RDStore: In-Network Resource Datastore with Distributed Processing of Resource Graph

Masanori Miyazawa, Michiaki Hayashi
KDDI R&D Laboratories, Inc.
2-1-15 Ohara, Fujimino City, Saitama Prefecture, JAPAN
ma-miyazawa at kddilabs.jp

Abstract— **The delay in comprehending network resources potentially harms precise network management activities, such as root cause analysis and traffic visualization. The delay tends to be remarkable with the growth of the network. This growth basically arises from the number of routers, and the number is easily multiplied by virtualization technology. Hence, the increase in resource data is accelerating, and in future, we will see a fatal limitation of the current management approach, that is, centralized management. To tackle this issue, a decentralized resource management architecture is investigated, and a distributed datastore for in-network resource information sharing, called in-network Resource information DataStore (RDStore), is proposed. In the framework, a simple network modeling based on a resource graph, called the Graph-based network Resource Information (GRI) model, is also proposed for effectively accessing resource data as well as for reducing resource data itself. The demonstration result shows the capability of the proposed accelerated comprehension of network resources, compared with the conventional centralized management approach.**

Keywords; Decentralized management, graph-based distributed database, in-network management.

I. INTRODUCTION

Today, network resource information dispersed throughout the network is consolidated using a centralized management system, such as a Network Management System (NMS). Centralized management has been a common approach because it is simple and convenient to implement management functions in the NMS. In contrast, the amount of resource data continues to increase and is about to increase still further due to network virtualization technologies, such as logical routers. The growth tends to delay comprehension of the network structure (e.g., topology), and further, requires the processing power of the NMS to maintain pace with the growth. In the near future, centralized management will face difficulties with comprehending the network condition in real time. The delay in understanding the network structure may harm network management activities, such as fault management (e.g., root cause analysis) and performance management (e.g., traffic management) since existing analysis requires the current network structure [1], and failure identification is thus delayed, resulting in degradation of service quality. To offload this process of the centralized manager, a distributed management approach has been discussed [6]. Introducing the distributed approach is expected to enable scalable management such as faster fault localization and performance management.

Typical decentralized management approaches have been studied, and have implemented a part of the management functionality within the network on behalf of the NMS [2-4, 6] The authors of [6] proposed a concept of In-Network Management (INM) architecture to facilitate the embedding of management functionalities inside the network. In [2, 3], the authors also defined a decentralized management model, which is similar to the approach of [6]. The main focus point was a peer-to-peer management model that can achieve scalability through the use of a distributed management protocol for monitoring resource discovery and distributed polling on the management plane. In [4], the authors focused on the network searching mechanism for network information including network resource and traffic data in a decentralized environment without the above management protocols, proposing uniform access to network data stored in local DataBase (DB) in network devices. This approach is important for easily gathering network resource data without implementing complicated protocols and interfaces, and is basically similar to our approach. However, the collected resource data needs to be aggregated on a centralized station for comprehending network topology. This means that processes with a high processing load, which is creating network topology for example, still remain in a centralized system. Therefore, there is a need to examine the structure of the in-network distributed DB itself for efficient offloading of centralized processing. To achieve an efficient in-network DB, not only offloading the processing load but also reducing the data volume may need to be considered in order to deal with large-scale network resources.

In this paper, we propose a decentralized network resource information sharing framework driven by simple network modeling based on a resource graph. The proposed graph-based simple network modeling enables not only instant access to the resource data but also a reduction in the resource data itself. The proposed framework is implemented in network devices for showing proof of concept, and reduced delay in comprehending network resources is demonstrated by comparison with the conventional centralized management system.

II. PROPOSED MANAGEMENT ARCHITECTURE

A. Architecture of RDStore

Figure 1 shows the proposed decentralized datastore for in-network resource information sharing, called in-network Resource information DataStore (RDStore). In order to offload

978-3-901882-50-0 ©2013 IFIP

the processing load using the decentralized management approach, RDStore has several functionalities executed inside network devices to manage the network topology. RDStore is basically composed of four function blocks: one is a distributed DB module designed to construct a single DB system between several network devices and store the integrated network topology information in a distributed manner; the second one is an adapter module that can collect network resource data from each network device and filter unnecessary resource data; the third one is a data processing module that can process the gathered resource data and send requests to a distributed DB for their storage; the final one is a network topology search API that enables control of access to the DB module and provides other systems with integrated network topology information. Each network device basically has one adapter and one data processing module, and the DB system is implemented in network devices; thus, it provides a single management platform to manage integrated network information in such distributed network devices. These functionalities in the RDStore are assumed to be implemented in network devices. For example, commercial routers such as the Cisco Services Ready Engine (SRE) have the functionality of a common server blade. Therefore, such network devices are available for implementing our proposed RDStore and supporting the acceleration of our proposed decentralized approach as shown in Fig. 1.

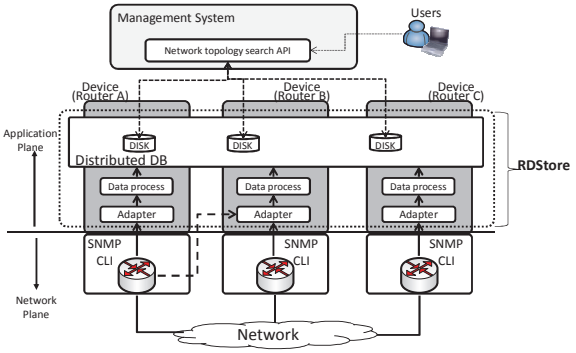

Fig. 1. Fundamental structure of the RDStore

B. Distributed DB system

Distributed DB systems are widely used today due to an increase in the amount of data. Such DBs allow applications to access data installed in multiple servers located in different locations. This DB has several advantages; for instance, data extraction is definitely more effective and more efficient, compared with a RDBMS DB because it is easier to accommodate the increase in network size by adding processing and storage power. In addition, RDBMS DBs are not very convenient for hierarchical or graph-like data modeling and processing. In contrast, a graph-based DB is utilized as a distributed DB because it can satisfy the above powerful functionalities. The graph DB, for instance, not only manages the most generic of data structures, but also represents any kind of data in a highly accessible way. Hence, the graph-based distributed DB is suitable for managing network topology modeling.

Thus, a graph-based distributed DB system is implemented to construct a single DB system between several network devices. By using this DB, the DB module can be designed to integrate the distributed resource data and retrieve the data in a uniform format from the decentralized environment. In the current implementation, the network topology search API is responsible for management of saving the location of the resource data in the network device and for retrieval of the data from the distributed DB. The data saving location can be decided from several network devices based on the free disk space available. However, it is necessary to implement the functionality in centralized manner because the location of the network devices that store the resource data must be precisely understood. Thus, the function needs to be implemented in one network device or an external system such as NMS.

C. Network topology information model in distributed DB

Figure 2 depicts an inheritance tree of the proposed graph-based network information model that can represent the integrated network topology, which is called the Graph-based network Resource Information (GRI) model. The proposed GRI model is a fairly abstract model, compared with the structure of MIB. Note that the GRI model focuses on the MPLS core network in this paper. The GRI model consists of six Node instances, i.e., the System, Interface, IpAddress, Network, Connection, and ConnectionLess objects, six Edge instances, and several Property instances as shown in Fig. 2. The System object is designed to manage physical network device information equivalent to MIB information. Attribute information related to the system (e.g., sysName and sysDesc in MIB-II [7]) can be modeled in the Property instance. The Interface object is able to manage physical interfaces such as the physical port and logical interfaces such as the logical port (e.g., the Loopback interface). As with the System object, attributed information related to the Interface object is stored in the Property object. Likewise, the IpAddress object is designed to manage IP address information related to physical and logical interfaces. In addition, the model is required to express connections between networks such as routing information and logical connection information. The GRI model is designed to model them as a Network object (e.g., routing information), Connection object (e.g., logical connection such as the MPLS path), and ConnectionLess object (e.g., VLAN). In addition to this, linkages between the Node instances are represented using the Edge instance, which is, for example, the relationship between the Interface object and the IpAddress object (e.g., ConsistOf object). As defined in the GRI model, RDStore is available to manage and store all network topology in a distributed graph DB.

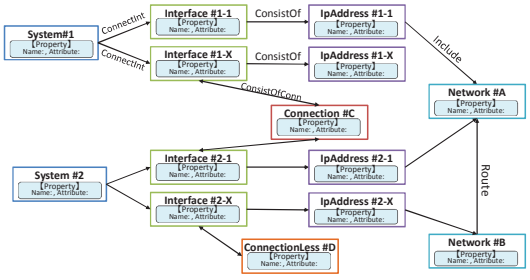

Fig. 2. Inheritance tree of GRI model

D. *Adapter module*

An adapter module is designed to control a collection of network resource data via SNMP [5] and/or CLI, and filters unnecessary data. Regarding collection, there are two collection patterns to maintain resource data freshness. One is the "all data collection" pattern that enables all the data to be obtained with the network device. The second is the "partial data collection" pattern that has a mechanism to update the network resource data when the network topology is changed by modification of the network configuration or a network failure. The second pattern is performed by receiving SNMP traps. Regarding unnecessary data filtering, it is expected to reduce the amount of resource data passed to the data processing module. In the current implementation, the filter conditions can define the data required for creating network topology such as OID information.

E. *Data processing module*

The data processing module has functionalities to provide distributed data processing in each network device, and to integrate all the resource data with other data processing modules in each network device via the DB in a distributed manner. First, this module converts the input format received from an adapter module into a format required for input to a DB module. This process means that conversion into a graphed data structure is required for insertion into the GRI model since the received resource data is based on the OID structure. After that, the converted formats are distinguished as three types of requests: one is a registration request that can register the new resource in the DB; the second one is a deletion request; and the last one is an update request that can update specific resource data.

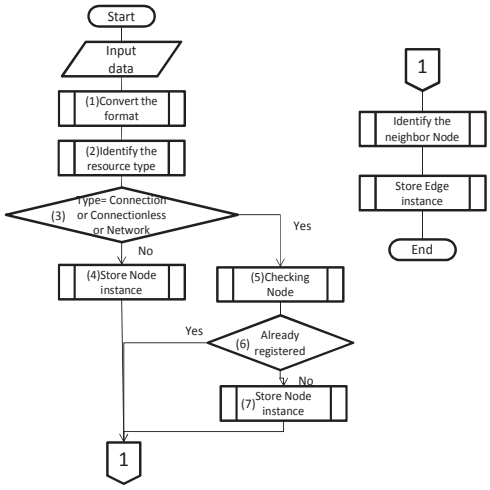

Fig. 3. Flowchart of the registration process in the data processing module

Figure 3 depicts a flowchart of the registration process excuted in one devices. After converting the data format (Fig. 3 (1)), the module looks up the name of the Node instance to divide the process into two processes (Fig. 3 (2) and (3)). If the Node instance includes Connection, ConnectionLess, or Network objects, the module checks whether the same resource data is already registered in the distributed DB (Fig. 3 (5) and (6)). This process is important for avoiding registration of duplicated data in a distributed DB. When there is no data yet registered in the DB, the module sends a registration request to the DB for the creation of a new Node instance (Fig. 3 (7)). Conversely, if there is already registered data, no registration of the Node instance is executed because the module judges that another data processing module in a different network device has executed the registration process. In contrast, if the Node instance is a System, Interface, or IpAddress object, the module sends a registration request to the DB in order to save the data as a Node instance (Fig. 3 (4)). This means modules implemented in different network devices cannot register such data related to physical resources since it is device-specific data. Finally, the module analyzes the relationship between resource data to create an Edge instance that can connect to each Node instance. After searching, the module sends the registration to the DB for storing the Edge instance such as a Route object, ConnectInt object, and so forth.

III. PERFORMANCE EVALUATION

A. *Experimental configuration*

In order to evaluate the availability of our proposed RDStore, we implemented a prototype of the RDStore in four routers that have server blade capability (CPU: Intel Core2 Solo 1.86 GHz, memory: 4 GB, disk size: 500 GBytes). To evaluate this prototype system, a simple network testbed that simulates a telecom network was configured as shown in Fig. 4. In this testbed, we assumed two types of VPN services including IP and Ethernet VPN services. The IP network consisted of four routers with several IP technologies such as MPLS-TE, MPLS-LDP and OSPF, and the Ethernet network was also composed of four Ethernet switched with VLAN technology.

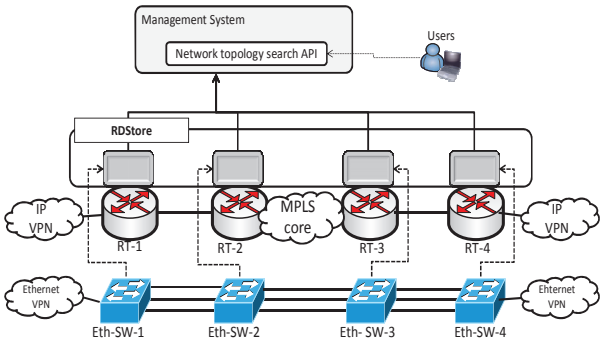

Fig. 4. Network configuration for the demonstration

To manage the network topology information in the proposed RDStore, an adapter, data processing, and graph-based distributed DB functions were deployed in each router, which collected network information, such as the physical port, logical port, IP address, MPLS connection, OSPF routing information, and so forth by using SNMP and CLI. As for the Ethernet network, since Ethernet switches have no server blade capability, this means the structure of sharing the RDStore is required to collect Ethernet network information as shown in Fig. 4. Thus, the functionality of the RDStore with routers was provided to Ethernet switches. Resource information was finally stored in a graph-based distributed DB based on the

GRI model. The modules are written in Java language and a graph-based DB [8] was used to cope with the high volume of transactions. The management system was deployed using an open source software [9] in order to display the network topology collected through the network topology search API.

B. Demonstration of RDStore

An RDStore was performed to verify the efficiency of our proposed one in the network testbed. When the RDStore was started, the network resource data was collected including the IP network as well as Ethernet network. The adapter module could receive approximately 5,000 data as the total number of resource data, and then filtered the 1,000 useless data such as packet counter data. The data processing module was able to process the data, and sent 3,000 registration requests to the distributed DB for storing the large number of Node, Edge and Property instances. This means the RDStore could reduce 1,000 requests by an analysis of the resource data using the registration process as shown in Fig. 3. Finally, the management system could collect and display not only the IP network but also the Ethernet network on the management system through a network topology search API. Therefore, we confirmed that the proposed RDStore successfully performed the management of network topology inside network devices.

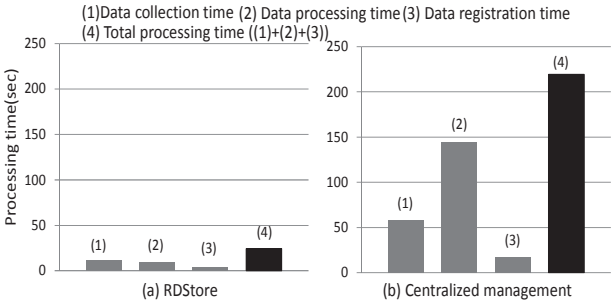

Fig. 5. Processing time of each function

In order to evaluate the effectiveness of the RDStore, each processing time must be verified compared with a centralized management system. Figure 5 indicates the processing time taken to process 25,000 resource data on each function such as a data collection, data processing and data registration. Note that Fig. 5 (a) shows the averaged time which was calculated by measuring the processing time based on four routers because the RDStore consisted of four routers as shown in Fig. 4, and Fig. 5 (b) indicates the processing time executed by a single centralized system implemented in a server (CPU: Intel Core (TM)2 Quad CPU 2.40GHz). The performance results showed that the total processing time (i.e., (4) in Fig. 5) of the RDStore was reduced to 1/10 compared with the centralized management. In particular, the data processing time (i.e., (2) in Fig. 5) of the RDStore was reduced to 1/16 compared with the centralized management. The improvement of the processing time has two major reasons; one is that the reduction of the data processing time is exactly the effect of the distributed graph DB which has a great advantage of generating resource graph compared to the conventional relational DB, and the second one is amount of processing data was reduced to 1/3 by adaptor module, compared to the centralized management system. By contrast, if we assume managing small scale network (e.g., 100 resource data), there is the possibility of break-even point where the centralized approach is superior. However, the result shows that our approach is superior to the centralized approach in case of the large scale network (e,g., over 10,000 resource data).

IV. CONCLUSION

In this paper, we proposed a distributed datastore for in network resource information sharing called RDStore, which enables the offloading of data processing and implementation of a distributed DB inside network devices for reducing the delay in comprehending network topology. In addition, we also presented a simple network resource model based on a resource graph, called the GRI model, to represent all network topology in the distributed DB and to effectively access network data. Finally, we evaluated the effectiveness of our proposed RDStore using the prototype system implemented in four network devices. The result of demonstration showed that the proposed RDStore successfully collected and managed network topology information inside network devices, and its processing time of comprehending network resources was significantly improved compared with the conventional centralized management system. Thus, the proposed RDStore is expected to facilitate scalable network management toward more complex network environments. In this paper, we only focused on resource management because the delay in comprehending network topology leads to major problems to achieve the fault and performance management, but the fault and performance management is also important because the amount of alarm data and performance data increases in association with the increase of network resource data. In our future work, the RDStore is required to support the fault and performance management such as an alarm mask, root cause analysis and performance monitoring on a real-time basis.

REFERENCES

[1] M. Miyazawa, K. Nishimura, "Scalable Root Cause Analysis Assisted by Classified Alarm Information Model Based Algorithm," Proc of IEEE/IFIP CNSM 2011, pp.1-4, 2011.

[2] K.-S. Lim, R. Stadler, "Real-time Views of Network Traffic Using Decentralized Management," Proc of IEEE/IFIP IM2005, pp.119-132, 2005.

[3] K.-S. Lim, R. Stadler, "Weaver: Realizing a Scalable Management Paradigm on Commodity Routers," Proc of IEEE/IFIP IM2003, pp.119-132, 2003.

[4] M. Uddin, R. Stadler, A. Clemm, "Management by Network Search," Proc of IEEE/IFIP NOMS2012, pp. 146-154, 2012.

[5] J. Case, M. Fedor, M. Schoffstall, J. Davin, "A Simple Network Management Protocol (SNMP)," RFC1157, IETF, May 1990.

[6] D. Dudkowski, M. Brunner, G. Nunzi, C. Mingardi, C. Foley, M. Poncede Leon, C. Meirosu, S. Engberg, "Architectural Principles and Elements of In-Network Management," Proc of IEEE/IFIP IM2009, pp.529-536, 2009.

[7] K. McCloghrie, M. Rose, "Management Information Base for Network Management of ICP/IP-based internets:MIB-II," RFC1213, IETF, March 1991.

[8] http://objectivity.com/products/infinitegraph/overview

[9] http://www.opennms.org