Updating Software And Configuration Data In A Distributed Communications Network

Carl W. Symborski

Hughes Network Systems, Inc.
11717 Exploration Lane
Germantown, MD. 20874

Abstract

Distributed communications networks often consist of numerous independent or loosely coupled network components. These components are typically sophisticated computing devices that collectively support the performance of network functions. Frequently, an individualized set of software and configuration data may be required for each component. Control of this data should include mechanisms for update and distribution to all network components. These mechanisms should provide for the initial load of information to network components as well as support the application of updates without network disruption. This paper presents a method for the organization and controlled distribution of software and configuration data throughout a communications network. This methodology is illustrated by its implementation in an advanced, commercially available packet switching network.

Introduction

One of the numerous tasks that fall within the realm of network management involves the management and control of internal network data. As opposed to data carried by the network on behalf of its users, internal network data includes data associated with all network components, external devices, and users attached to the network. Software, for example, is required by computer based network components in order to support their function. Specific software releases may be required by these components to support a specific mix of services provided to network subscribers. Feature packages may be offered to subscribers in regionalized areas of a network. This can potentially prompt regionalized distribution of software releases. In order for network components to function effectively as a part of a network, additional information such as topology and routing data must also be supplied. Additionally, devices and users attached to the network often have certain adjustable parameters associated with them. These parameters can be collected into distinct profiles and need to be supplied to applicable network components so that service may be provided.

The tasks of network management in these areas can be summarized as [1]:

1. Accumulating and managing data required by network components in order to support useful operation.

2. Facilitating the distribution of topological configuration, routing information, user profiles and other necessary information by translating and binding this data into formatted tables.

3. Providing a mechanism for the distribution and update of software and required data tables to network components in a network.

The main thrust of this paper is an investigation of the last task, specifically, the controlled distribution of network software and data tables throughout a network. Central to this discussion is the definition of two terms: download, and data reconciliation. Download can be thought of as a mechanism by which network software and data are obtained and installed in a network component. Data reconciliation can be thought of as the process by which the consistency of all downloaded information within a network component is assured. A network component can use data reconciliation to determine what data is needed for operation and when that data has changed, thus requiring update. The download process can then be used to obtain the needed data.

Design Considerations

Some goals have been identified concerning the implementation of download and data reconciliation functions within a network. These are:

1. The ability to update software and data tables associated with a network component with minimal service impact.

2. The maintenance of consistent network data within and between network components.

One method of downloading a network component is to place the component out of service while it is being loaded. This method can cause several problems. Removing a component from service may terminate ongoing data transfers processed by that component. Service may be suspended for minutes depending on the amount of software and data tables that need to be downloaded. Additionally, some updates may only involve a modification to tables that define parameters of users and equipment attached to a component. To cause the reload of all software and tables to accomplish this change has a high cost in terms of service availability. It would be desirable to support the update of data tables independent from software. Ideally the network would support the download of individual tables that can be intercepted by a network component while it is still in service [1]. From a customer's point of view, minimizing the service impact of a component update can be seen as an advantage. If a network component supports the update of tables or fragments thereof, only users associated with the updates are affected. This reduces the down time from the viewpoint of the customer.

Software and data tables, once downloaded into a component, must be maintained so as to be consistent with the component's definition within the network. This is necessary in order for the component to support its role in the overall network configuration, if modified by the network managers. A component must determine, or be informed, that its definition has changed and that updated information should be downloaded. To reduce the problem of coordinated update of data between components, cross-component redundant data should be minimized. Information should be held only where and when it makes sense to do so. If duplicate data exists across component boundaries, one copy should be designated as the master [2]

In addition to the above goals, the rate of change that network data may experience over time can influence the type of distribution mechanism used. In networks where change is rare, network data may be created as an offline operation, possibly at a centralized network control facility. This data, along with software, can then be delivered to the different network component locations and installed on local storage devices. In this scenario, a component can download software and data tables using its local storage requiring little or no network dependence [1].

In more volatile networks, network data is changed often. Users and devices are typically added, changed, or deleted on a regular basis. To support the timely distribution of this dynamic information, the network can be used to download software and data tables from a central location to the network components [1]. This is the method utilized in the Integrated Packet Switching Network product line developed by Hughes Network Systems, Inc. In this implementation, a centralized network management system is used to create or modify the definition of network components and their related users and devices. These changes are then provided to affected components using a hierarchical distribution mechanism over the network.

The following sections focus on techniques that utilize the network for the controlled distribution of software and data. A download and data reconciliation methodology is presented, which can maintain the consistency of all network components with their current definitions as well as accommodate online component update with minimal service impact.

Organization Of Download Data

When designing a system which supports the download of online components, many problems can arise. For example, a component will need a procedure to determine if it needs data or that some data has changed. The specific data items required by each component must also be determined. To some extent, the organization of the downloaded data is the first step towards simplifying these and numerous other problems. As previously discussed in relation to volatile networks, data that is downloaded to network components is often stored at a central location. At this location, typically a network control center, the data is initially entered and maintained using available network management functions. It is suggested here that the data that is distributed to network components be divided into a number of blocks, called descriptors. Data should be partitioned into descriptors for a number of reasons:

1. A given data update applied at the network control center may only affect a part of the total amount of data downloaded to a component. By dividing this data into blocks of related information, only modified data have to be obtained.

2. Data can be grouped according to how they are generated, allowing easier management.

3. Data can be grouped according to how they are used, allowing easier use by components.

4. Data which are used by many components can be stored as a single descriptor, rather than as one per component.

Categories Of Data

Using descriptors, the download of network data to a component can be viewed as the transfer of a set of descriptors to the component. The types of descriptors that need be created depend on the composition of the original data. Generally, data can be organized into three categories for distribution and reconciliation purposes. These categories are:

1. Component specific data

2. Shared data

3. Summary data.

Component specific data includes data specific to individual network components. Data identifying protocol parameters and hardware configuration of ports associated with a component are examples of this category of data. Each network component will have its own unique set of component specific data for its exclusive use. For download purposes, component specific data can be grouped into a single descriptor. Each component would therefore have a descriptor created specifically for it. A component could download its specific descriptor and thus obtain all of its component specific data.

The second category, shared data, includes all data that might be shared by multiple components. Any data needed by more than one component falls into this category. Global network parameters are an example. Software images used by components and software patches to these images also fall in the shared data category. Shared data should be grouped into multiple descriptors according to usage.

Summary data is a category of data that exists only to support the implementation of a data descriptor organization. For identification and data reconciliation purposes, each descriptor should contain a unique timestamp. The timestamp indicates when a descriptor was created or last updated. Summary data for a component consists of all the timestamps of the descriptors associated with the component. These timestamps can be gathered into a single summary descriptor and downloaded to the component. Using the timestamps in its summary descriptor, a component can apply data reconciliation procedures to identify the latest versions of specific and shared descriptors if required.

The usefulness of a descriptor organization will become more apparent as the overall concept of download service is presented.

Download Service

Given that downloaded data can be partitioned into descriptors, a download service mechanism must then be provided to support the distribution of these descriptors throughout the network. Specifically, a download service should:

1. Provide a general mechanism for the distribution of descriptors to network components.

2. Ensure the timely distribution of descriptor changes to online components with minimal service disruption.

3. Ensure that network components hold descriptors that are up-to-date and consistent with those held by all other components in the network.

It can be seen that this download service creates an implicit service hierarchy in the network. At one end of the hierarchy are those components that distribute descriptors. As discussed previously, descriptors should be generated and maintained at a single location. This centralized data management point is at the top of the service hierarchy. A master set of all descriptors is stored at this top level. All descriptors that are distributed to one or more components originate from this master set. Since descriptors are created and emanate from a single point, network-wide descriptor consistency can be achieved. At the other end of the hierarchy are those components that need and use descriptors. These components can maintain local consistency of the descriptors they hold by validating their descriptors with those in the master set. Through this process, called data reconciliation, a component can determine which descriptors are needed and request their transfer from the master set. The download service hierarchy therefore provides for centralized descriptor management at the top level as well as distributed download control driven bottom up from the lowest level.

Network components can be identified within this hierarchy in terms of the roles that they assume: consumer or master server. A consumer's functions include:

1. Periodic reconciliation of the descriptors it holds with the master set.

2. Generation of requests for descriptors from the server as needed.

3. Utilization of (consuming) the data in the descriptors.

4. Initiation of a transfer of all required descriptors when a full download is required.

The master server's functions include:

1. Maintenance of a master set of descriptors.

2. Providing descriptors to consumers upon request.

3. Expediting the distribution of descriptor updates by informing all components of online updates, thus speeding up a component's periodic reconciliation cycle.

Exploiting its role in the service hierarchy, a master server can better support network management functions associated with the accumulation and management of the data contained in the descriptors. This may include operator interfaces, network configuration databases, report generation, and other network management features as required. Freed of these tasks, consumers are specialized in actual use of the data in the descriptors. Being data users, consumers have implicit knowledge about the different descriptors they need to operate. By making requests to the master server, consumers can detect online configuration changes for the specific descriptors that they depend on. Initiating the transfer of changed descriptors in this bottom-up fashion ensures the distribution of changes only to affected consumers, minimizing service disruption and overhead on network resources.

Intermediary Servers

Large networks will often consist of numerous consumer components. This number may generate a demand for service that is greater than the capacity which the master server can handle. In these networks, intermediaries can be inserted in the hierarchy between the master server and consumers. Intermediaries can reduce the load on the master server by providing some server functions to consumers. Specifically, the functions of an intermediary include:

1. Maintaining a local descriptor cache that contains copies of descriptors from the master set.

2. Accepting descriptor requests from consumers, attempting to satisfy these requests using its cache.

3. Obtaining requested descriptors from the master server as required, if not present in the cache.

Intermediaries therefore behave both as a server and a consumer. Intermediaries act like a server to consumers since they can satisfy descriptor requests. However, intermediaries are really consumers themselves since they are not involved in the creation of descriptors but rather obtain them as consumers from the master server.

As a member of the service hierarchy, a major utility of the intermediary is to provide a level of server redundancy by increasing server availability to consumers. The complete service hierarchy is illustrated in Figure A.

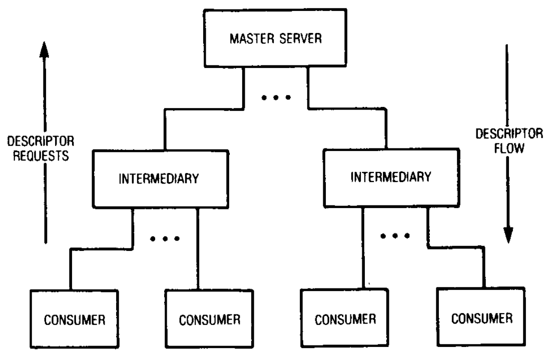

Figure A. Download Service Hierarchy

Sample Consumer Processing

Given the definition of descriptors, servers, intermediaries, and consumers, a sample data reconciliation procedure can be outlined. This procedure, executed by all consumer components consists of the following steps.

1. Wait until a reconciliation timer expires, or receipt of notice that some descriptor may have changed.

2. Request current summary descriptor applicable to this component from a server (either master server or intermediary).

3. Accept summary descriptor from server and compare with local summary descriptor describing the components current descriptor profile.

4. If no change is detected, no work need be done. Return to step 1.

5. If change is detected, then request each changed descriptor from the server.

6. Install each new descriptor in the component, replacing the old descriptor.

7. Save the new summary descriptor for the next reconciliation cycle.

8. Return to step 1.

Using this procedure a consumer component will maintain a local set of descriptors that are consistent with the master descriptor set. This procedure provides the consumer with the ability to determine on its own which descriptors it is holding are out-of-date. The consumer can then initiate the download of new descriptors. Freed of this burden, the master server can be dedicated to the maintenance of a consistent master set of descriptors.

The Integrated Packet Network

The download and data reconciliation methodology presented in this paper has been implemented in the Integrated Packet Switching Network (IPN) product line developed by Hughes Network Systems, Inc. Before illustrating this implementation, it will be appropriate to give a brief synopsis of the architectural aspects of the IPN [3].

Within the IPN, network components are divided into two generic categories: packet switching node components and network control system components. Packet switching node components provide the packet protocol and switching function. The network control system components provide both centralized and distributed management and control functions for the entire network.

The network's packet switching node components are implemented using one or more CP9000 Series II line clusters. The CP9000 Series II line cluster allows for flexible configurations based on modular multiprocessor design. When configured for the purposes of switching data, the CP9000 Series II line cluster is referred to as a Packet Switching Cluster (PSC). All network nodes consist of one or more PSCs which provide the functions of routing and physical, link, and packet level management. PSCs are interconnected to form the network backbone topology. A backbone protocol is used for the transfer of data between PSCs.

The network is monitored and controlled by the Network Control System (NCS). The NCS performs administrative functions and provides the network with various services. The NCS components are the Network Control Processor (NCP), the Network Operator Console (NOC), and the Auxiliary Service Processor (ASP).

The NCP is capable of providing all network administration functions and all network service functions of the NCS. The NCP also performs all functions that may be required by the network operators. As the prime handler of administrative information, the NCP is the highest authority within the network from an administrative point of view.

Network operators control and administer the network using a Network Operator Console (NOC). NOCs provide access to the network administration facilities performed by the NCP.

Auxiliary Service Processors (ASP) are used to provide a subset of NCP network services to PSCs. Performance of these network services by ASPs offloads the NCP from providing these services in large networks. ASPs are implemented using CP9000 Series II equipment configured with a disk for local storage of PSC software and configuration data.

All of the network components are required to communicate with each other to support network services and administration. In order to facilitate this communication, an interconnection between NCP, ASPs, NOCs, and PSCs is provided. This interconnection is called the Supervisory Network. The Supervisory Network is implemented using the network itself and is used to perform all network services and some network administration functions. Supervisory circuits are established between the components of the supervisory network. These circuits are set up using the same facilities provided by the network to its users. Although used for more than just component download, the supervisory network relates well to the download service hierarchy previously described. A typical structure of the supervisory network is illustrated in Figure B.

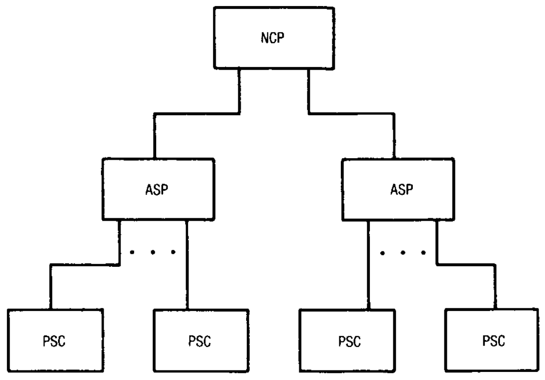

Figure B. Supervisory Network Structure

In this figure, the NCP corresponds to the master server in the download service hierarchy. PSCs are the consumers. ASPs are consumers themselves, but also act as intermediaries by providing download services to PSCs.

The downloaded network data, generated at the NCP, has been divided into numerous download descriptors. These descriptors, loaded into PSC and ASP clusters, are organized in the following manner:

1. Component Specific Descriptors:

    - A cluster configuration data descriptor (CCD) contains all configuration data that applies to an entire cluster.

    - A module configuration descriptor (MCD) contains data that applies to a single CP9000 Series II module of a cluster.

2. Shared Data Descriptors:

    - Universal shared system parameters (USSP) is a descriptor that contains configuration data that applies to all clusters.

    - Shared software descriptors contain the software images to be used by clusters.

    - Software patch descriptors contain patches that apply to corresponding software descriptors.

3. Summary Descriptors:

    - A cluster specific descriptor (CSD) contains the timestamps of all configuration data particular to a cluster, and the names of all shared data used by a cluster.

    - A first reconciliation descriptor (FRD) contains the timestamps of all shared data used by a cluster, and the timestamp of the CSD for a cluster.

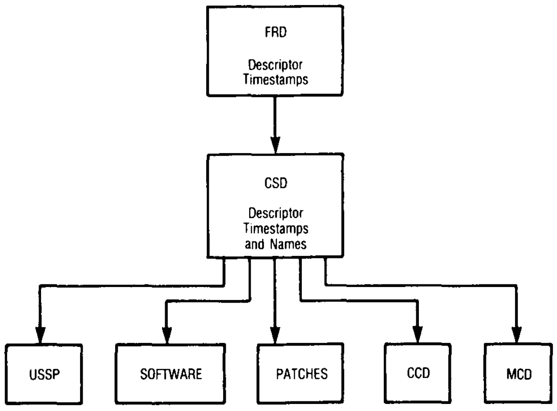

Figure C. Download Descriptor Organization

Summary descriptors contain information identifying the latest shared data and component specific descriptors for a cluster. The master descriptor set, stored at the NCP, includes unique FRD and CSD summary descriptors for each cluster in the network. The division of the summary descriptors into the FRD and CSD allows a cluster to verify using a single packet transfer, that all data it holds is up-to-date. Figure C illustrates the relationship of the summary descriptors with all other descriptors.

Descriptors are requested from the servers and transferred to the clusters using an internal download protocol. This protocol operates over the supervisory network between download and server applications executing on the clusters and servers. A descriptor is identified in a download request depending on descriptor type. Component specific descriptors are requested by supplying a cluster identifier uniquely identifying a cluster, and a timestamp from the CSD. Shared data descriptors are identified in a request by supplying the name of the descriptor from the CSD and a timestamp from the FRD. Summary descriptors are identified by supplying a cluster identifier and a timestamp. An exception is the First Reconciliation Descriptor (FRD) which has no timestamp. Being the first descriptor always requested, an FRD can be obtained using cluster identifier only.

Full Download Procedure

Upon startup, clusters utilize the download service to obtain the descriptors they require to become operational. Each cluster is provided with a minimum amount of identification information in order to obtain a download. This information, provided by hardware and nonvolatile memory, includes the name and identification codes of the cluster as well as the addresses of two devices through which it might contact a server. Using this information a call is placed to a download server. Once access to a server is established, a cluster performs the following steps to obtain its required descriptors:

1. The cluster obtains its First Reconciliation Descriptor (FRD) from the server.

2. The Cluster Specific Descriptor (CSD) is then obtained, requested from the server using the timestamp identified in the FRD.

3. All shared descriptors are obtained. The specific shared descriptors to request are named in the CSD and requested using these names and the timestamps identified in the FRD.

4. The cluster obtains component specific descriptors identified by the CSD,

requested using the timestamps identified in the CSD.

5. If all required data is obtained successfully, the cluster begins operating with the received data.

The above steps simply follow the descriptor relationships illustrated in Figure C. A cluster uses its summary descriptor information to request all required descriptors.

Online Reconciliation Procedure

Download descriptors which have been distributed to system components may be modified at the NCP using configuration management functions. Network components perform data reconciliation to detect such changes, and obtain up-to-date information. This reconciliation processing may be triggered periodically, or upon receipt of change notices. The NCP generates a change notice whenever download descriptor information is modified. Change notices are propagated through ASPs and are received by ASPs and PSCs. Clusters initiate reconciliation of descriptors they use whenever they receive any change notice. Additionally, clusters initiate reconciliation periodically to determine if any change notices have been missed. The following summarizes the procedure by which a cluster reconciles its download descriptors:

1. The cluster obtains its First Reconciliation Descriptor (FRD) from the server.

2. A comparison is made between this new FRD and the cluster's local copy of the FRD.

3. If no difference is detected, the cluster is considered reconciled and this procedure terminates.

4. If a difference is detected, then some of the descriptors currently held by the cluster are out-of-date. In this case, the cluster is considered unreconciled and the procedure continues.

5. The FRD identifies the latest version of the Cluster Specific Descriptor (CSD). If the cluster does not hold the latest version, it is obtained from the server.

6. The cluster evaluates the shared descriptor timestamps in the FRD, comparing each with the timestamps in the corresponding descriptors it currently holds. Where necessary, up-to-date shared descriptors are obtained from the server.

7. Lastly the cluster evaluates the component specific data timestamps in the CSD, comparing each with the timestamp in the corresponding data it currently holds. Where necessary, up-to-date component specific descriptors are obtained from the server.

In the above steps, a cluster obtains only those descriptors that have changed since the last time the cluster reconciled.

Descriptor reconciliation can take place at any time while a cluster is operational. This process can be performed without causing the interruption of any other cluster function. The only possibility for disturbance occurs when an updated descriptor is installed in a cluster. Most descriptor updates can happen with no service interruption. Any disruptions that do occur are localized as much as possible to minimize any global cluster effects. For example, a descriptor change representing an update to the parameters associated with a specific port will be installed without affecting any other port. Utilizing reconciliation in this way, online updates can be accommodated with minimal service disruption.

Conclusions

This paper has defined the roles and goals of download and data reconciliation functions in a distributed communications network. A basic methodology that can be used to download and control a large communications network has been presented. The concepts of descriptor organization, download service hierarchy, and data reconciliation are described as the basis of a distributed download philosophy. These concepts have been used in the implementation of Hughes Network Systems Integrated Packet Network product line. Field proven, this methodology has shown to be workable and effective. The capability of updating the online components of the IPN without disrupting or degrading performance of the system is one of the features that ensures reliability and maintainability of the system.

Acknowledgments

The IPN download descriptor and data reconciliation philosophy was developed through the cooperation and technical contributions of many members of the technical staff at Hughes Network Systems, Inc. The efforts of all involved are hereby acknowledged.

References

[1] Howard A. Seid, "The Ins and Outs of Managing a Packet Network," Data Communications, October 1983, pp. 149-161.

[2] M. J. Golaszewski, P. E. Janssen, and A. S. Modi, "Decoupled Software Recovery Concepts in a Distributed Processing Environment," IEEE Global Telemcommunications Conference (GLOBECOM '83), San Diego, California, November, 1983, Conference Record vol. 2, pp. 622-625.

[3] Thomas H. Scholl, "Electronic Communication Handbook on Packet Switching," McGraw Hill Electronics Communications Handbook edited by Andrew F. Inglis, in publication 1987.