Robustness and Security in a Mobile-Agent based Network Monitoring System *

A. Tripathi, M. Koka, S. Karanth, I. Osipkov, H. Talkad, T. Ahmed, D. Johnson, and S. Dier
Department of Computer Science, University of Minnesota, Minneapolis MN 55455

1. Introduction

We present here the mechanisms for self-recovery in Konark[1] [2], a mobile agent based system for monitoring network computing systems. An important aspect of our design is the use of the monitoring system's inherent capabilities to detect its own component failures. The Konark system is implemented using Ajanta [1]. Each host runs an agent server to support execution of mobile agents. The main functions of an agent fall into three broad categories: execution of local detector functions, event notification, and subscription and correlation of events from other agents. A monitor agent is launched with a variety of *detectors* for detecting basic events at its host. A detector can be triggered by events that are generated by other detectors using triggering dependencies maintained in a *trigger table*. Each detector has a handler object, which sends the generated events to their subscribers. System Management Agents (SMAs) are primarily responsible for managing system-wide monitoring policies. They typically run on secure hosts (possibly replicated for fault tolerance), launch monitor agents to hosts, and remotely control and modify these agents. The monitoring configuration is checkpointed, whenever it is updated by the administrators.

2. Robustness of the Monitoring System

Our monitoring system achieves robustness by incorporating mechanisms for *self-monitoring* and *self-configuration* at different levels of the system architecture. The event detection, correlation, and notification mechanisms are used as the basic building blocks for failure detection. Our design uses the notion of continuous periodic detection and notification of a failure event until the failed components causing it are repaired.

The recovery mechanisms are designed to address failures of hosts, agent servers, agents, and detectors. A host crash implies crash of its agent server and all agents running on it. Agent servers may also crash independent of their host crashes, or they may fail partially. The recovery of an agent server requires restoring all agents that are permanently installed at that server for performing monitoring functions. The agents themselves may fail in unpredictable ways. They may incur partial failures or fail completely and stop communicating with other components. When a detector, which runs as a separate thread within an agent, fails and stops performing its monitoring functions, it is remotely replaced with a new one by a remote agent performing recovery actions.

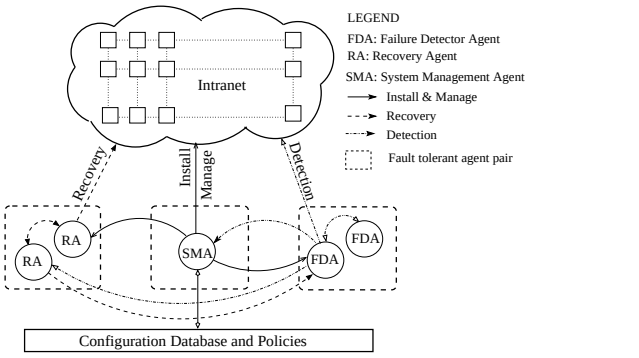

Figure 1. Recovery architecture

The state of an agent consists of various detectors and the events that it should subscribe to from other agents in the system. Information about any new subscriptions registered by an agent during its execution forms *soft state* and is reconstructed through interactions with other agents during recovery, where as the remaining configuration is checkpointed by SMAs. *Soft state* alleviates the checkpointing overhead. Most of the detectors are either state-less or they maintain soft state, i.e. they are designed to reconstruct the state when restarted. For example, the soft state of a detector processing log file events consists of the position offset in the log file for the last processed event. On restart, the detector obtains this information from the event databases.

Figure 1 shows the central components of this system for performing self-monitoring and recovery. In addition to the System Management Agent (SMA), recovery involves two other kinds of agents: *Failure Detection Agent* (FDA) and *Recovery Agent* (RA). Any monitoring agents can be used

* This work was supported by National Science Foundation grant ANI 0087514.

1 Further information can be found at http://www.cs.umn.edu/Ajanta

to perform these functionalities by adding required detectors and handlers. The System Management Agent installs a configuration for self-monitoring by dispatching Failure Detection Agents and Recovery Agents at various hosts.

Figure 2 illustrates the interactions among these components. Each agent is equipped with an *AgentAliveDetector*, which periodically checks the internal state of the agent and generates appropriate heart-beat *AgentAlive* events to indicate the health of the agent. An *AgentAlive* event contains a list of detectors which are functioning in the agent. This event also contains a *current configuration number*. Whenever an agent's configuration is changed with the addition or deletion of a detector, this number is incremented. This number is also incremented when an agent is re-launched on recovery. The purpose of this is two-fold: first to make sure that the subscribers of an *AgentAlive* message would note that the configuration has changed; second, it is used to ignore any failure events related to old configurations.

Each Failure Detection Agent subscribes to *AgentAlive* events from all agents in the system (step 1). If no such event is received from an agent over a pre-defined number of consecutive timeout periods, it generates an *AgentFailure* event (step 2). It keeps on generating such events until the agent is restarted and a heart-beat message is received, or the configuration information is altered to ignore that agent. When a heart-beat message is received, the Failure Detection Agent compares the list of detectors in the event with its configuration information. It generates an *AgentFailure* event if a detector is found to be missing.

A Recovery Agent implements recovery procedure in the handler associated with the *AgentFailure* event (step 4). The recovery action is executed by one of the Recovery Agents, as described below. The recovery agent performs the following kinds of recovery functions. On a detector failure at an agent, it tries to re-install that detector. On agent failure, it recreates the agent based on its most recent configuration information, and re-launches it to the target host. Before sending the agent, it makes sure that the target host, its agent server, and RMI registry are running. Otherwise it first tries to restart them. To recover a failed System Management Agent, it recreates it with the most recent checkpointed configuration state.

Two or more Failure Detector Agents execute at different nodes and monitor each other to make sure that the loss of a Failure Detection Agent is detected and a new agent is created in its place. A pair of recovery agents executing on different hosts subscribe to the failure events generated by the Failure Detection Agents. This pair works in primary-backup mode – i.e., only the agent in the primary mode initiates any required recovery action. Failure Detection Agents also monitor the Recovery Agent pair. The agent in the backup mode becomes the primary when an event indicating the primary agent's failure is received.

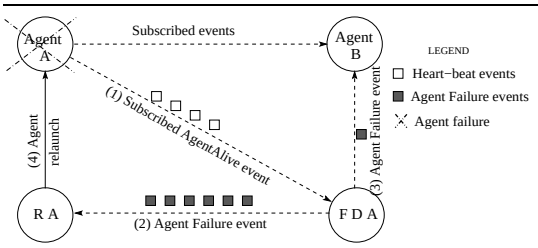

Figure 2. Recovery of a Monitoring Agent

When an agent is restarted at a host, it does not have any information about its subscriber agents. In our model, the subscribers are required to register themselves with the event publishers to get the events of interest. Therefore, when an agent failure is detected, we need to inform all of its subscribers of this failure event. To facilitate this, each *AgentAlive* event from an agent also contains the list of its current subscribers. On detecting an agent failure, the Failure Detection Agent sends the *AgentFailure* event to all of its subscribers (step 3). Each agent maintains a list of events that it subscribes to from other agents. It also maintains a list of *outstanding-subscriptions*, containing the list of agents with whom it has not yet succeeded in registering its subscriptions. Failed agents are added to this list for re-subscriptions. *EventSubscriptionThread* in an agent periodically attempts to register subscriptions with the agents in this list and remove them from the list on success.

3. Security of the Monitoring System

Security becomes an important issue as monitoring system needs to run in untrusted domains and security requirements also arises as a direct result of system capabilities. Security is ensured by using agent server's capabilities to create *distinct protection domains* for agents to execute, and to enforce agent admission control and resource access policies, using the set of unforgeable credentials agents possess. These policies can be changed dynamically. Agents also enforce policies specified by their owners, maintain a list of authorized subscribers and senders with whom they communicate using authenticated inter-agent communication provided by Ajanta.

References

[1] N. Karnik and A. Tripathi. Security in the Ajanta Mobile Agent System. *Software Practice and Experience*, 31(4):301–329, April 2001.

[2] A. R. Tripathi, M. Koka, S. Karanth, A. Pathak, and T. Ahmed. Secure Multi-Agent Coordination in a Network Monitoring System. In *Software Engineering for Large-Scale Multi-Agent Systems, Springer, LNCS 2603*, pages 251– 266, 2003.