing models we first use the following rules:

1. Inputs and outputs are regarded as tasks;

2. The rate at which external stimuli are received is captured by the arrival rate(λ);

3. Actions take time to perform so they get abstracted by processing times(μ_i);

Table 2. TechSpecs: Infrastructure Perspective

Property / Attribute	Node	Agent	Plugin
hasProcessor	Yes (1 atleast)	Depends on node	Depends on agent
hasBandwidth	Yes (1 atleast)	Depends on node	Depends on agent
Processor Speed	Update	Depends on node	Depends on agent
Bandwidth	1 Mb/s	Depends on node	Depends on agent
Location	IP address	Node ID	Agent ID
Operating System	Win2000/Linux	-	-
Memory	1 GB	Depends on node	Depends on agent

4. QoS Metrics such as freshness are in terms of average waiting times at several nodes ($\sum W_{ij}$, i is the node, j is the class of traffic);

5. If tasks follow a particular route (or flow as described in Section 3.1.1), then that route gets associated to a class of traffic;

6. If a particular task goes into the node and gets converted to another task, we say class-switching has occured. For example, in our application *update* tasks go to BN and get converted to *plan* tasks;

7. If a connection exists between two nodes, this is converted to a transition probabilty p_{ij}, where i is the source and j is the target node.

Using the above rules as well as the aforementioned representations of TechSpecs we develop a mapping between the TechSpecs and a queueing model. Although the current procedure is manual, in thoery this procedure could be automated. Such an automatic capabilty of translating TechSpecs would prove very beneficial for predicting performance of the MAS in real-time. Table 3 captures the queueing model abstraction from TechSpecs for the CPY agents. Similarly, we can establish the mapping for other agents as well. Some useful guidelines that were followed in order to translate the TechSpecs into models are as follows:

- Identify *flows* of traffic: Trace the route followed by each type of packet completely within the system boundary i.e. from the entry into the system until it exits the system. These would subsequently form classes of traffic in the queueing model. Care has to be taken to note any class switching.

- Identify the network type: The network could be closed (fixed number of tasks) or open. The CPE is an open system because tasks constantly enter and exit the system.

- Does any parameter of the model require empirical data from the actual society?

Although some aspects in this research are currently being resolved, the following observations can be made.

- Who does the TechSpecs translation? Where does the model run? In our case the translation is done manually at present. The model would run at a place visible to the controller (possibly as a seperate agent at the highest level). The controller we refer to here is the actual effector of control actions throughout the CPE society and is seperate from all we have discussed so far. The role of the controller is also to balance between other threads such as robustness and security.

- The identification of control alternatives is currently centralized. However, we visualize a decentralized, hierarchical controller for effecting the changes.

4. Queueing Network Models (QNMs)

A complex logistics system such as the CPE society has numerous interactions. Yet, if the functionalities are abstracted to capture some application level specifics in terms of queueing model elements (example as shown in Table 3), analytical predictions on the behavior of the MAS can be made. Analytical models are good candidates for enforcing adaptive control quickly and in real-time. Each agent behaves like a server that process jobs waiting in line. Hence, the mapping between an agent and a server with a queue is easily established. Because of the task flow structure and the superior-subordinate relationships in the TechSpecs, queues can be connected in tandem with jobs entering and exiting the system. This results in the formation of an open queuing network.

We conducted initial experiments using an actual Cougaar based MAS, an analytical formulation and an Arena simulation. We used this experiment to bootstrap our modeling process in terms of parameter estimation and calibration. However, working with the MAS was time-consuming as our goal was to identify modeling alternatives and control ramifications. Hence we continued our experimentation with a scaled up queueing model and simulation with the insight gained from working with the actual society.

Thus the open queueing network's parameters were carefully chosen and tasks sub-divided into mutiple-classes to denote a particular task within the MAS. The TechSpecs clearly delineate the input and output tasks facilitating the

Table 3. Queuing Model Abstraction from TechSpecs for CPY Agent

TechSpec Entry	Description	Queueing Model Abstraction
Update (Input)	Input called update arrives from BN	Update packet enters CPY node from environment
Update (Output)	Update packet leaves CPY to go to BN	Route exists from CPY to BN for update packet. Reflects as a probability of transition
New Maneuver Plan (Input)	Input received at CPY from BN. Yet to be processed	Plan packet enters CPY and waits in queue
Maneuver Plan (Output)	Output from CPY to SUPP. Plan is processed and is ready to leave.	Plan packet that is ready to exit the queue after service
Update (Action)	Perform the action called update at CPY. Would consume finite time.	Modeled by the processing time for the maneuver plan packet. The maneuver plan would be processed at a specific rate. This rate would be determined by experiments.
Maneuver Plan Freshness (QoS Point)	Calculate this QoS at this node. Using timestamps for a flow the agent will compute the freshness.	The freshness is the sum of waiting times in the queues that the packet has visited until this point.
Status Update Rate (Operating mode)	The rate at which the stimuli from environment is received.	If the status update rate is changed, this can be accommodated in the queueing model by a new arrival rate for the update tasks. Also the processing rate for the class may have to be adjusted.

mapping to arrivals and services in a queueing network. Application level QoS measures of the MAS are calcuated in terms of the waiting times (or other equivalent perfromance measures) at the individual nodes of the QNM.

Figure 4 is a representation of the CPE society from a queueing perspective. We show two types of tasks flowing in the network namely the *plan* (denoting maneuver and sustainment) and the *update* tasks. These tasks can be divided further into three classes of traffic. The first class refers to *update* packets entering at the CPY nodes and proceeding further as *updates* to BDE through BN. Class 2 relates to those *update* packets that are converted to *plan* tasks. There is class-switching at nodes 2 and 3 and we introduce approximations to deal with this later in the paper. The third class relates to the maneuver *plan* tasks that reach SUPP nodes through CPY. Although we know multiple task types exist in the MAS, by making the simplifying assumption and treating all job classes alike we analyze the MAS using Jackson networks [5] in Section 4.1. We further analyze the system taking into accout multiple classes of traffic as discussed in Section 4.2. We compare the two analytical approaches with a simulation model.

4.1 Jackson Network Model

We apply a single class Jackson network [5] formulation for open queuing networks to our example by choosing a weighted average service time for nodes with multiple classes. The nine agents of the MAS considered here can then be assumed to be M/M/1 systems. The arrival rates of the open network can be computed by solving the traffic equations. Assuming the load is balanced to start-with, the routing probabilities are also known. If each node of the

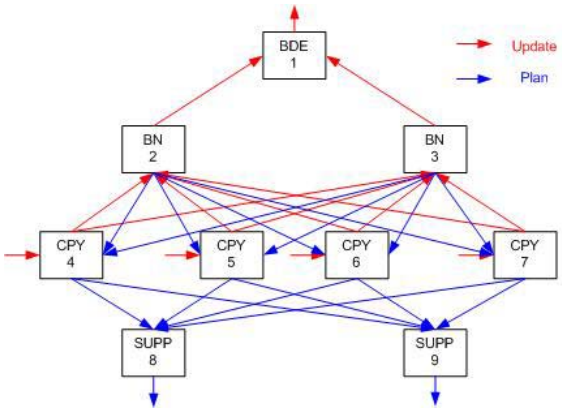

Figure 4. Task Flow in the MAS

system is ergodic, we can calculate the steady state probabilities and performance measures of the entire network by computing these measures for every agent exactly as in an M/M/1 system.

We consider a simple example. For this queueing model, we assume all tasks are of a single type and do not distinguish between classes as shown in Figure 4. Let λ_{0i} and λ_{i0} be the rate of arrival and exit into and from the i^{th} node respectively. Since the routing probabilties are known we can calculate the arrival rates λ_i of each of the nodes of the open network by solving the following traffic equations:

$$\lambda_i = \lambda_{0i} + \sum_{j=1}^{9} \lambda_j p_{ji}\,,\ i = 1, ..., 9\,.$$

The routing probabilties (p_{ji}: probability from i (column index) to j (row index)) for the balanced case are as follows:

$$\begin{matrix} 0 & 1/5 & 1/5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/4 & 1/4 & 1/4 & 1/4 & 0 & 0 \\ 0 & 0 & 0 & 1/4 & 1/4 & 1/4 & 1/4 & 0 & 0 \\ 0 & 1/5 & 1/5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/5 & 1/5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/5 & 1/5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/5 & 1/5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/4 & 1/4 & 1/4 & 1/4 & 0 & 0 \\ 0 & 0 & 0 & 1/4 & 1/4 & 1/4 & 1/4 & 0 & 0 \end{matrix}$$

Note that the customer exits from a node i with probability $1 - \sum_j p_{ji}$. Once the arrival rates are known, we can calculate the average waiting times at the nodes by using the following formula:

$$W_i = \frac{1/\mu_i}{1 - (\lambda_i/\mu_i)}\,,\ i = 1, ..., 9\,.$$

The QoS metrics namely maneuver plan freshness (MPF) and sustainment plan freshness (SPF) are calculated in

terms of the average waiting times of the nodes at each level (W_{CPY} , W_{BN} , W_{SUPP}) as follows:

$$MPF = 2W_{CPY} + W_{BN} \,,$$

$$SPF = 2W_{CPY} + W_{BN} + W_{SUPP} \,.$$

If the load is not balanced and the waiting times are different for the different branches, the QoS measures are accordingly calculated. It can be observed that two methods of control are straightaway obvious: 1) Adjust the μ_i so that we could process faster if possible, 2) Alter the transition probabilties p_{ji} to divert traffic to nodes that are less loaded. Although we allude to some control methods, these are outside the scope of this paper.

4.2 BCMP Network Model

We apply the Baskett, Chandy, Muntz and Palacios (BCMP) algorithm [5] with a small modification to the above example. The network considered here consists of nine nodes and three class of traffic. The first class correponds to the stream that enter the CPY nodes and get sent to BDE through BN as updates. The second class corresponds to the tasks that enter the CPY nodes and get sent to the BN nodes for planning. The second class is converted to a plan and fed back to the CPY nodes. As class-switching occurs here we make a first order approximation and feed this as an independent class back at CPY nodes as tasks of the third class. Since most tasks are of the update type it makes sense to serve the latest update first and hence we follow the LCFS-PR (last come first served with preemptive resume) scheme whereever there are multiple classes. This allows us to assume the service rates to be exponential. Since all tasks arrive from the environment we assume the arrival process to be a Poisson Process.

If λ_{ir} is the arrival rate of the r^{th} class at the i^{th} node, $\lambda_{0,ir}$ is the arrival rate of the arrival rate of the r^{th} class at the i^{th} node, and $p_{js,ir}$ is the probability that a task of class s at the j^{th} node is transferred to a task of class r at the i^{th} node, then the arrival rates for each class at the individual nodes can be calculated using the following traffic equations:

$$\lambda_{ir} = \lambda_{0,ir} + \sum_{j=1}^{9}\sum_{s=1}^{3} \lambda_{js} p_{js,ir} \,, \;\; i = 1, ..., 9 \,.$$

The routing probabilties (p_{ji}: probability from i to j) for the class 1 tasks (portion of update tasks that go to BDE) are as follows:

$$\begin{array}{ccccccccc}
0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1/2 & 1/2 & 1/2 & 1/2 & 0 & 0 \\
0 & 0 & 0 & 1/2 & 1/2 & 1/2 & 1/2 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}$$

The routing probabilties (p_{ji}: probability from i (column index) to j (row index)) for the class 2 tasks (portion of update tasks that leave at 2 or 3) are as follows:

$$\begin{array}{ccccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1/2 & 1/2 & 1/2 & 1/2 & 0 & 0 \\
0 & 0 & 0 & 1/2 & 1/2 & 1/2 & 1/2 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}$$

The routing probabilties (p_{ji}: probability from i (column index) to j (row index)) for the class 3 tasks (portion of update tasks that enter node 4,5,6 or 7 and proceed to node 8 or 9) are as follows:

$$\begin{array}{ccccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1/2 & 1/2 & 1/2 & 1/2 & 0 & 0 \\
0 & 0 & 0 & 1/2 & 1/2 & 1/2 & 1/2 & 0 & 0
\end{array}$$

Once the arrival rates for the different classes at all nodes are known, the waiting time (W_{ir} *or* $W_{i,r}$) at node i for class r was calculated as follows:

$$W_{ir} = \frac{\lambda_{ir}/\mu_{ir}}{(1 - \sum_{r=1}^{3} \lambda_{ir}/\mu_{ir})\mu_{ir}} \,.$$

The application level QoS measures were calculated in terms of the node level average waiting times of the different classes of the BCMP network as follows:

$$MPF = W_{CPY,2} + W_{BN,2} + W_{CPY,3} \,,$$

$$SPF = W_{CPY,2} + W_{BN,2} + W_{CPY,3} + W_{SUPP,3} \,.$$

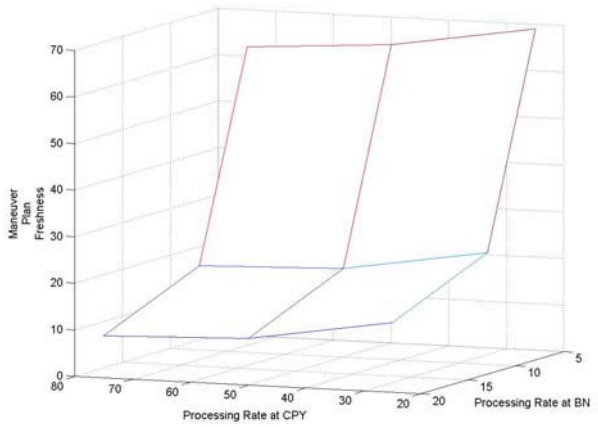

Figure 5. Maneuver Plan Freshness using Jackson Network

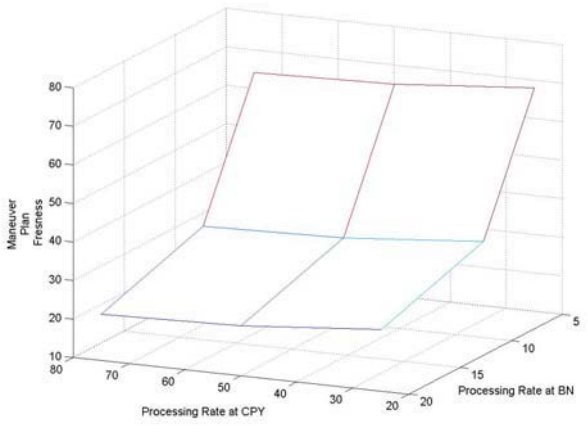

Figure 6. Maneuver Plan Freshness using BCMP Network

We assume that the load is initially balanced. Yet in the unbalanced case, waiting times for the different branches can be calculated seperately.

We studied the impact of changing the processing rates at the nodes to illustrate the benefit of deriving a online queueing model that could form an integral part of a controller. Three methods were followed: 1) Jackson network model, 2) BCMP network model, 3) A Discrete-Event Simulation Model in Arena [1]. We compute the maneuver plan and sustainment plan freshness from the average waiting times of the individual nodes. We assume the processing rate for *class 1* tasks, $\mu_{update\_tasks} = 10Mb/s$ at all the nodes. We assume that the overall arrival rate from the environment is according to a Poisson Process with $\lambda = 2\,Mb/s$. We vary the processing rates for the *class 2* tasks at BN and CPY and observe the impact on maneuver plan freshness as shown in Figure 5 and Figure 6. The low value of processing rates at the BN agent for class 2 tasks are in line with reality, wherein the BN agent implements a search procedure that is more time-consuming than to process class 1 tasks which are updates meant for superiors in the chain of command. We found that the Jackson network matched reasonably well with the simulation results. The multi-class BCMP method performed better than the Jackson network because it was able to capture more of the MAS's characteristics using different classes of traffic. This can be observed by comparing Figure 5 and Figure 6 with Figure 7.

4.3 Discussion

We consider only two parameters (processing rates for the *class 2* tasks at BN and CPY) for variation and nine experiments for each method. We do this to keep the calculations simple. It can be observed from Figure 5 and Figure 6 that adjusting the processing rate in BN impacts the QoS significantly as opposed to altering processing rates at CPY. Hence, to increase performance, the controller may have to adjust the application level knobs to provide a greater processing rate for the planning tasks. Similarly, other trends can be observed by adjusting other parameters.

With these models, we believe it is be possible to identify unstable regions and steer the MAS towards regions providing better QoS. The running time of these models in $Matlab$ is less than one second per iteration. If embedded within the system, several alternate and feasible system configurations can be simulated to identify candidate choices for performance improvement.

5. Results and Future Directions

The hierarchy within the MAS, the specification of static attributes and the similarity between a distributed MAS based planning procedure and queueing network with multiple classes facilitate the performance modeling of the MAS using QNMs. TechSpecs are a structured method to encapsulate static data and distribute them because agent based planning applications are inherently distributed. From TechSpecs, queueing models (offline and online) can be developed for a cluster of nodes. The QNM will serve as an performance analysis tool for that cluster of nodes.

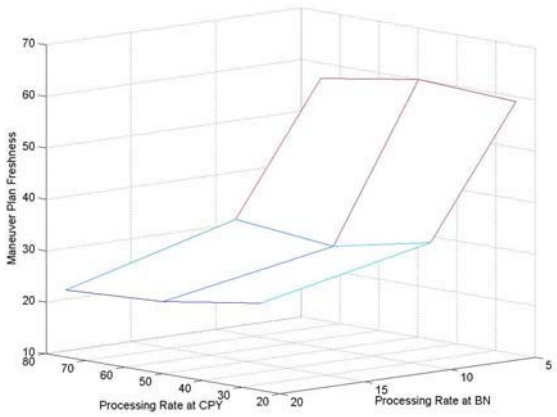

Figure 7. Maneuver Plan Freshness using Simulation

The main contributions of this work is that we have identified that TechSpecs could serve as good template that can guide MAS design and model development in a concurrent fashion. We have codified the static attributes of MAS in such a way that QNMs may be constituted from distributed information, especially in realtime. This technique for adaptivity by using a model on demand to predict trends in QoS may be helpful in building survivable systems.

Currently, work is ongoing to identify an appropriate method of representation of TechSpecs that would have some reasoning and deduction capabilties such as OWL [4]. A module that could convert this representation of TechSpecs into queueing models automatically would be useful in this endeavor. An approach that would identify alternate choices for performance improvement is also necessary. Finally, a controller that actually uses the analysis from the QNMs to optimize the global utility is also being pursued.

Acknowledgements

The work described here was performed under the DARPA UltraLog Grant#: MDA972-1-1-0038. The authors wish to acknowledge DARPA for their generous support.

References

[1] Arena. *www.arenasimulation.com*. Rockwell Automation.

[2] Cougaar open source site. *http://www.cougaar.org*. DARPA.

[3] Ultralog program site. *http://www.ultralog.net*. DARPA.

[4] Web-ontology (webont) working group. *http://www.w3.org/2001/sw/WebOnt/*.

[5] G. Bolch, S. G. H. de Meter, and K. S.Trivedi. *Queuing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications*. John Wiley and Sons, Inc., 1998.

[6] M. Brinn and M. Greaves. Leveraging agent properties to assure survivability of distributed multi-agent systems. *Proceedings of the Second Joint Conference on Autonomous Agents and Multi-Agent Systems (Poster Session)*, 2003.

[7] A. Cassandra, D. Wells, M. Nodine, and P. Pazandak. Techspecs: Content, issues and nomenclature. *Technical Report, Telcordia Inc. and OBJS Inc.*, 2003.