MAX-Sense: A Mobile Agent Approach to Active and Adaptive Distributed Monitoring

Imad Riachi, Mounir Mouawad, Nada Abou-Naccoul, Georges Khazen, Hassan Artail
Computer and Electrical Engineering Department
Faculty of Engineering and Architecture
American University of Beirut
{ijr00, mjm04, nga09, gak07, hartail}@aub.edu.lb

Abstract

In this paper we are proposing a new distributed system architecture to address the problem of very large scale sensor network monitoring. By dynamically aggregating and processing data from sensor networks, databases and other sources, the system can detect hazardous patterns soon enough to prevent serious human damage. The system involves discovering and studying the underlying network topology then dispatching Mobile Agents (MA) for collection, analysis, and processing of data that is packed in XML format. This paper describes the architecture and its elements, including the employed mobile agents, which are designed to respond to different levels of sensed security threats in a timely manner. We also present a prototype system implementation that was developed to test the performance of the developed architecture.

1. Introduction

In most of the distributed data monitoring approaches, since no processing is done at the sensors' data acquisition system, a continuous flow of sensor data has to be sent to a central processing unit, which has to continuously process the sensor data streams to take decisions accordingly.

Two major issues arise as a result of this setup. First, since the processing unit has to process the large volumes of incoming data, it tends to provide limited analysis and reporting capabilities. Second, such systems tend to score low on the scalability scale as more sensors means additional data that the processing unit has to handle and more network bandwidth is consumed.

The ultimate solution is thought to be based on mobile agents, where algorithms were proposed to find the optimal location where the agent ought to run. No complete architecture of such systems was provided though. Focus was rather on the relevance of the mobile agent approach while the implementation-related issues were left open for further study [1].

In this paper, we describe a system for distributed monitoring based on mobile agents, which we call MAX-Sense. Simulations of an actual environment within the context of a real-life application were used to generate promising results that support the viability of the approach.

The rest of the paper is organized as follows. In Section 2 we present the system architecture and the described the mode of operation in Section 3. In Section 4 we show the implementation results and then conclude the paper in Section 5 with a discussion of the results obtained.

2. System Architecture

2.1. System Topology

At the lowest level the sensors collect raw data about the environment and send it to data acquisition units (sinks), which perform required signal conditioning to generate readable data for the processing units (computers, laptops, PDAs…), which we refer to as access points or APs. The latter provide the necessary execution environment for the mobile agents to use for analyzing the data and taking corresponding actions. At a higher level, the networking hardware provides the connectivity and finally, the server hosts the software responsible for managing the mobile agents and providing the user interface for monitoring and reporting status.

2.2. Mobile Agents

The agents represent the workhorse of the system as they are responsible for closely monitoring the status of the overall system by inspecting the aggregated readings from the sensors and taking appropriate responses. Most of the responses relate to progressively increasing the level of monitoring and data inspection. There are four types of agents that are designed to perform specialized tasks, including adapting to changes in security conditions as reflected in the sensor readings:

Discovery MA (DMA): This agent is dispatched into the network whenever a new network is connected. It roams the network indefinitely and retrieves the configuration of each AP, relating to connected sensors, databases, and offered services (which are listed in a resources.xml file). The DMA is programmed to handle certain abnormal situations such as buffering readings locally on the AP when connections are lost.

Reading MA (RMA): Also referred to as Monitor MA, it gets launched with the DMA and also roams the network indefinitely, to collect readings of AP channels, to perform preliminary checking (e.g., out of range values), and to send samples of the collected readings to the server. This agent keeps track of the last collected readings on each AP through timestamps (in its briefcase) so as to avoid taking duplicate readings.

Channel MA (CMA): It is a channel-specific agent and its purpose is to a keep a closer eye, for a given time, on one or very few channels (sensors) whose readings deviated from normal levels, as reported by the RMA.

Pattern Specific MA (PMA): It is dedicated to specific patterns (potential threats) and its operation space may be restricted to one or more specific APs. It looks for patterns according to criteria specified in its briefcase and generates AP level alerts in accordance with criteria that are specific to each AP. Moreover, at the end of its path, the PMA performs network level analysis and may throw network-level alerts. When alerts are thrown, it may ask the AP to clone it, upon which it divides its route between it and its clone in accordance with related briefcase rules. The process of cloning may repeat recursively up to an extreme case where one PMA is assigned to every AP. When the pattern recedes, the process reverts back to the case where all PMAs are terminated.

The communication of the agents with the server is done using XML-files. As was implied, every mobile agent carries with it a briefcase, which is an XML file that travels with the Mobile Agent and specifies its tasks and path along with other information related to its operation. Although the briefcase is specific to each type of agents, there are common parts that are shared by all agents:

Agent Info: includes the agent type, agent ID, IP and port number of officer (dedicated process on the server – more on this later).

Path Info: consists of a list of APs, each defined by a MAC address, IP, port number, and residence time (how long the MA should stay on each AP). To avoid redundant readings, each AP on the path is associated with a timestamp referring to the time it was last visited (not applicable to the PMA).

In addition to the above, the briefcase of the CMA includes a *termination count* (*TC*) that determines its lifetime and two levels of categories for sensor readings. For the briefcase of the PMA, the pattern and its related data (criteria, min value, max value etc.) are specified in addition to the TC. This briefcase further contains a list of all APs that generated warnings or alerts and basic information about existing duplicates and their paths.

2.3. Server Modules

The server consists of five main processes, in addition to the user interface and the database.

Processor: This module creates a new thread each time a new network is added to the system. Its purpose is to process readings sent by the RMA and to check for deviations from what is considered normal level. The normal level and warning plus alert thresholds are values that are sent by the DMA upon network discovery time or afterwards when they change on the AP. It is this process that decides to dispatch a CMA when certain readings are deemed suspicious. Actually, the processor sends a request to the agent factory for deploying the CMA and specifies in the same request the destination APs and the channels that need monitoring, which end up being written to the CMA's briefcase. Furthermore, the factory writes to the CMA's briefcase a termination count that specifies the number of round trips that the agent will make through the designated APs before self-terminating. Once this takes place, the processor uses the collected readings to perform curve fitting, which enables it to estimate future readings. The latter are then used to detect if given patterns are likely to occur and whether precautionary actions should be executed. To ascertain that certain patterns are in fact taking place, a PMA gets dispatched to verify the occurrence of suspected patterns and to issue warnings or alerts on the APs (or via connected hardware) to circumvent possible security threats or malfunctions.

Officers: These processes are responsible for dispatching mobile agents to their assigned networks and for handling the aggregated data received from them.. The officers receive readings from the RMA and CMAs, alerts from the PMAs, and writes this data to the server's database. They also handle all notifications for mobile agents' duplication (cloning) and deletion (abortion) and are always aware of the locations of their respective mobile agents. This enables officers to communicate relevant updates to their agents through their briefcases. In all, an officer acts as a proxy of the concerned group of agents at the server.

Access Point (AP) Manager: This process receives updates about network topology changes. For instance, when a new AP is added, related information such as IP address, MAC Address, and network number are retrieved from the message and stored in the database. Queries are then run against the database to determine properties of available APs at runtime.

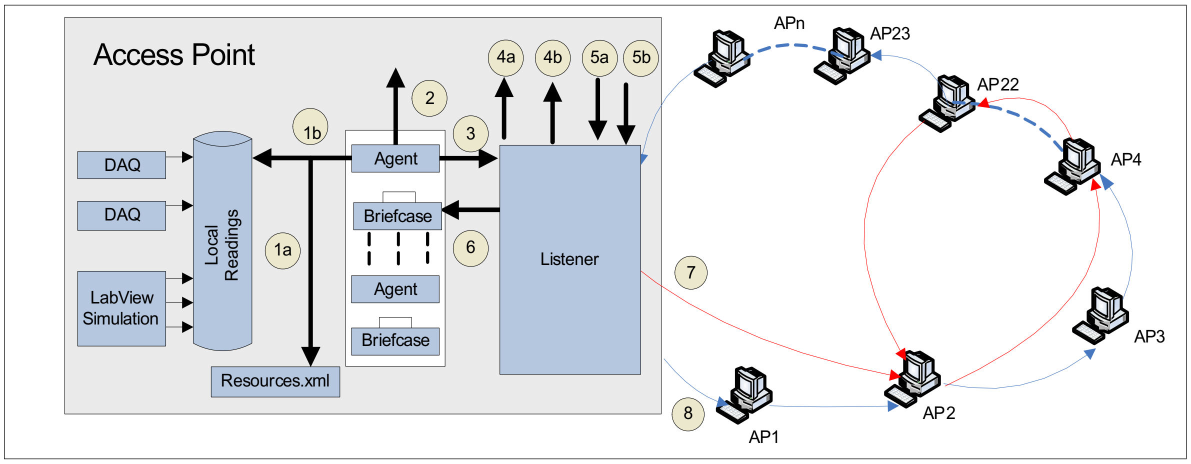

Figure 1. Mobile agents activity and movement

Coordinator: This process was deemed necessary during implementation and acts as a router for the other processes. It is responsible for synchronizing and coordinating server tasks as well as routing incoming messages. It knows how to forward messages to the appropriate processes based on the label of the message and the extension bytes (when applicable).

Agent Factory: This process is responsible for all requests of agent creation and management and launches when the system is initially started.

Database: It consists of tables that store all data related to network topology, available resources, types of readings, patterns detected, agents dispatched, alert conditions, and so on.

3. Mode of Operation

Whenever a new access point is added to the network, it is supposed to send a notification message to the server informing it about its address. If there is no DMA roaming that particular network, a new one is dispatched. Otherwise, the briefcase of the existing DMA is updated by the corresponding officer to include the new AP in its path. Subsequently, the DMA sends the server configuration information regarding the attached sensors and later, sends updates when the configuration changes. Different mobile agents are then sent out, duplicated, or removed depending on the aggregated readings and their relevance to particular patterns. The autonomous and dynamic behavior of each mobile agent is augmented by the briefcase it carries, which fully specifies its functions and the path it should follow. Data analyses are performed both at the server level by the processor and at the network level by the pattern specific agents (PMAs). Should the existence of PMA-detected patterns get confirmed by the processor at the server, a state of alert is declared and subsequent actions are initiated.

For the system to be scalable, the time to monitor the system should not continuously increase with the number of hops per network. Our design takes this into account by adopting a simple convention referred to as the network partitioning scheme to establish an upper bound on the time between two consecutive readings for a given sensor, which corresponds to the round trip time (RTT) of the monitoring agent (RMA).

At the Access Point, a listener process was implemented to manage agents. A different thread is created to handle every incoming agent and to provide it with services. One of these services has to do with shipping the agent's code to a given destination, e.g., another AP. If it cannot connect to the next AP on the Mobile Agent's path, the listener makes a local broadcast to determine the AP's new IP address. If this procedure fails, it moves the agent to the next AP specified in the list.

As indicated in Figure 1, the listener has to send the mobile agent to the next hop on its path (step 8). It does so after notifying the officer of the MA's current position (step 4a) and inserting any briefcase updates (step 6) received by the corresponding officer (step 5a). The listener uses a `backup.xml` file to buffer all data sent by the agents whenever the connection to the officer is down and then periodically checks the connection status, when it is reestablished, it sends the content of the file to the officer. Furthermore, when cloning occurs, the listener builds the clone's briefcase, and sends the clone to its first hop (step 7).

4. Results

The programming Language used to implement the system was C# and the sensors were simulated using LabView from National Instruments. Sub-networks included Pentium4 computers running Windows XP and having 512 MB of RAM, and connected through a 100Mbps switch. We tested our system during daytime and under normal network traffic conditions to obtain more realistic results. The access points were designed so as to allow the administrator to describe their attached sensors in a `resources.xml` file through a user interface that was designed for this purpose.

The average time an agent spent on one AP before it transferred to the next AP was less than 2.5 seconds when there were up to five concurrent agents on the same AP.

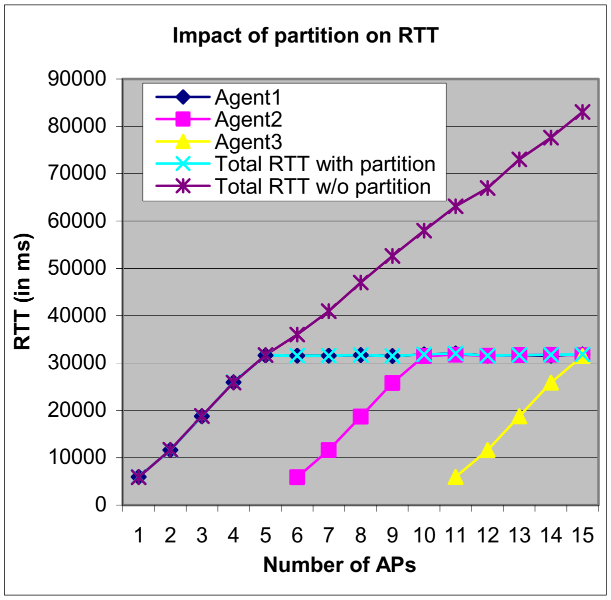

Figure 2. RTTs versus number of APs

To demonstrate the advantage of partitioning a network when deploying discovery or reading mobile agents, we linearly increased the number of APs on a network and observed the total round trip time (RTT) (Figure 2). First we plotted the RTT for the "no network partitioning" scenario and obviously observed a linear increase of the RTT as the number of hops increased (see RTT w/o Partitions plot).

To contrast the latter case with our system, we set the discovery and reading agents to be deployed for a maximum of 5 hops under normal conditional, resulting in $\lceil M/5 \rceil$ agents deployed for M hops (the symbol $\lceil\,\rceil$ signifies rounding up). We plotted the RTT for each new agent deployed as the number of hops increased with and without partitioning).

By using a test run that mimics the occurrence of a network level alert (corresponding to a suspicious detectable activity), we recorded all the updates sent by the agents to the server on a given network and plotted the results in Figure 3. We also compared the case when we have enabled cloning of an agent and the case when it was disabled. Figure 3 clearly shows that using agent cloning (duplication) enables closer observation of the sensor values in the case of when suspicious events were detected. The deletion of the clone agent after the suspicious activity has disappeared guarantees maximal efficiency of the system through keeping the overall network traffic to a minimum in cases of non-emergency.

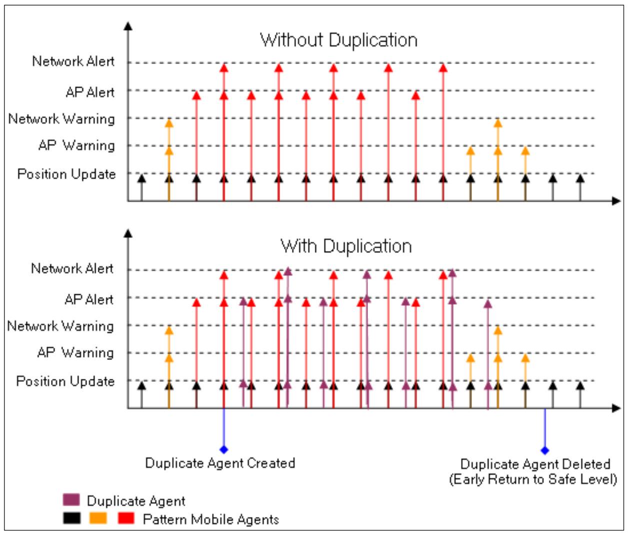

Figure 3. Messages sent to the server

5. Conclusion

The architecture proposed in this paper was proven to adapt and react to changing environments while detecting potential threats on a large scale and in real time. While calling for a fast response, MAX-Sense's distributed architecture supports a fault tolerant, autonomous and highly adaptive system's behavior. The developed system also demonstrated flexibility through the employment of mobile agents. Modifications to the preprocessing that has to be done locally on the access point can be altered just by modifying the mobile agent's behavior through the rules in its briefcase without having to introduce any changes locally on every single AP. The results have also shown that the proposed architecture is highly scalable, thus it can be applied to increasingly large networks without any performance fallbacks.

Our simple but representative implementation proves that mobile agents are able to dramatically increase the performance of large-scale sensor network monitoring and allow for the ease of configuring

6. References

[1] A. Liotta, G. Pavlou, G. Knight, Exploiting Agent Mobility for Large Scale Network Monitoring, IEEE Network, special issue on Applicability of Mobile Agents to Telecommunications, Vol. 16, No. 3, IEEE, May/June 2002