INTEGRATED NETWORK MANAGEMENT OF HYBRID NETWORKS

John S. Baras, Mike Ball, Ramesh K. Karne,
Steve Kelley, Kap D. Jang, Catherine Plaisant,
Nick Roussopoulos, Kostas Stathatos,
Andrew Vakhutinsky, and Valluri Jaibharat
Center for Satellite and
Hybrid Communication Networks
Institute for Systems Research
University of Maryland
College Park, MD 20742
(301) 405-7901

David Whitefield
Hughes Network Systems
11717 Exploration Lane
Germantown, MD 20876
(301) 212-7909

Abstract

We describe our collaborative efforts towards the design and implementation of a next generation integrated network management system for hybrid networks (INMS/HN). We describe the overall software architecture of the system at its current stage of development. This network management system is specifically designed to address issues relevant for complex heterogeneous networks consisting of seamlessly interoperable terrestrial and satellite networks. Network management systems are a key element for interoperability in such networks. We describe the integration of configuration management and performance management. The next step in this integration is fault management. In particular we describe the object model, issues of the Graphical User Interface (GUI), browsing tools and performance data graphical widget displays, management information database (MIB) organization issues. Several components of the system are being commercialized by Hughes Network Systems.

INTRODUCTION

Hybrid communication networks provide an economically feasible and technologically efficient means to implement the global information infrastructure. The management of such heterogeneous networks is a critical market differentiator for telecommunications companies and a formidable technical task. In this paper we describe the collaborative effort between he University of Maryland and Hughes Network Systems, under the auspices of the Center for Satellite and Hybrid Communication Networks, to design and implement an integrated network management system for such networks. This paper is a continuation of the Baras (1995) paper and provides a description of our progress in the second year of this two-year joint research and development effort.

The major accomplishments during the second year were: the extension of the object oriented data model to hybrid networks consisting of satellite networks and terrestrial ATM networks; the extension of the system to hybrid networks with as many as 300,000 nodes; the improvement of the browsing graphical tools so that mesh-connected graph networks (as opposed to tree networks) can be efficiently queried; specially designed graphical widgets for displaying performance data continuously from the network management information database (MIB); extensions of the GUI to distributed operation including appropriate designs for consistency and concurrency between the GUI display and the network MIB; storage and organization issues for performance data in the MIB; integration of configuration management and performance management. In addition to the heterogeneity stemming from the interconnection of a terrestrial ATM network to a satellite LAN network, the system must handle vendor and protocol heterogeneity as well as operate in a distributed interactive (with the operators) environment.

SYSTEM ARCHITECTURE

The approach we are following in designing and implementing the INMS/HN is as follows. We first represent the network in a carefully designed Object Oriented data model in an OODB, following the principles of Haritsa (1993). We develop advanced GUIs linked to this OODB representation of the network including efficient browsing tools which exploit hierarchies in the data model. The OODB is linked to network simulation for comparisons and "what-if" decision assistance. We employ innovative dynamic query techniques which can be invoked from the GUI. We develop and implement performance objects in the OODB and link them to sophisticated graphical widgets in the GUI for performance monitoring and management. We allow multi-resolution (temporal and in dynamic range) performance data storage for economy of storage and speedy recovery of relevant information. We embed operational and management constraints in the OODB and we embed multi-criteria optimization tools and fast search algorithms

in the OODB for fast trade-off analysis and decision assistance. The resulting architecture of the software system is shown in Figure 1.

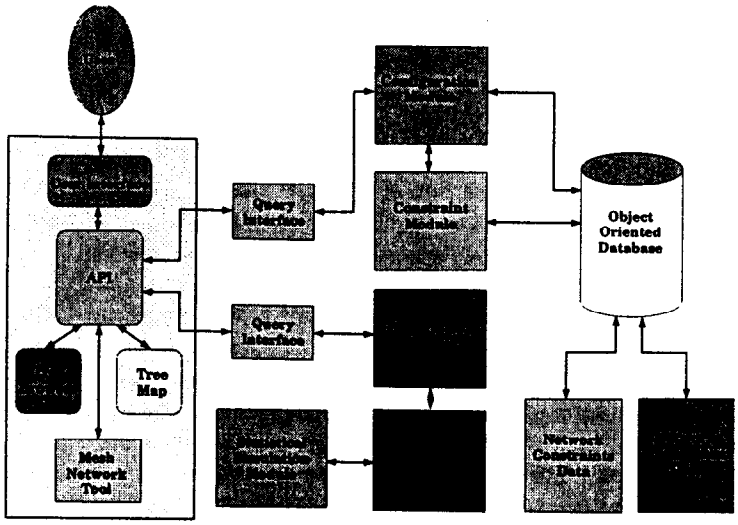

FIGURE 1. Architecture of the Integrated Network Management System.

We have currently completed a prototype of integrated configuration and performance management. Our next milestone is the efficient integration of fault management as well. The current implementation has been tested with simulated data of a 300,000 node hybrid network.

EXTENSIONS/IMPROVEMENTS OF THE MIB DESIGN

Object Oriented Data Model

We have extended the data model to include ATM networks. We also include frame relay X.25 over ATM in the terrestrial portion of the hybrid network. A hybrid network with an excess of 300,000 nodes and links was created and stored in Object Store. The object model hierarchy is depicted in Figure 2.

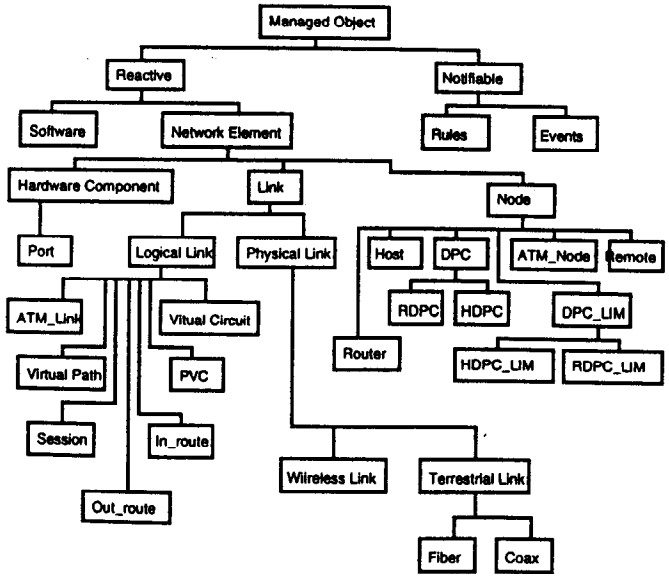

FIGURE 2. The Implemented Object Class Hierarchy for the Hybrid Network Data Model.

PERFORMANCE DATA MODEL AND STORAGE

We have developed and implemented efficient methods for storing and viewing performance information from a large hybrid network. A network simulation was designed and implemented in order to populate the MIB with performance data and related statistics. This simulation can set up Permanent Virtual Circuits (PVC) and vary traffic over them. The simulation periodically reports network traffic, error rate and cell loss rate which are stored in "Performance Objects" in Object Store. The system supports two types of user queries:

- Queries on a single object : Typical queries would be
  - Utilization of a particular Link at some specified time.
  - Buffer capacity at a given Node.
  - Delay and Error rate over a specific link.
- Queries across objects : These queries are more complex and involve attributes of more than one object. Typical queries would be of the form
  - The aggregate delay over a specific Virtual Circuit.

Such queries would require computation based on the attributes of different objects.

The operator may want to see the state of the Network at some earlier instant in order to analyze the nature of a fault that may have occurred on some part of the Network. Hence it is critical to store, the state of various Network elements at different instants, along with the instant at which it was recorded for a sufficient period of time. It would be neither practical nor necessary to store all the information gathered over a period of time to be stored at the same granularity. A reasonable solution to this would be to reduce the precision of information stored as the information gets older i.e for the most recent information we could store every update from the network, for slightly older information we could store an average over 4 periods, for even older information we could store an average over 20 periods etc. There are three different processes in our implementation, one for each level of granularity.

We have implemented a performance model that is suited for distributed implementation. Sensors periodically report snapshots of different network elements recorded at certain time instants. These snapshots include all the performance parameters being monitored for each network element. Since all the performance parameters for each time instant are being reported together it would be more efficient if we stored them together rather than storing them in different objects.

We have also investigated methods for finding the appropriate precision levels, as well as for efficient storage of time series data (such as those from the network sensors) and their related statistics. We shall report on these results elsewhere due to space limitations.

IMPROVEMENTS IN GUI DESIGN

Efficient Browser for Mesh Connected Graph Networks

In our previous work (Baras 1995) we designed and implemented efficient browsers and visualizations of the OODB representing the network, by exploiting the tree structure of the network. In the present extension to hybrid networks which include ATM terrestrial networks, we have to deal with fully mesh connected graph topologies, not just tree topologies. Therefore, there are no unique or obvious hierarchies to drive the browser, like the Tree Map and Tree Browser of Baras (1995). Instead we designed the Mesh Network Browser which explores the various partial orders the operator can create by using subnetworks of the network. This browser can then be used to invoke dynamic queries in the underlying OODB representation of the network, for selectively viewing desired parts of the hybrid network. The display/visualization of the network obtained using the Mesh Network Browser is shown in Figure 3.

A major problem we have addressed is generated from the fact that the area of a computer screen cannot visibly represent large networks including thousands of nodes. We designed our Mesh Network Browser so as to allow the operator to select the subnetworks and components of each subnetwork and even network nodes that he/she wants to query for network information. Our method provides an important innovation and it is a significant departure from current commercial practice, where this browsing is typically done using zooming of subnetworks or nodes. We allow for network component selection and display across a subnetwork or node hierarchy. Our browser allows selection capabilities based on a network element menu, based on subnetworks and also based on filtering of network elements using specific assignable attributes. This is implemented by using the option menu to allow the user to

select specific attributes of nodes and by using a Range widget to specify the range of the attribute. For instance, Figure 3, only displays workstations within a certain range of the load attribute. Our browser and display also categorizes network components into several groups. It allows the user to quickly navigate a network group for status and concentrate only on group components which are of interest. Components in the layout are represented as an icon, a colored dot, or are completely hidden by user definition or filtering.

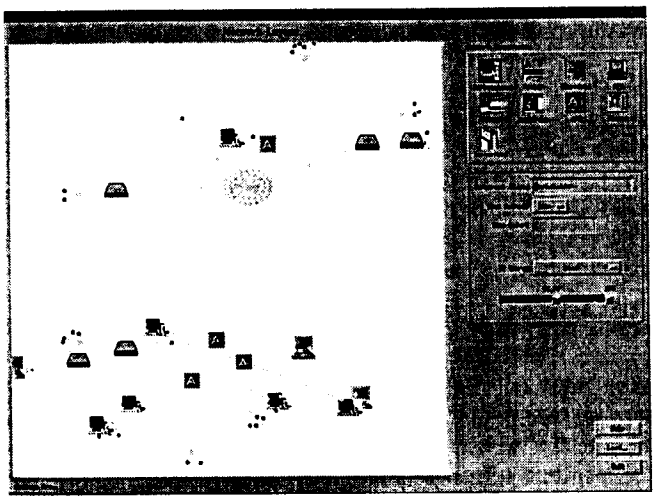

FIGURE 3. Illustrating the Mesh Network Browser Display.

Our browser is directly connected to the OODB representing the network and retrieves data (or objects) using dynamic queries. In this way, it represents an efficient way to capture and display network component status dynamically, for continuous network monitoring and management.

Wheel Widget Performance Data Display

One of the critical requirements for the INMS/HN is to design and implement efficient displays of continuous monitoring of performance data in an integrated fashion; i.e. simultaneous visualization of several attributes or performance metrics. We have designed and implemented one such graphical display/visualization: the "Wheel Widget."

The Wheel Widget displays four numeric values that fall within their upper and lower bounds. The widget allows the user to change those values interactively using a grab & drag mechanism. The widget is also useful and intuitive to use if it is operated with four Range widgets as controllers of its interface. Figure 4 shows how a Wheel Widget can be used with four Range widgets in an application. The widget allows the user to change its data, ranges, colors, size, etc. at run time. In Figure 4, each spoke in the widget represents network performance/monitoring data which has multiple attributes. It might be characterized by four attributes related to the corresponding data. This Wheel widget is used to display large numbers of point to point connections from a single source along with several attributes about each connection at the same time. Each spoke represents the following attributes:

1. Utilization which is represented by a distance from the center of the wheel to the center of a spoke.
2. Virtual Path Identifier which is represented by an angle from the vertical axis clockwise.
3. Capacity which is represented by the size of a spoke.
4. Throughput which is represented by the color of a spoke. The widget provides a spectrum of colors used to represent a range of colors.

The Wheel Widget provides direct data-filtering capabilities on attributes which are related to the ring (distance from the center to spokes) and rim (angle between vertical axis and spokes). The Wheel widget can continuously display network monitoring information and can therefore be incorporated in feedback schemes for automated network management.

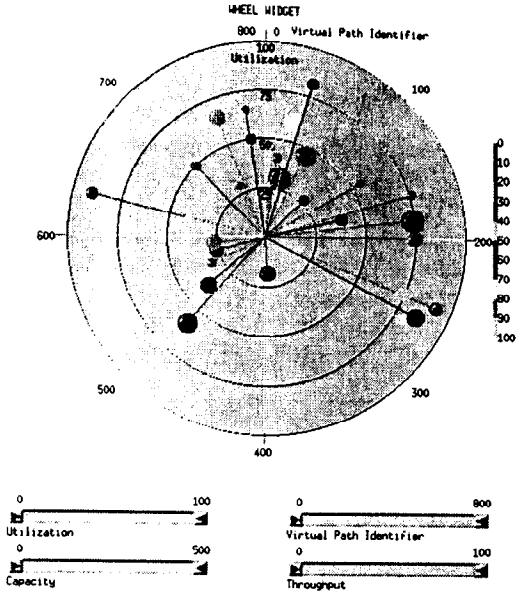

Figure 4: The Wheel Widget visualizing network monitoring data.

Distributed Concurrent Display Of Network Data

Network management systems must be able to operate in a distributed environment. This requirement creates several problems related to the consistency, concurrency and performance of the GUI and its link to the OODB under distributed conditions. We describe here a summary of our results and their implementations for network management. For the full details we refer to (Stathatos 1995).

One of the main concerns of application developers is that users are very sensitive to the response time of the system. Therefore, high variability in the response time of the user interface should be prevented. So, building a GUI that displays large amounts of information stored and managed by a DBMS can be very challenging. Client data caching appears as the best approach to deal with the performance problems of the user interface. Database objects cached in client's main memory can be directly used for user interface manipulations. This can reduce secondary storage accesses and client-server communication overhead. However, data caching as has been implemented in current systems does not completely address the user interface requirements.

A non-trivial problem for the graphical user interfaces of database applications is presenting a consistent and up-to-date view of the database. This problem is more evident and more difficult in a multi-user environment (as network management) where different users may view and possibly update the same database objects. From the database perspective, the *display consistency* problem is not much different than the client cache coherency problem. Displaying some database objects can be considered a kind of long transaction, a *display transaction*, which spans the lifetime of the display. However, traditional transaction semantics cannot be used to preserve GUI consistency since they are much too restrictive. We propose that, for each interactive application, a proper *external display schema* should be defined over the existing database schema. Such display schemes are composed of *display classes (DCs)* that encapsulate the desired user interface functionality and form inheritance and/or containment hierarchies that better meet GUI requirements (e.g. for screen layout computation, for screen navigation etc.) both in terms of implementation effort and runtime efficiency.

The graphical elements that compose the image displayed by a GUI must be instances of display classes, i.e. *display objects (DOs)*. Display objects are created by copying and/or computing the necessary information from database objects. During their lifetime, they are explicitly associated and kept consistent with those database objects. This association turns the collection of display objects into an active (updateable) view of the database as opposed to a passive snapshot. We also propose the introduction of *display cache* as an additional level in the memory hierarchy

on top of the client's database cache. Since, the display cache replicates data that may already exist in another part of the same physical memory space, it may appear as an unnecessary overhead. However, it has two major performance advantages over traditional caching (Stathatos 1995).

For the relaxed correctness requirements of display transactions we propose a non-restrictive form of shared locks, called *display locks*. These are non-restrictive in the sense that display locked database objects can be updated, provided that at any time all lock holders get notified about the updates committed to the database.

We have demonstrated these ideas in a multiple user, limited functionality version of a network configuration management application. This application employs two different visualization techniques, the *Tree-Map* and the *PDQ Tree-browser*, to display complex hardware hierarchies (Baras 1995). ObjectStore, a commercial object-oriented database system, was used to store the network database.

The overall system architecture is presented in Figure 5. For more details on the implementation for network management we refer to (Stathatos 1995).

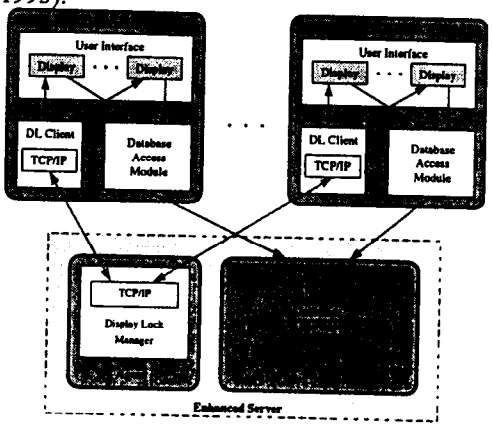

FIGURE 5. Implementation Architecture.

CONCLUSIONS

We presented the design and implementation of an integrated network management system which incorporates several advanced technologies predicated by current and future hybrid network management requirements. The next step in our efforts is the integration of Fault Management to the INMS/HN. The prototype implementation has given ample evidence of the advantages offered by our techniques and implementation methods. As a result several of these innovations are being commercialized to HNS commercial products.

Acknowledgments

This work was supported by the Center for Satellite and Hybrid Communication Networks, under NASA contract NAGW-2777, Hughes Network Systems and the State of Maryland under a cooperative Industry-University contract from the Maryland Industrial Partnerships Program (MIPS).

References

Baras, John S., et al. (1995) "Next Generation Network Management Technology," *Proc. of the Conference on NASA Centers for Commercial Development of Space*, M. S. El-Genk, ed., AIP Conf. Proceedings No. 325Albuquerque, NM, 75-82.

Haritsa, J.R., et al. (1993) "MANDATE: Managing Networks Using Database Technology," *IEEE Journal on Selected Areas in Communications*, 1360-1372.

Stathatos, K., S. Kelley, N. Roussopoulos and J.S. Baras (1995) "Consistency and Performance of Concurrent Interactive Database Applications," *Institute for Systems Research Technical Report TR 95-79.*