

ACTIVE: Agile Coordinator Testbed Integrated Virtual Environment

R. Emami, J. Cheng,
D. Cornwell, M. Feldhousen,
C. Long, V. Malhotra, I. Starnes
Global InfoTek, Inc.
1920 Association Drive
Reston, VA 20191-1545
1-703-652-1600

{gemami, jcheng, dcornwel,
mfeldhousen, clong, vmalhotra,
istarnes} @globalinfotek.com

L. Kerschberg*, A. Brodsky**
George Mason University, *KRM, Inc.
and **Adaptive Decisions, Inc.
ISE Department, MS 4A4
Fairfax, VA 22030-4444
1-703-993-1640
{kersch, brodsky}@gmu.edu

S. Zhang
University of Mass-Dartmouth
Computer and Information Science
285 Old Westport Rd.
North Dartmouth, MA 02747-2300
1-508-999-8294
x2zhang@umassd.edu

ABSTRACT

This paper describes the specification, design and development of ACTIVE, a testbed for the testing and simulation of large-scale agent-based systems. ACTIVE is being developed as part of DARPA's Coordinators program. The goal of the ACTIVE testbed is to support the simulation of large collections of distributed cooperating agents in solving constrained scheduling problems.

Categories and Subject Descriptors

I.2.11 {**Artificial Intelligence**}:Distributed Artificial Intelligence.

I.2.5 {**Artificial Intelligence**}:Programming Languages and Software.

I.2.8 {**Artificial Intelligence**}:Problem Solving, Control Methods and Search.

General Terms

Algorithms, Measurement, Performance, Design, Reliability, Experimentation, Languages, and Verification.

Keywords

Agent testbed, Scalable testbed, Test Harness, CTAEMS, GMASS.

1. INTRODUCTION

The goal of the ACTIVE system is to provide the infrastructure to support the configuration, simulation, execution, monitoring and evaluation of teams of distributed agents that cooperate in solving large-scale scheduling problems that cannot be solved in a centralized fashion. The distributed nature of the problem, the sheer number of agents involved and the complexity of the tasks

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Conference '04, Month 1-2, 2004, City, State, Country.
Copyright 2004 ACM 1-58113-000-0/00/0004...\$5.00.

together with their constraints, invites new approaches to solving this problem. The Coordinators Program — managed by Dr. Tom Wagner of the Defense Research Projects Agency (DARPA) — seeks to research, develop, and deploy agent-based systems in support of emergency responders in disaster relief scenarios as well as war-fighters in combat situations.

It is essential to automate the tests, in order to support the number of software agents needed to support these personnel. Large scale testing will be required. Hundreds or even thousands of tests must be run automatically and unattended. The Coordinators agents must be tested with numerous input scenarios and the tests must be repeatable and the results quantifiable. We at Global InfoTek, Incorporated are designing, building and testing the ACTIVE testbed to support three agent-based projects, each of which is specifying, designing, implementing and testing their own Coordinators agents. The performer teams are from Honeywell, Information Sciences Institute at USC, and SRI International.

The ACTIVE Testbed is designed to support both testing and evaluating the results of the tests. A Scenario Generator has been developed to generate scenarios based on input parameters. Developing the scenario manually would neither be feasible nor reliable given the scale of the testing. The Centralized Scheduler provides a baseline set of solutions for each input scenario that will be used to test the Coordinators agents. Chronos provides time management for the Centralized Scheduler.

The ambitious goals of the Coordinators Program, the size of the agent communities being fielded by the three teams, and the complexity of the problem space in which the agents operate, requires that the simulation testbed be both scalable and robust. Although we are aware of other testbeds for agent-based systems, we feel that our approach is better in that it is modular, service-oriented, scalable, and able to operate "24/7" so as to support the vast number to tests needed to evaluate the performance of each performer team's solutions to the agent-based team scheduling and coordination problems.

1.1 Related Work

Several agent-based testbeds have been proposed and built. Examples are the ART Testbed (Agent Reputation and Trust) at the University of Texas at Austin [2], JADE [1] at Telecom Italia in Naples, the Autonomous Multi-Agent Testbed at the University of Washington [5], and the AgentCities Testbed [7].

The ART Testbed [2] has an international team of researchers whose focus is to coordinate game design, game specification, testbed development and competition administration. The competition testbed is designed to fulfill two roles: 1) provide a forum in which competitors can compare results using objective metrics, and 2) provide a suite of tools with flexible parameters to allow researchers to perform customizable and easily-repeatable experiments. The architecture consists of multiple agents connected to a simulation engine which accesses a database, a game setup interface and game monitor interface. The focus is to measure agent trust and reputation in the game-playing environment. This is different from the ACTIVE Testbed environment.

The Java Agent DEvelopment Framework (JADE) [1] has a large following in the FIPA agent community. It is designed to enable agent-based system development through middleware services. The framework provides for scalable messaging among agents.

The Autonomous Multiagent Testbed [5] at the University of Washington supports seven miniature robots, a global vision system and a custom-designed infrared communication structure. The wireless aspect of the testbed gives the agents real autonomy.

The AgentCities testbed [7] supports FIPA agents and is intended to handle a very general service-oriented architecture for the dynamic search, discovery and composition of Semantic Web Services. This testbed also runs in 24/7 mode. As we shall explain in this paper, the ACTIVE Testbed under development at Global InfoTek to support the DARPA Coordinators Program has unique features not found in other testbeds.

1.2 Organization of Paper

This paper is organized as follows. Section 2 provides an overview of the ACTIVE testbed architecture and the layered services approach. Also discussed in Section 2 are the subsystems for the testbed, infrastructure and benchmarking. Section 3 gives a brief description of the Testbed Lab. Section 4 outlines our future work for the ACTIVE system.

2. ACTIVE TESTBED ARCHITECTURE AND SYSTEM

Global InfoTek (GITI) is a company dedicated to providing advanced middleware solutions to government and industry. It has a long history of successful projects with DARPA and other government agencies. The Coordinators program requires an industrial-strength testbed in which to load, initiate, run, monitor, and evaluate the agent-based schedules, execution traces, and overall quality of the performer team solutions.

2.1 The ACTIVE Testbed Architecture

The ACTIVE testbed is a robust, scalable and lightweight experimental framework to support scientific evaluation of Coordinators agents. It is a system-agnostic framework, which supports any Coordinators agent interface, and requires minimal effort to integrate with the testbed. It also provides agent-based tools to orchestrate Coordinators agents and supporting testbed services, rapidly set up experiments, and quickly reinitialize the baseline system to support repeatable experiments.

Figure 2 provides an overview of the ACTIVE services for testing and evaluation for the Coordinators Program. The Testbed

consists of three service layers: Test Harness, Command and Control (C&C), and Evaluation/Experimentation utilities. The CoABS Grid [6] middleware is used for agent communication. The Command and Control module manages the complex, distributed environment for the Coordinators agents and controls and monitors experiments. Evaluation/Utility Services include a Scenario Generator and a Centralized Scheduler as well as collecting and archiving test outputs for analysis and evaluation.

2.2 Test Harness

Figure 3 shows the ACTIVE layered services architecture, and Figure 4 shows the calling relationships among the services. Figure 5 presents an event-sequence diagram showing how the ACTIVE services interact in support of a test. The Test Harness layer encapsulates the agent tasking language, CTAEMS (Coordinators Task Analysis, Environment Modeling and Simulation) and the simulation engine, GMASS (GITI Multi Agent System Simulator). It also provides the interface for Coordinators agents to interact with the testbed services and with other Coordinators agents.

TAEMS (Task Analysis, Environment Modeling and Simulation) is a language designed to model tasks for software agents. TAEMS models the quantitative behavior of tasks, task activities and interactions between tasks [4]. CTAEMS was created by modifying TAEMS to meet the unique needs of the Coordinators Program. The primary modifications are:

- Removal of GUI infrastructure, preprocessor, and commitments
- Changes to core data structures, virtual nodes and parser
- Addition of exogenous events

There are three types of CTAEMS Nodes: method, task, and task group. Methods are the leaf nodes of the graph and represent primitive executable actions. These actions are grouped together as tasks. A task can contain one or more methods and represents a goal that can be accomplished by its method(s). A task group represents a higher-level goal that is accomplished by the tasks in its structure. Figure 1 shows a simplified example of a task group for building a model car.

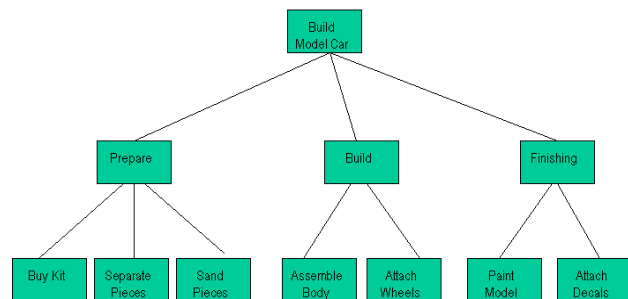


Figure 1. Hierarchical Task Structure

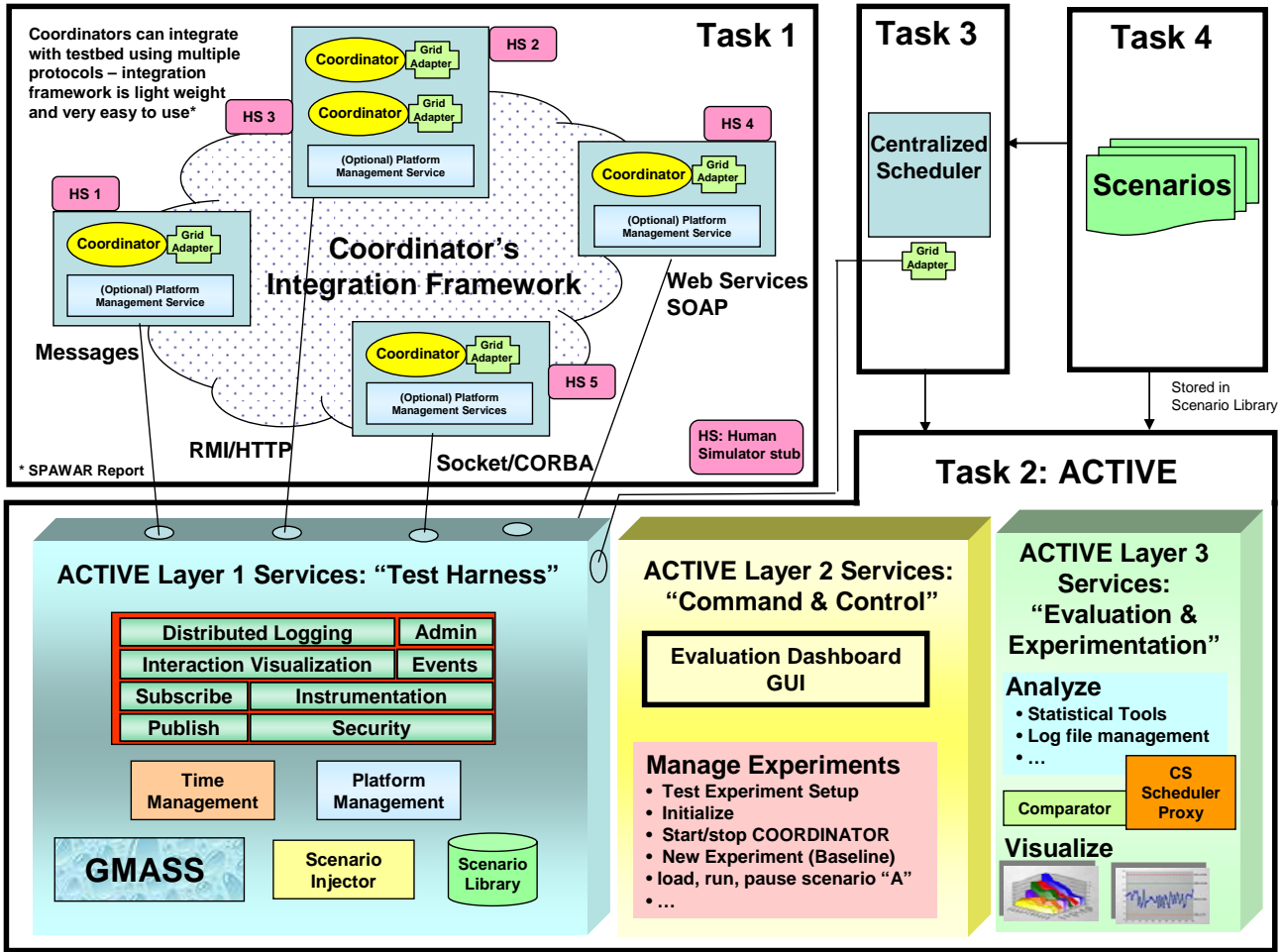


Figure 2. ACTIVE Services for Testing and Evaluation

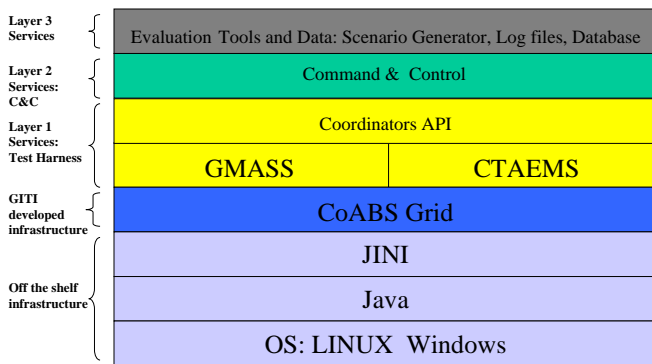


Figure 3. ACTIVE Layered Services Architecture

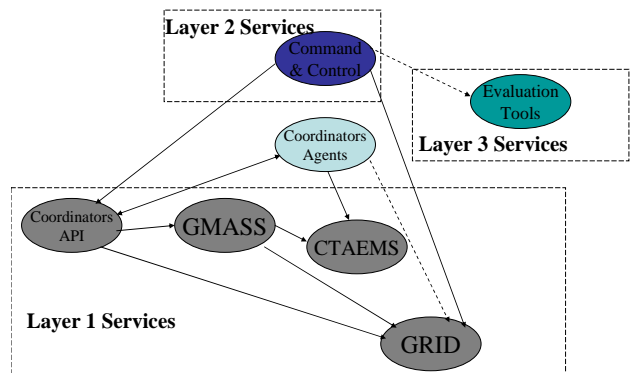


Figure 4. Relationships Among Services

GMASS (GITI Multi Agent System Simulator) provides the simulation engine for testing coordination and negotiation among teams of Coordinators agents. GMASS was created by modifying MASS (Multi Agent System Simulator) to meet the unique needs of the Coordinators project. The primary modifications are:

- New Time Management model
- Rewrite of core event handling
- Communication changed from sockets to CoABS Grid

The simulator itself runs as a single process. All Coordinators agents involved in the simulation are connected to the simulator through the CoABS Grid [6]. This allows the simulation to monitor all Coordinators agent communication with the simulator and with other Coordinators agents. Each Coordinators agent has its own subjective view of the world and its own goals. The simulator has the objective view of the world containing information that the Coordinators agents do not see. These views are created by the Scenario Generator based on input parameters.

GMASS simulates the Coordinators agent's method execution. The actions of the simulator are directed by a queue containing a time-ordered list of events. The simulator receives two basic types of messages from agents: Method Abort and Method Start. Each message causes a method to be added to or deleted from the event queue. At each pulse the simulator selects the correct events and realizes their effects.

A CTAEMS Scenario for the Coordinators program contains a hierarchy of tasks assigned to each Coordinators agent along with deadlines and constraints. There are two types of views of a CTAEMS scenario, objective and subjective. The objective view contains the entire correct scenario. The subjective view does not include everything that is contained in the objective view; it may also have some incorrect information. Agents only have access to a subjective view for their planning and interacting. The subjective view is different from the objective view because agents may have an imprecise model of what will actually happen, performance-wise when the method is executed. The objective view is created by the Scenario Generator. The objective views of the methods are stored in the simulator's CTAEMS database; the subjective views are created by the simulator and passed to the agents.

When an agent requests that the simulator start a method, the simulator retrieves the method from the objective CTAEMS structure. The first step of the simulator is to calculate the values of the quality, cost and duration that will be produced by this execution. A random number generator is used to calculate these values. Any event that occurs before a queued event is performed can affect the results of that event's execution

The simulator controls time simulation for the agents. A single unit of time is called a *pulse*. The simulator periodically sends a pulse to each of the agents. MASS manages time so that a pulse is complete after all agents have completed an action. The pulse time is not proportional to clock time or CPU time, but can be of varied length. A pulse ends when all agents have reported in. The MASS time management scheme has been altered for GMASS so that the pulse time is based on clock time and is the same length throughout a run. The length of a pulse is controlled by a setting in a configuration file.

GMASS time management provides a guarantee window that ensures that requests sent to the simulator within a time window in a given pulse will take effect within that pulse. Coordinators agents can run at any time they please and exchange messages within or outside the guarantee window. The simulator will process such requests within the given pulse period provided it can do so without delaying the next pulse. It will send a failure message to the requesting agent if it cannot do so.

The Coordinators API provides a single programming interface for accessing the simulator, representing the agent task structure, communicating among agents, and communicating between agents and the simulator. The API is both a convenience for developers of Coordinators agents and a tool for observing and instrumenting the Coordinators agents. The Coordinators API serves as a wrapper for GMASS and CTAEMS.

2.2.1 Agent Startup

All Coordinators agents are required to implement the Coordinator interface, which allows them to receive messages from GMASS and from other agents. It also provides the capability to be started by the ACTIVE testbed.

The methods used during startup are initialize and begin. Initialize tells the agent to start any processes and perform all initialization that it can without having scenario-specific information. Begin notifies the agent that the simulation is starting. It passes the CTAEMS task structure and other information necessary for the agent to begin the simulation.

2.2.2 Agent to GMASS Communication

Agents send requests to GMASS using method calls defined in the Simulation interface. When an agent is started by its Agent Controller, it is given an object that implements this interface. The Simulation interface allows agents to start methods and abort methods in GMASS.

2.2.3 GMASS to Agent Communication

The Coordinator interface allows agents to receive messages from GMASS (with `receiveMassMessage`) and from other agents (with `receiveMessage`). Messages that agents may receive from GMASS are defined in the `com.globalinfotek.coordinator.message` package. Changes in the CTAEMS task structure are defined by classes in the `com.globalinfotek.coordinator.ctaems.event` package. Agents are not permitted to communicate with one another until their begin methods have been called.

2.2.4 Agent Shutdown

When a simulation ends, agents will receive a disconnect message from the simulator. Upon receipt, the agent should shut down any external processes, close open files, etc. Once the agent is ready to exit, it should notify the Simulator by calling `Simulation.bye`. If another simulation is to be run, the command and control infrastructure will initialize and restart a new instance of the agent.

2.3 Command and Control

The primary requirements for the C&C system are to provide completely automated unattended testing on a large scale of up to thousands of tests. In addition, tests are to be run simultaneously on multiple subnets.

C&C provides services to manage the large-scale experiments using the Test Harness and collecting data for evaluation. Features include automation and remote control, installation and configuration, test definition and management, preflight check, control of test results, instrumentation, logging and data capture. Services include experiment setup, initialization, load and run a scenario.

C&C monitors the full execution of a trial, which is one execution of a test or group of tests. A test refers to a test specification that can be repeated many times and run for different teams. A trial refers to a specific test run at one particular time for one particular team.

The test lab configuration consists of a simulation host that runs the simulation engine, a C&C host that runs the C&C software and logging database, and one or more client workstations that run the Coordinators agents.

2.3.1 Preparation for Testing

There are several steps which must take place prior to testing:

- Create the .xml file with input parameters for generating the CTAEMS scenario file.
- Run the Scenario Generator to create the CTAEMS file containing the objective view.
- Run a utility to extract the simulator parameters and other info from the Scenario Generator input and output files.

2.3.2 Configuration and Initialization

In order to minimize time and effort required to prepare test hosts, C&C requires very little software on each test system. Only Java, Ant, and Bash Shell are required on each machine. All other software is pushed out by C&C just prior to running each trial. This software is prepared once for each type of system and is called a “mold.” A mold consists of the baseline home directory and all files required for a Coordinators agent or the simulator to run. This includes software, data, configuration files, scripts, and utilities. For each trial, the software is essentially reinstalled. This ensures that the software is always up to date and that each test is run from a consistent clean initial state. When the software to be tested is modified or updated, it only needs to be delivered and installed once - to the C&C host. Installation on all other systems will be done automatically at the time of testing by the C&C services.

The techniques used by C&C ensure that test results are reliable and not invalidated by files or processes left behind by previous tests or by failure to keep software up to date and consistent between systems. C&C also supports distribution of other software packages to all machines in the lab, or specific subsets of machines. This facilitates the process of updating machine software, and will be used to ensure software consistency between all machines in the lab.

When C&C is invoked, it loads the Test Definition and other configuration information. For each trial, C&C verifies that all required hosts are available and running. Each trial will require one simulation host and one client workstation for each Coordinators agent to be run. C&C then baselines each host, by clearing the home directory and copying over a “mold” for each type of system. .

2.3.3 Monitoring Network Status

C&C monitors the network and all nodes to verify that all agents are running, GMASS is properly registered with the CoABS Grid, and that the test has started. If a required service is not running, C&C will attempt to restart it.

2.3.4 System Preflight Check

C&C’s System Preflight Check utility verifies that the test environment is operating properly prior to execution of a test or experiment. It checks network connectivity and verifies that all required processes are running on the systems involved in the test. It checks for sufficient available disk space and system memory to execute the test and store logs.

2.3.5 Execution of a Test

The test definition files consists of a CTAEMS scenario file and a properties file. The properties file contains simulator parameters, the names of other configuration files, and the name of the CTAEMS file, generated by the scenario generator, contains the scenario (both the objective and subjective views). These files will be queued for processing by placing them in a specified directory. The CTAEMS file will be passed to the simulator by C&C and the simulator will provide the s agents with the subjective views only.

C&C will begin each trial by starting the CoABS Grid and then starting the simulator.

The C&C will use the Test Harness API to communicate with the Agent Conductor to initialize and start up all the s agents.

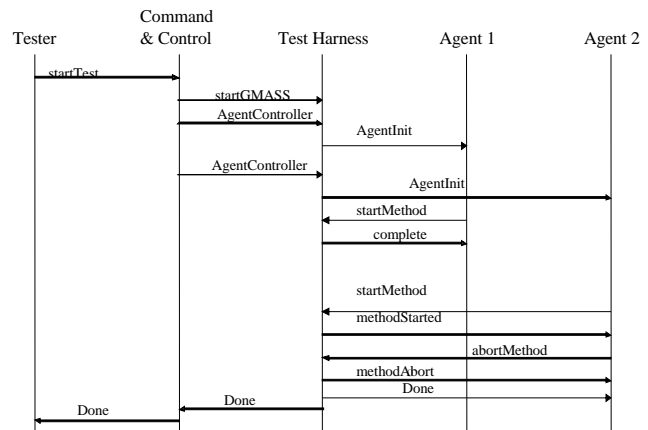


Figure 5. Single Test Event Sequence Diagram

2.3.6 Data Collection

Data is collected into logs during execution of each test. In order to minimize network traffic, logs are written to the local systems during the execution of the tests. After the test run is completed, the logs are collected, compressed, and moved to the C&C host. The original CTAEMS scenario files, property files and other configuration files are stored with the log files for use in evaluation. In the event of a failed test, the logs are still moved to the C&C host for analysis..

2.4 Evaluation

The ACTIVE Evaluation layer includes tools used to support evaluation of performance of Coordinators agents.

2.4.1 Scenario Generator

The Scenario Generator automatically generates scenarios based on the input parameters. The scenarios are expressed as CTAEMS structures. A scenario consists of 1 to 10 concurrent problems linked by a top-level SUM task. The input parameters are concerned with the number of problems in the scenario and specifications for window or deadline tightness and overlap, number of agents, nonlocal events, and uncertainty. Input parameters can be specified exactly, or can be randomly generated within a specified range. From the input parameters, the Scenario Generator produces a template file that includes all parameters and their default values and a CTAEMS text file that describes the scenario generated according to the input parameters.

2.4.2 Centralized Scheduler

The Centralized Scheduler is a program that solves a centralized coordination problem. It computes a set of solutions via an exhaustive search for experimentation and evaluation. The Scheduler operates on the same scenarios as the Coordinators agents. However, the Scheduler does not operate under the same time constraints as the Coordinators agents – it has unlimited time to create solutions. The results from the Centralized Scheduler will be used as a baseline for evaluating Coordinators agents.

2.4.3 Chronos

Chronos provides the time management for the Centralized Scheduler. The Centralized Scheduler can be used to determine whether a feasible initial schedule exists for a static scenario. However, in general, scenarios are dynamic – during the course of a scenario execution, several types of “unexpected” events may occur, such as method failures, new task arrivals, and deadline changes. Chronos integrates the centralized scheduler and GMASS, resulting in a centralized agent capable of handling these exogenous events. As with the distributed Coordinators agents, Chronos cannot see the future – dynamics are not known ahead of time, and past decisions (actions or non-actions) cannot be reversed. However, unlike the distributed agent teams, Chronos is not subject to time constraints between clicks, and Chronos is given the objective (rather than subjective) scenario view.

2.5 Infrastructure

2.5.1 CoABS GRID

The CoABS Grid [6] is middleware that integrates heterogeneous agent-based systems, object-based applications, and legacy systems. It includes a method-based application programming interface to register services, advertise their capabilities, discover services based on their capabilities, and send messages among those services. The Grid also provides a logging service, to log message traffic and other information, a security service to provide authentication, encryption, and secure communication; and event notification when agents register, deregister, or change their advertised attributes.

The Grid is built using Jini™ Network Technology developed by Sun Microsystems. Jini is a specification for service discovery and provides the robust and dynamic nature of the CoABS Grid. The Grid provides helper utility classes that are local to an agent and that shield the Grid user from the complexity of Jini. It also provides Jini services to facilitate agent communication and

logging. Although on the one hand, the Grid shields the user from Jini, on the other, it exposes the underlying Jini classes, so that the programmer can use Jini directly, if desired.

2.5.2 Labmin

Labmin, the GITI Lab Manager, is a set of tools for managing a dynamic test lab environment. Labmin was developed with open source software and utilities. The primary utilities are SSH, partimage, Python, GRUB bootloader, and the PXE framework.

Labmin provides for remote reboot, restart, and shutdown. Systems can be turned on using the PXE framework. Running machines can be rebooted or shut down using secure channel commands. Remote change of operating system is also supported by this utility. Individual partitions can be re-imaged either from a network store, or a file on the local drive. This feature can be used for a return to baseline to ensure identical initial states for test runs.

For system integrity, Labmin commands are authorized only after they have been authenticated using public key cryptography. All Labmin communication is encrypted using asymmetric key algorithms such as RSA and DSA.

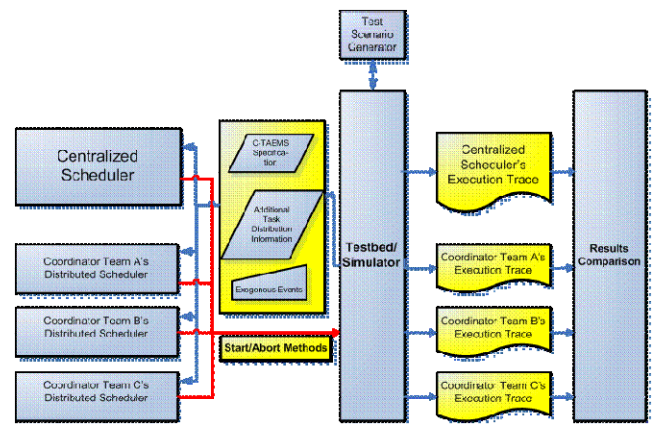


Figure 6. Centralized Scheduler in Context

2.6 Benchmarking of Coordinator Teams

Benchmarking of the Coordinators teams is done with the help of Centralized Scheduler. The conceptual benchmarking scheme is depicted in Figure 2. Multiple scenarios are prepared for evaluation by the evaluation team. Each scenario contains (1) CTAEMS specification of the requested distributed tasks/methods, (2) information on specific probabilistic outcomes of methods in terms of duration and quality, and (3) exogenous events. The input (1) is given as input in the beginning of each test to both the Coordinators teams and the Centralized Scheduler. The inputs (2) and (3) are released the Coordinators teams later, in real time, during the testing process. The centralized scheduler runs off-line. However, similar to the Coordinators teams, it is not allowed to use inputs (2) and (3) to schedule tasks/methods that occur before the information (2) and (3) becomes available on the simulated timeline. The Coordinators teams and the Centralized Scheduler submit start/abort method instructions to the Testbed simulator. The simulator executes the instructions which results in producing an execution trail for each of the Coordinators teams

and Centralized Scheduler. Each of the execution trails are assigned the total Quality measurement according to the CTAEMS specification. The Quality measurement of the Centralized Scheduler execution trail serves as the benchmark for the Coordinators teams. The Centralized Scheduler has the advantage of knowing the *objective view* in the CTAEMS specification, whereas the Coordinator teams know, at least initially, only the *subjective views*, i.e., parts of the CTAEMS specifications relevant to individual agents, and their environment. Also, the Coordinators teams need to produce decisions on starting/aborting tasks/method within strict time-limits, dictated by real-time constraints (enforced by the simulator), whereas the Centralized Scheduler has more time to make its scheduling decisions. However, due the volume of computations necessary in the testing process, the Centralized Scheduler is using a simple combinatorial algorithm, that sacrifices optimality for computational time. Thus, the quality of execution trails of the Centralized Scheduler may either outperform or under-perform the Coordinators teams' execution trails. In either case, the quality of execution trails produced by the Centralized Scheduler is used as a benchmark.

3. TESTBED LAB

The current Testbed Lab is configured with a separate subnet for each Coordinators Performer team, each of which contains 11 workstations. One of the machines on each subnet is a reserved host used for simulation execution and contains a 200 GB disk that will be used as a software and data repository. The 10 remaining machines are reserved for Coordinators agent execution. The current scenarios do contain not more than 10 agents, each agent on a Coordinators team can execute on its own machine. Figure 7 shows a diagram of the Lab Design. These machine clusters give high system/agent efficiency, enabling all three teams to execute scenarios independently in parallel, while the subnet topology ensures inter-team security. The ACTIVE Testbed software can support additional subnets, which may be required in later phases of the Coordinators program.

A server outside the individual subnets will serve as the C&C host to run the C&C services and host archived log data for evaluation. The C&C host is used to monitor the progress and operation of each cluster, manage the clusters, and manage large-scale experimentation and evaluation.

4. FUTURE WORK

The ACTIVE testbed will be advanced to support the continuing development of Coordinators agents. CTAEMs will be extended to include more non-local effects such as Disables and Hinders. The capability may be added to modify the task structure by adding exogenous events, removing nodes from the task structure, and parenting nodes in the task structure. Currently the communications model for the agents is completely reliable; it may be changed in future to be unreliable in order to more realistically model the environment. GMASS may be modified to support a different timing model. All software components will be enhanced as needed to support testing on a larger scale.

5. ACKNOWLEDGMENTS

This work is supported by the Defense Research Projects Agency in the Coordinators Program.

We also thank the performer teams for their helpful comments and suggestions.

6. REFERENCES

- [1] Cortese, E., Quarta, F. and Vitaglione, G. Scalability and Performance of JADE Message Transport System, Telecom Italia LAB, Naples, Italy, 2005.
- [2] Fullam, K.T., Klos, T.B., Muller, G., Sabater, J., Schlosser, A., Topol, Z., Barber, K.S., Rosenschein, J.S., Vercouter, L. and Voss, M., A Specification of the Agent Reputation and Trust (ART) Testbed: Experimentation and Competition for Trust in Agent Societies. in *Fourth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2005)*, (Utrecht, 2005), 512-518.
- [3] Horling, B., Lesser, V., Vincent, R., Wagner, T., Raja, A., Zhang, S., Decker, K. and Garvey, A. The TAEMS White Paper, <http://mas.cs.umass.edu/research/taems/white/>, UMASS, 2005.
- [4] Horling, B. and Vincent, R. TAEMS/MASS Technical Overview, <http://mas.cs.umass.edu/research/taems/Coordinators05-Taems-MASS.ppt>, UMASS, 2005.
- [5] Hoyt, S., McKennoch, S. and Bushnell, L.G. An Autonomous Multi-Agent Testbed using Infrared Wireless Communication and Localization, University of Washington, Seattle, WA, 2005, 6.
- [6] Kahn, M.L. and Della Torre Cicalese, C., CoABS Grid Scalability Experiments. in *2nd Intl Workshop on Infrastructure for Agents, MAS and Scalable MAS, Autonomous Agents 2001*, (Montreal, Canada, 2001).
- [7] Willmott, S.N., Thompson, S., Constantinescu, I., Bonnefoy, D., Charlton, P., Zhang, T. and Dale, J., Agent Based Dynamic Service Synthesis in Large-Scale Open Environments: A Report from the Trenches. in *AAMAS'04*, (New York, NY, 2004).

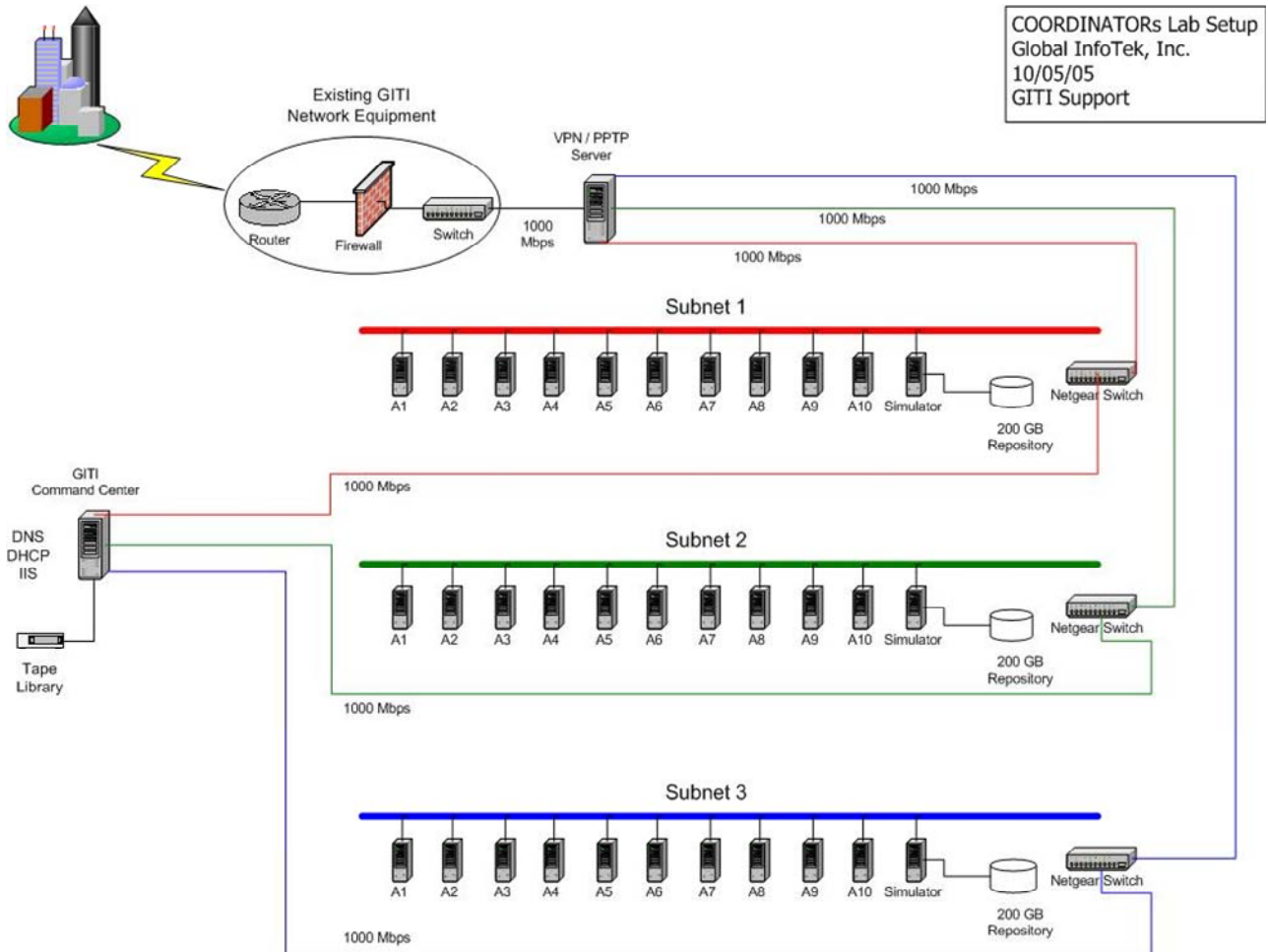


Figure 7. Testbed Lab Setup