INTEGRATED MANAGEMENT OF LARGE SATELLITE-TERRESTRIAL NETWORKS*

J. S. Baras, M. Ball, N. Roussopoulos, K. Jang, K. Stathatos, J. Valluri

Center for Satellite and Hybrid Communication Networks
Institute for Systems Research
University of Maryland
College Park, Maryland 20742

ABSTRACT

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 our efforts in the Center for Satellite and Hybrid Communication Networks, to design and implement an integrated network management system for such networks. The major accomplishments to date are the design and implementations of an object oriented data model for 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 development of 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.

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. Hybrid networks are also the architecture of choice for military networks. In this paper we describe a prototype system under development in the Center for Satellite and Hybrid Communication Networks, for the integrated network management of such networks. The major accomplishments todate are: 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.

A typical network for which the system developed is intended consists of an ATM fiber network connected to a high data rate satellite constellation. In addition to the heterogeneity stemming from the interconnection of a terrestrial ATM network to a satellite 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

* This work was supported in part by the Center for Satellite and Hybrid Communication Networks under NASA cooperative agreement NCC3-528 and by grants from Hughes Network Systems and Space Systems Loral.

0-7803-4249-6/97/$10.00 © 1997 IEEE

of [2]. 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.

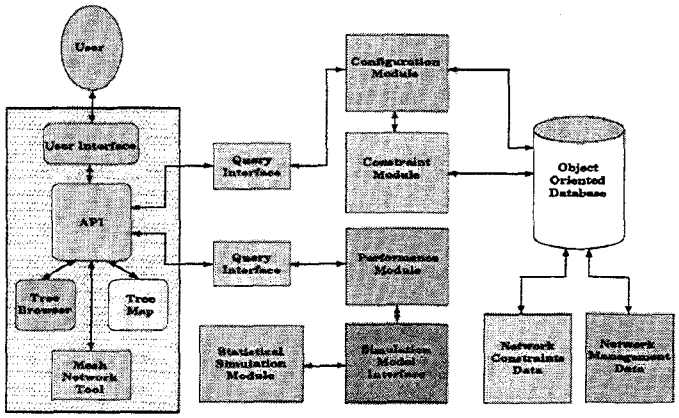

Figure 1. Architecture of the Integrated Network Management System

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 and 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.

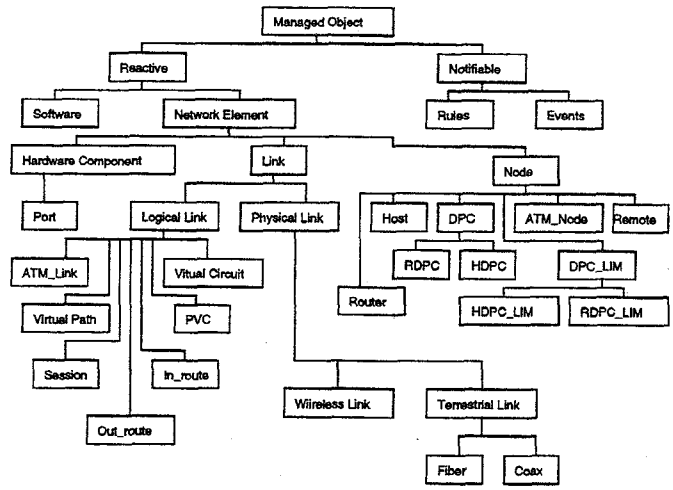

Figure 2. The implemented object class hierarchy for the hybrid network data model

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.

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 in 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 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. The high precision process periodically takes values stored in memory and updates the database. The processes at the other two levels periodically take a block of values stored at the previous level of precision, compress them and store them at the next lower level.

We have implemented a performance model that is suited for distributed implementation shown in Figure 3. 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. Another point in favor of this kind of storage is that in cases where the performance of a particular network element is degrading, we might want to see all the parameters for that network element recorded at some time instant. In the case of such queries it would be inefficient to search for these values across several objects.

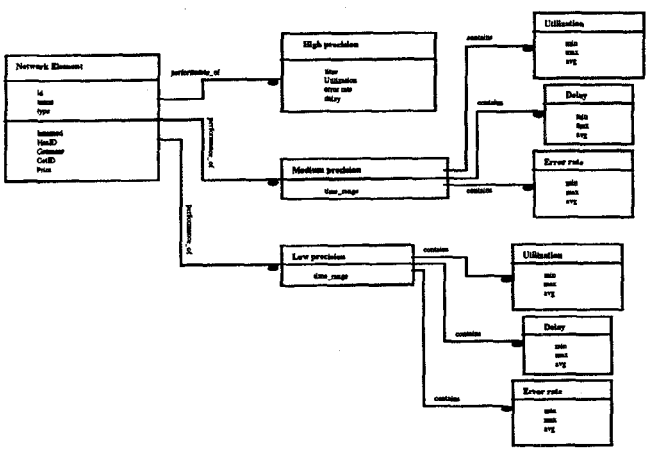

Figure 3. Integrated performance data model

IMPROVEMENTS IN GUI DESIGN

Efficient Browser for Mesh Connected Graph Networks

In our previous work [1] 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 [1]. 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 4.

We have adopted the Sgraph structure to display large hybrid network configuration stored in an OODB. The configuration data in the database consists of network nodes, links, and connectivity information between nodes. However it does not include x,y coordinates to efficiently display a three dimensional network onto the two dimensional computer screen. The Mesh Network Browse constructs an Sgraph structure corresponding to the network objects displayed. Each network object has an assigned icon, of scaleable size. Then by the layout algorithm employed, it traverses the Sgraph structure to assign x,y coordinates for each network node (object) without allowing nodes to overlap and eliminating unnecessary edge-crossings at the same time. Based on the Sgraph data structure, several layout algorithms are currently available; we have used the Springer Embedder algorithm which emphasizes display of symmetry and isomorphic components.

Nevertheless, 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 4, 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.

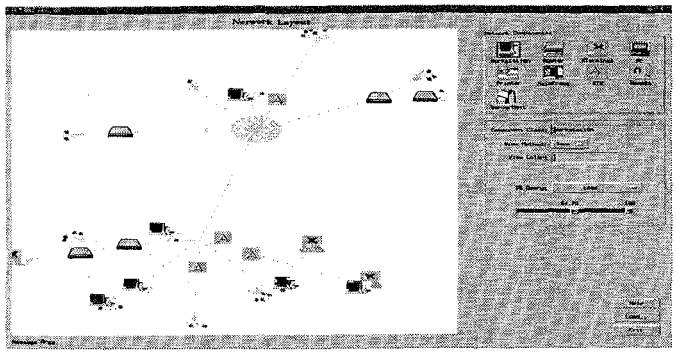

Figure 4. Illustrating the Mesh Network Browser display

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.

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 [3].

One of the main concerns of application developers is that users are very sensitive to the response time of the system. Lengthy response times are usually detrimental to productivity, increasing user error rates and decreasing satisfaction. 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. Many potential performance problems exist since the response to a user action may require extensive data processing, a number of network of message exchanges as well as several data retrievals from secondary storage.

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. Obviously, some sort of display synchronization mechanism is required which preserves the consistency of the user interfaces, under the performance requirements mentioned above. Generally, the straightforward approach of periodically refreshing the user interfaces is not considered acceptable since it may cause excessive overhead.

We propose that, for each interactive application, a proper *external display schema* should be defined over the existing database schema. Such display schema 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 definition of a DC depends on the database class(es) it represents as well as the user interface context. It should include only attributes and methods that are necessary for the display and manipulation of the corresponding user interface elements. These attributes may be a subset of the database class(es) attributes as well as additional GUI specific attributes (e.g. screen coordinates).

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 (updatable) 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.

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.

The display locking protocol is quite simple and can be easily integrated with a strict avoidance-based protocol. A client requests display locks for all database objects that are associated with display objects. The database lock manager on the server is expected to grant those locks, since display locks are compatible with all types of locks. When a transaction wants to update some data, it does so after obtaining an exclusive lock for that data. When the update is committed to the database, the lock manager releases the exclusive locks and notifies all clients that hold display locks on the updated data. The notified clients refresh the associated display objects (and therefore the display) by reading the new data from the database. We call this protocol *post-commit notify protocol.*

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 [1]. ObjectStore, a commercial object-oriented database system, was used to store the network database.

The implementation included three major tasks:

1. Extend the database server with display locking capabilities,
2. Enhance the client applications structural design to incorporate the display locking mechanism, and
3. Design the user interface in terms of
   defining appropriate display classes for the tree-map and PDQ tree-browserq

The overall system architecture is presented in Figure 5. For more details on the implementation for network management we refer to [3].

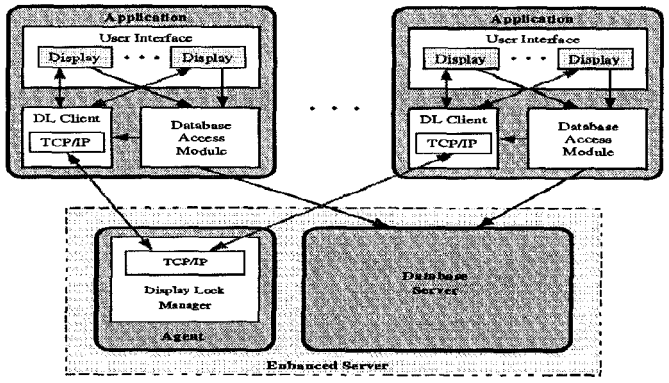

Figure 5. Implementation Architecture

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.

REFERENCES

[1] Baras, John S., et al., January 1995, "Next Generation Network Management Technology", in AIP Conference Proceedings 325, Conference on NASA Centers for Commercial Development of Space, Albuquerque, NM, pp. 75-82.

[2] Haritsa, J.R., et al, December 1993, "MANDATE: Managing Networks Using Database Technology", IEEE Journal on Selected Areas in Communications, pp. 1360-1372.

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