BATTLEFIELD NETWORK APPLICATIONS OF THE SHAMAN MANAGEMENT SYSTEM

Adarshpal S. Sethi
Dong Zhu
Vasil Hnatyshin
Prasanth Kokati

Department of Computer and Information Sciences
University of Delaware, Newark, DE 19716
{*sethi, dzhu, vasil, kokati*}*@cis.udel.edu*

ABSTRACT

SHAMAN (Spreadsheet-based Hierarchical Architecture for MANagement) is a novel management framework developed at the University of Delaware; it extends the traditional flat SNMP management model to a hierarchical architecture. Effective management of battlefield networks requires such a hierarchical management architecture wherein managers can dynamically delegate management tasks to intermediate managers. The SHAMAN framework includes a spreadsheet-based intermediate manager with a scripting language and MIB, a polling subsystem, and an event model; a prototype implementation of the system is available. Our research has explored several applications of the SHAMAN system to tactical battlefield networks for the US Army. These applications described in this paper include a Location Management application, an application to reconfigure dynamically changing topology of Tactical Internets, and another application to interface with OPNET simulations of battlefield networks.

Keywords: Network Management, Hierarchical Management, SNMP, Tactical Internet, Battlefield Networks, Location Management.

INTRODUCTION

One of the significant achievements of the ATIRP Consortium in Technical Factor 2 (Tactical/Strategic Interoperability) has been the design and development of an integrated framework for hierarchical management called SHAMAN (**S**preadsheet-based **H**ierarchical **A**rchitecture for **MAN**agement). This management system developed at the Network Management Laboratory of the University of Delaware incorporates management by delegation concepts into the Internet management framework to facilitate the management of distributed systems [1], [2], [3], [4]. This architecture allows a manager to delegate routine management tasks to an intermediate manager and facilitates user configurability of management information and control in a value-added manner.

A hierarchical management strategy is an effective means of managing the large and complex internetworks that are in use today [5]. The need for hierarchical management is particularly acute in tactical battlefield networks which are expected to have tens of thousands of nodes. Unfortunately, the most popular management framework in use today, the SNMP framework [6], [7], [8], only supports the flat management model. The framework provides no means for managers to delegate tasks to intermediate managers or for peer-to-peer communication between intermediate managers during the execution of these tasks. SHAMAN provides this much-needed capability to the Internet management framework.

This paper provides a brief introduction to the SHAMAN architecture and also describes how it can be used for the management of tactical battlefield networks. We start in Section 2 with a brief overview of SHAMAN. Section 3 describes a Location Management Application and its interfacing to an OPNET simulation, while Section 4 presents a Topology Reconfiguration Application. Finally, Section 5 presents the conclusions.

Prepared through collaborative participation in the Advanced Telecommunications/Information Distribution Research Program (ATIRP) Consortium sponsored by the U.S. Army Research Laboratory under the Federated Laboratory Program Cooperative Agreement DAAL01-96-2-0002. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon.

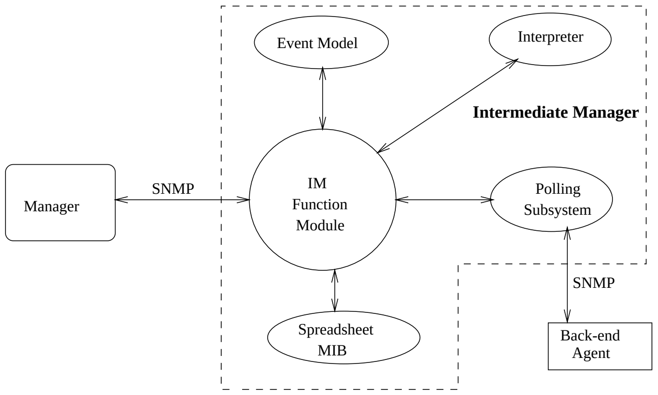

Fig. 1. Prototype Implementation of SHAMAN

I. Overview of SHAMAN

SHAMAN allows a manager to delegate tasks to an intermediate manager (IM) by downloading scripts expressing these tasks into a spreadsheet-like structure of the IM called the Spreadsheet MIB [9], [10]. This MIB is divided into a two-dimensional structure of cells called a spreadsheet, with each cell having a control part that stores the script and a data part that contains the result of executing the script. One IM can support multiple spreadsheets.

The spreadsheet MIB implements the spreadsheets using SNMP tables. All operations on the spreadsheet including a manager's downloading of scripts into the control parts and accessing the results in the data part are carried out via the Get and Set operations of the SNMP protocol. User operations on cells map to operations on tables that are part of this MIB. When the IM receives an SNMP request from the manager that translates to an operation on a cell, the IM performs the necessary operations on the spreadsheet MIB to implement the cell abstraction. Once the request has been carried out, the IM returns a response to the manager that requested the cell operation.

The scripts in SHAMAN are written in a special language called the Spreadsheet Scripting Language (SSL). This interpreted language contains features that facilitate the development of procedural scripts as well as event specifications. The language includes

- procedural language related features including operators, variable support, and control flow constructs
- network management specific features including polling specification and management variable access
- paradigm specific features including cell access, retrieval, modification, and multiple value access
- event model related features including event and event dependency specification

A prototype implementation of SHAMAN has been developed at the University of Delaware and is available on the WWW at the URL

http://www.cis.udel.edu/∼shaman

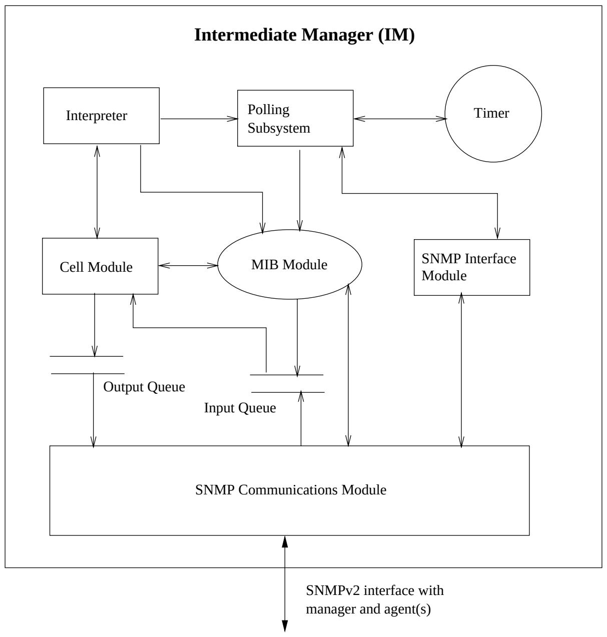

Figure 1 depicts the various modules that constitute the Intermediate Manager (IM) in the prototype implementation of SHAMAN. Figure I shows the software architecture of the IM and the interdependencies of the various modules that constitute the IM. Among these modules, the MIB Module, the Interpreter Module, and the Cell Module together implement three of the logical components of the IM. The Polling Subsystem implements the polling of the agents. The other modules perform support functions like timer services and providing a communication interface for polling the agents.

II. Location Management in Battlefield Networks

Consider a scenario of location management for mobile nodes in a battlefield network,. in which a group of nodes individually move on a battlefield according to the needs of the situation. Each node has an SNMP-manageable MIB with variables for the x and y coordinates of the current node position, and other variables corresponding to various quantities of interest, such as amount of fuel or ammunition remaining in the node. Each node requires to be periodically monitored by a manager who keeps track

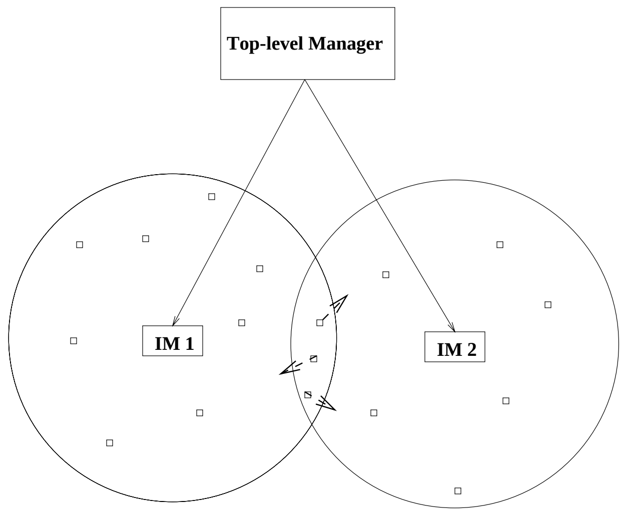

Fig. 2. Hierarchical Location Management for Mobile Nodes in a Battlefield Network

of the current location of the node and the amounts of fuel and ammunition left, and which may take appropriate action if these amounts fall below specified limits.

The total number of such nodes to be managed may be too large for a single manager to handle. Moreover, there may be distance constraints so that we may wish to have a node be managed by a manager that is located close by. Figure 3 depicts a hierarchical management solution using the SHAMAN approach that is appropriate for this situation. We designate two Intermediate Managers, named IM1 and IM2, with management authority over nodes that are within their spheres of communication as shown by the circles in the figure. Each IM periodically polls each node within its management domain to obtain its current variable values. If any action is required for the fuel or ammunition, then the top-level manager is informed.

As the nodes in this system move around, they may migrate from the management domain of one IM to the domain of the other. This may necessitate a "handoff" of the management authority over this node to the second manager. The need for a handoff may be detected by the IM responsible for each node. Each time a node's location is polled, it can be determined if the node has entered an intermediate zone (shown in the figure as the intersection of the two management domains). If it has, and if it is rapidly moving towards the second zone, the top-level manager is informed who then initiates the handoff of the node to the second IM.

Even though the example presented here is somewhat simplistic and a real-life situation has to consider other factors such as failures of intermediate managers, it serves to illustrate the power and utility of hierarchical management. This example has been programmed into SHAMAN in the form of a demo which is available at the SHAMAN web page. In this example, for which scripts have been written in the Spreadsheet Scripting Language (SSL), domains are defined for two Intermediate Managers (IMs) separated by vertical lines with a common overlapping portion. Each IM periodically polls the nodes which are in its own domain. The top-level manager can get an IM to either start or stop the polling of a given node by activating or deactivating the appropriate scripts in that IM. When a node's poll reveals a new position for that node, this event triggers execution of another script that determines if the node is still within the IM's domain, and also computes its direction of travel and its velocity. If the node has entered the overlapping zone between the IM's and is traveling quickly towards the other IM's zone, then a handoff is initiated immediately; if it is traveling slowly, a handoff is performed only after it has actually entered the other IM's zone. To perform the handoff, the top-level manager is informed who then activates the script for that node in the other IM and deactivates it in the first IM. It is also possible in the SHAMAN architecture for the two IM's to directly communicate with each other and perform the handoff without involving the top-level manager.

Since it is very difficult to demonstrate applications such as Location Management on a real system with more than a few nodes, we must resort to simulation in order to study their scalability to networks with large numbers of nodes. Our partner in the ATIRP Consortium, Motorola, has developed simulation models of realistic battlefield situations that include knowledge and modeling of terrain information. These simulations are implemented using OPNET [11] and demonstrate node mobility in different types of terrain and the effect on communication patterns [12]. We have developed a simulated MIB interface that allows us to connect the OPNET simulations with the SHAMAN management system [13].

III. Topology Reconfiguration with SHAMAN

A tactical battlefield network is characterized by a sporadically changing topology, due to the sudden appearance and disappearance of the network elements (e.g., routers, switches, links, etc.). Thus, an important component of the an adaptive configuration management system is to generate/re-generate an appropriate network connectivity after a change has occurred in the underlying network (i.e., adapt and respond to network triggers). Broadly speaking, while many schemes exist to generate network topology the complexity of the scheme increases in direct relation to the degree of optimality required. Strictly optimal solutions may not only be too complex, but also may be intractable. On the other hand, completely ad-hoc solutions may not yield consistent results.

Our partners in the ATIRP Consortium, Telcordia, have designed a heuristic algorithm to compute the new topology in such a situation [14], [15]. This algorithm applies a number of heuristics to generate connectivity within a Tactical Internet (TI) which are based on using the well-known Dijkstra's shortest-path algorithm to compute the shortest path tree from a given node to all other nodes within the network. This shortest path algorithm is modified to accommodate the constraints that are imposed on the configuration by the application. Three constraints that are used are:

- Every node in the network must be able to communicate with every other node via at least one path that is less than H hops long.
- Every communicating node pair has more than one possible routes through the underlying network (to provide redundancy in case there are failures).
- There is a a maximum limit on the number of other nodes with which any given node has direct connectivity.

To implement this algorithm in SHAMAN, we have planned to use a structure of cells in the spreadsheet which will make it easy to modularize the different logical components of the algorithm. The first column of cells are used for external interaction from the SHAMAN entity, i.e., for communication with the top-level manager. This column includes a cell used as the trigger for topology generation. The manager will do an SNMPGet operation on this cell to command the SHAMAN IM to begin topology reconfiguration when it is necessary to do so. Another cell is used to store the generated topology; this topology is then set into an agent MIB that is used to drive the input to the directory service system from where it is accessible to the manager.

A second column of cells stores global parameters and data. These include the number of nodes in the network, the node addresses, the maximum limit on nodal connectivity, and the limit on the maximum length of the shortest path between two nodes. The third column contains local structures to be used by the various components of the algorithm. These are the node positions (which are polled from the agents in the nodes, or may be obtained from the DSS), the path cost matrix, the direct link costs, and the predecessor of each node in the shortest path tree.

The remaining columns and cells in SHAMAN's spreadsheet are used for the code corresponding to the algorithm's components. The first of these is the Chain generation function, which initially generates chains between the nodes that are as long as possible without violating the maximum length constraints. The second is Dijkstra's algorithm which finds the shortest paths from each node to every other node. The third component is a function to create shorter paths whenever the algorithm has paths that are unacceptably long. This is done by adding direct links between the nodes so that path lengths may be reduced. The final component is a function to perform checks on the nodal degree constraint.

Implementation of this algorithm in SHAMAN is currently underway and we hope to be able to give a demonsration of the SHAMAN part of the system at the upcoming Annual Conference. The integration of this with the rest of the system being developed by our other task partners will be done shortly afterwards, and the complete joint demo should be available for demonstration before the end of the project.

The remaining columns and cells in SHAMAN's spreadsheet are used for the code corresponding to the algorithm's components. The first of these is the Chain generation function, which initially generates chains between the nodes that are as long as possible without violating the maximum length constraints. The second is Dijkstra's algorithm which finds the shortest paths from each node to every other node. The third component is a function to create shorter paths whenever the algorithm has paths that are unacceptably long. This is done by adding direct links between the nodes so that path lengths may be reduced. The final component is a function to perform checks on the nodal degree constraint.

Implementation of this algorithm in SHAMAN is currently underway and we hope to be able to give a demonsration of the SHAMAN part of the system at the upcoming Annual Conference. The integration of this with the

rest of the system being developed by our other task partners will be done shortly afterwards, and the complete joint demo should be available for demonstration before the end of the project.

IV. Conclusions

In conclusion, one of the significant achievements of the ATIRP Consortium in the area of network management has been the development of the SHAMAN system. SHAMAN is an architecture and framework for hierarchical management of networks that can be applied to implement various management applications in the Army's tactical battlefield networks. We have described its utility for location management of mobile nodes in a battlefield environment and for a configuration management application that can be used to reconfigure network topology when node positions have changed. A prototype implementation of SHAMAN is available and will soon be ported to the ARL Testbed so it can be used for other management tasks and applications within the Testbed.

The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government.

REFERENCES

[1] P. Kalyanasundaram, A.S. Sethi, and C. Sherwin. Design of A Spreadsheet Paradigm for Network Management. In *Proceedings of the 7th IFIP/IEEE Workshop on Distributed Systems: Operations and Management*, L'Aquila, Italy, October 1996.

[2] P. Kalyanasundaram, A.S. Sethi, C. Sherwin, and D. Zhu. A Spreadsheet-based Scripting Environment for SNMP. In A. Lazar, R. Saracco, and R. Stadler, editors, *Integrated Network Management V*, pages 752–765. Chapman and Hall, London, 1997.

[3] A.S. Sethi, P. Kalyanasundaram, C. Sherwin, and D. Zhu. A Hierarchical Management Framework for Battlefield Network Management. In *Proceedings of MILCOM '97, IEEE Military Communications Conference*, Monterey, CA, November 1997.

[4] A.S. Sethi, P. Kalyanasundaram, C. Sherwin, D. Zhu, and S. Pakala. Battlefield location management applications of SHAMAN. In *Proceedings of MILCOM '98, IEEE Military Communications Con ference*, Boston, MA, November 1998.

[5] A.S. Sethi, Y. Raynaud, and F. Faure-Vincent, editors. *Integrated Network Management IV*. Chapman and Hall, London, 1995.

[6] J. D. Case, M. S. Fedor, M. L. Schoffstall, and C. Davin. *Simple Network Management Protocol (RFC 1157)*, May 1990.

[7] J. Case, K. McCloghrie, M. Rose, and S. Waldbusser. *Protocol Operations for Version 2 of the Simple Network Management Protocol (SNMPv2) (RFC 1905)*, January 1996.

[8] M. T. Rose and K. McCloghrie. *Structure and Identification of Management Information for TCP/IP based internets (RFC 1155)*, May 1990.

[9] A.S. Sethi, P. Kalyanasundaram, C. Sherwin, and D. Zhu. A Spreadsheet-Based SNMP Scripting Environment for Battlefield Network Management. In *Proceedings of the First ARL/ATIRP Annual Conference*, pages 251–256, College Park, MD, January 1997.

[10] A.S. Sethi, P. Kalyanasundaram, C. Sherwin, and D. Zhu. Battlefield applications of hierarchical management with SHAMAN. In *Proceedings of the Second ARL/ATIRP Annual Conference*, pages 235–240, College Park, MD, February 1998.

[11] Mil 3, Inc., Washington, D.C. *OPNET Modeler, v6.0*, 1999.

[12] M. Humphrey and B. Rivera. Investigation of static and dynamic wireless network routing including terrain effects and application layer tuning. In *Proceedings of the Third ARL/ATIRP Annual Conference*, pages 241–245, College Park, MD, February 1999.

[13] A.S. Sethi, K. Sivakumar, V. Hnatyshin, M. Humphrey, and B. Rivera. Integrating SHAMAN management applications with battlefield OPNET simulations. In *Proceedings of the Fourth ARL/ATIRP Annual Conference*, pages 319–324, College Park, MD, March 2000.

[14] A.S. Sethi, D. Zhu, Y. Zong, D.-P. Hsing, and L. Kant. Application of the SHAMAN management system to battlefield network configuration management. In *Proceedings of the Fourth ARL/ATIRP Annual Conference*, pages 187–191, College Park, MD, March 2000.

[15] L. Kant, C.-H. Zhu, D.-P. Hsing, and M. Lee. An adaptive configuration management architecture design for the army's networks. In *Proceedings of the Fourth ARL/ATIRP Annual Conference*, pages 223–227, College Park, MD, March 2000.