Suggested Running Head: Network Management Using XDD

Network Management Using XDD

Pagaporn Pengsart and Vilas Wuwongse[1]

[1] Computer Science & Information Management Program, Asian Institute of Technology, Pathumthani 12120, Thailand, Tel: +66-2-5245704, Fax: +66-2-5245721, Email: ccpea@mahidol.ac.th, vw@cs.ait.ac.th

Network Management Using XDD

ABSTRACT

The complexity of computer networks induces perplex management which requires expertise, error-prone configurations and lots of time and expenditure, whence management automation by means of network self-configuration is the answer. A new fundamental network management framework for self-managing systems is proposed and applied to distributed network management. It employs a declarative approach to the construction of network management by means of XML Declarative Description (XDD) and Common Information Model (CIM) – a standard network management model. With the former's expressiveness and knowledge inference capabilities, it uniformly represents network management facts, axioms, constraints and rules of network management knowledge, while the latter's interoperation with existing network management standards and protocols provides a framework with the ability to share management information on a multi-platform distributed application environment. An application to self-configuration management is given to demonstrate the effectiveness of the framework.

KEY WORDS: Distributed network management; network management framework; XML Declarative Description; CIM.

1. INTRODUCTION

In essence, network management involves monitoring, controlling, and maintaining network's components by collection and analysis of their data [1]. Nowadays, networks are growing in size and complexity in several dimensions, such as network topology, services, and protocols. Critical for proper deployment and management is the need to provide consistency and control over dynamic changes, a limitation of the impact that such changes have on performance and stability, required for robust communication.

Managing current networks is a complex and cumbersome task. Existing network management software only provides simple monitoring tools, so that management operations rely largely on the experience and expertise of individual operators. This approach encounters several problems concerning cost and time taken and is also error prone. In order to alleviate these problems, a management system must have the ability to automatically adapt to a variety of changes related to network conditions, network services, and user behavior. Most present networking research occurs along these lines [2, 3, 4, 5, 6].

Moreover, traditional network management protocols such as the OSI reference model, the Simple Network Management Protocol (SNMP), and the Telecommunications Management Network (TMN) are no longer adequate for the management of current networks. Having thrived on centralized or weakly distributed hierarchical models, they should be replaced by distributed network management to ensure scalability, efficiency and flexibility of management, discussed clearly in [7, 8]. Moreover, network devices have increasing processing power and capabilities so that distributed management becomes feasible by introduction of intelligence into their managed systems.

The present aim is to propose a new fundamental network management framework with self-management ability, aimed at inter-operation with other existing network management standards and protocols, and to represent network functional management knowledge. It can

be employed seamlessly in distributed network management. The framework is achieved by means of the Common Information Model (CIM) [9] for network modeling and XML Declarative Description (XDD) [10] for network knowledge representation. CIM, a standard network management model developed by the Distributed Management Task Force (DMTF), provides common descriptions of management information. XDD is a unified XML-based knowledge representation with well defined semantics and reasoning mechanisms (cf. Appendix for a review of its theory).

Section 2 describes the characteristics of self-managing systems and related work, Section 3 presents a framework for network management. Section 4 applies this framework to management functions integration such as configuration, performance and QoS management, and demonstrates it with an IP video conferencing (IPVC) example, Section 5 presents an architecture of employing this framework in the distributed network management system, and Section 6 draws conclusions and suggests further work.

2. SELF-MANAGING SYSTEM OF NETWORK MANAGEMENT

Recent activities in network management aim at building self-managing and self-organizing networks with the purpose of automation of some or all of the tasks typically carried out by an administrator or a management system. Characteristics of self-management with application examples and related work follow.

2.1. Characteristics of Self-Management

A self-managing system has the ability to determine best how to manage and control itself. Based on the OSI network management functions [11, 12], network self-managing systems can be classified into:

- *Self- configuration management*. The system has the capability to adapt and configure the network automatically to dynamically changing the environment.

- *Self- fault management*. The system handles error conditions that cause the loss of full functionality of a network resource. The mechanisms of this ability can be used to provide automatic service restoration, hence service survivability, in the event of network facility failures in the network.

- *Self- performance management*. The system is able to monitor, measure availability of resources, measure the use of and adjust resources automatically.

- *Self- security management*. The system has the ability to anticipate, detect, identify, and protect against attacks from anywhere.

- *Self- accounting management*. The system needs to be able to track the use of network resources by a user or a user class automatically for making efficient utilization of the network.

Automatic computing has been proposed [13] as an approach to reducing the cost and complexity of managing Information Technology (IT) infrastructure. An autonomic system is one which is self-configuring, self-healing, self-optimizing, and self-protecting, which conforms to the above characteristics.

Examples of typical tasks of current network administration requiring self-management ability include:

- Introduction of new network services, such as video conferencing, voice and video on demand.

These applications require a large bandwidth and QoS guarantee. When they are introduced to networks, they may have failed to work and have slowed down operations. An administrator needs to configure related network devices in support of their requirements.

- Virus protection at network level.

A virus can generate high volume random traffic patterns saturating networks. The result may be degradation in device performance such as the CPU and impact on memory

resources. The administrator must configure all network devices via access control lists (ACLs) to block virus attacks on ports.

- A fault event of a link or device on some paths or interfaces.

For the specific domain of network management, particularly for high-speed or critical networks, response time between the creation of failure, its detection, its analysis and its correction should be extremely short. Only a high-level of automation in management can allow such response time and correct problems properly.

Other routine tasks, such as changes of certain configurations of devices, have to be carried out frequently by an administrator. These tasks are cumbersome and, moreover, require fast and accurate response. There exist various approaches to a provision of fully automated techniques for dynamic networks in which there occur frequent and unanticipated changes.

2.2. Related Work

There has been conducted much research in automated management. Examples occurred in Active Networks [2, 3, 4], Network Self Management and Organization (NESTOR) [5], and Self-configuration of Networks (SELFCON) [6].

Active Networks [2, 3, 4] involve particularly fast dynamic of changing element configurations due to downloading and executing of Active Applications (AAs). An AA needs to configure its own parameters, and change those of its Execution Environment (EEs); the EE; in turn, they may have to change node configuration parameters automatically. Efforts in this area include the SENCOMM project [14], and the ANCORS project [15].

NESTOR [5] has been demonstrated in the automation of network configuration [5], network security [16], multimedia QoS, and Active Network applications [17]. It is an architecture for network management automation, which automates configuration management by means of policy scripts with access to and manipulation of respective

network elements. The configuration models are expressed in terms of Resource Definition Language (RDL). Constraints on configuration objects and relationships, which enrich the model and can be used to automate detection and reaction to inconsistencies, is expressed using Constraint Definition Language (CDL). Assignment of values to configuration model objects based on the configuration of related objects employs Policy Definition Language (PDL).

SELFCON [6] has been demonstrated in self-configuration in response to changed configuration policies. It is based on DEN specification, which is widely applied as the standard to model network elements and services.

3. A FRAMEWORK FOR NETWORK MANAGEMENT

Figure 1 depicts the present approach to self-management framework. Its components are:

(1) General Network Management

As a top-level conceptual model for semantic description of network capabilities and status, it enables each specific network management system to introduce and define a new domain – specific network capabilities and status as well as their properties and relationships. Therefore, it allows to instantiate specific network management such as configuration, fault, performance, security and accounting management.

(2) Static Network Capability

As static network information, independent of network conditions, it can be classified into network resources and service functions. Resources deal with network components: users, applications and network elements such as the number of link paths, and link bandwidths. Examples of service functions are the network's routing, mail service and bridging functions.

(3) Dynamic Network Capability

Defined as dynamic network information, derived from the network usage status at a certain time, it involves such information as the number of users concurrently connected to the network, traffic statistics and available link bandwidths.

Both static as well as dynamic network capabilities, definitions and instances can be represented by the xmlCIM language [18].

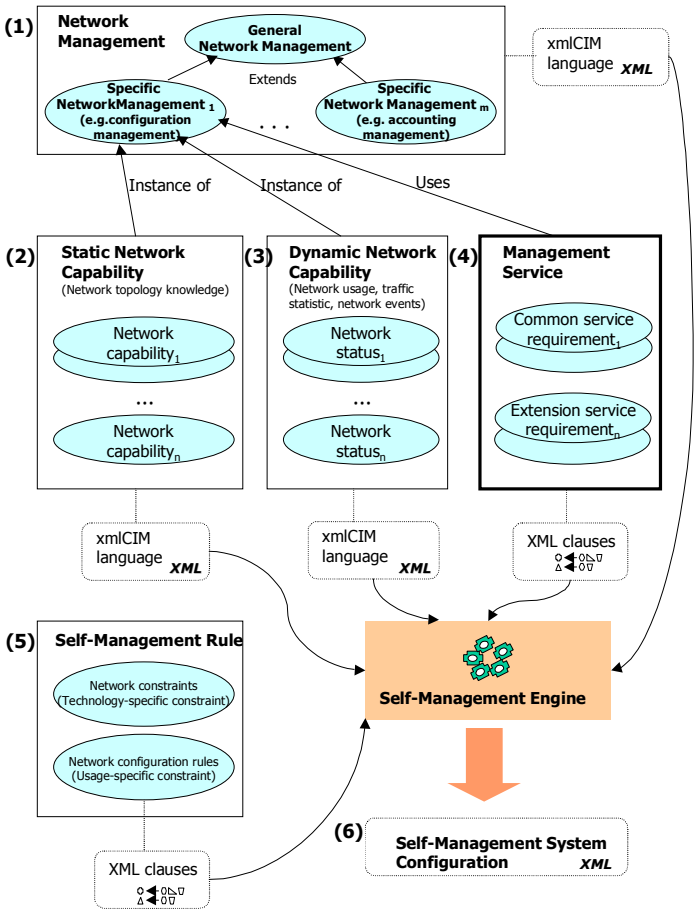


Figure 1: A framework of network management using XDD.

(4)Management Service

Network management service covers a broad area related to the operational and design aspects of a production network. It can be grouped into two services - common and extension services. Common service is a set of basic functions of management encompassing items such as monitoring, data collection, event and notification handling. Extension service is a set of

special management functions. Examples are configuration and code management for devices utilized to support the network, proactive monitoring of devices and links, analysis of events, resource utilization of devices and links as well as data collection, correlation and analysis for planning purposes.

In order to describe network management services as well as to represent inherent interrelationships and complex constraints and axioms on elements in a domain, XDD theory [10] is employed.

(5)Self -Management Rule

It contains the knowledge of network configuration for semi-automatic management in two parts: network constraints and network configuration rules. The former are technology-specific constraints over the valid values of managed elements and network events such as "one interface of a device can have one MAC address", while the latter are usage-specific constraints over the valid network rules such as "all computers using video conferencing must exist in network number 10.10.19.0/24".

This self-management rule is represented in XDD language as a corresponding set of XML clauses. The head of a clause identifies components of the configuration, while the body describes configuration restrictions including network constraints and configuration rules.

(6)Self-Management System Configuration

Whenever there occurs a network problem, the system will check the self-management rules based on functions of the management service and yield a list of possible configurations.

4. AN APPLICATION OF THE PROPOSED NETWORK MANAGEMENT FRAMEWORK

As an example, it is assumed that self-configuration management is the only specific network management required. Mechanisms are presented for modeling each component in

the proposed framework and for computing a list of possible configurations of a self-configuration management system.

4.1. Example : An IP Video Conferencing

The example - a demonstration of using an IP video conferencing (IPVC) - represents bandwidth-sensitive data applications. The IPVC is a process of conducting a conference between two or more participants at different sites by means of computer network transmission of audio and video data. The application requires proper bandwidth resources and QoS guarantee, e.g., its delay should be no more than 150 ms, its loss should be no more than 1%, and its minimum bandwidth guarantee is 384 kpbs per session [19]. The network-based quality of service (NQoS) service level is guaranteed service which requires a reservation of network resources in order to meet specified service classification requirements.

An administrator needs to provide this resource requirement by configuring all related network devices, such as routers and switches, with enabling and running Resource ReSerVation Protocol (RSVP), specified in RFC 2205 [20], in order to deliver QoS requests to all nodes along the path(s) of the traffic flows. RSVP requests will generally result in resources being reserved at each node along the data path.

Consider the IPVC system and its network configuration depicted in Figure 2. A user can employ this system anywhere by connecting an IPVC client at a port of LAN switch and setting a computer with IP address of IPVC subnetwork. The video conference server is installed at Site A. There are four sites connected with routers running via lease line at the speed of 8 Mbps. LAN switches are installed at every site and connected to the routers at the speed of 100 Mbps. Three layer-3-based VLAN [21] subnets are connected to each LAN switch, one for the IPVC system and two for other applications.

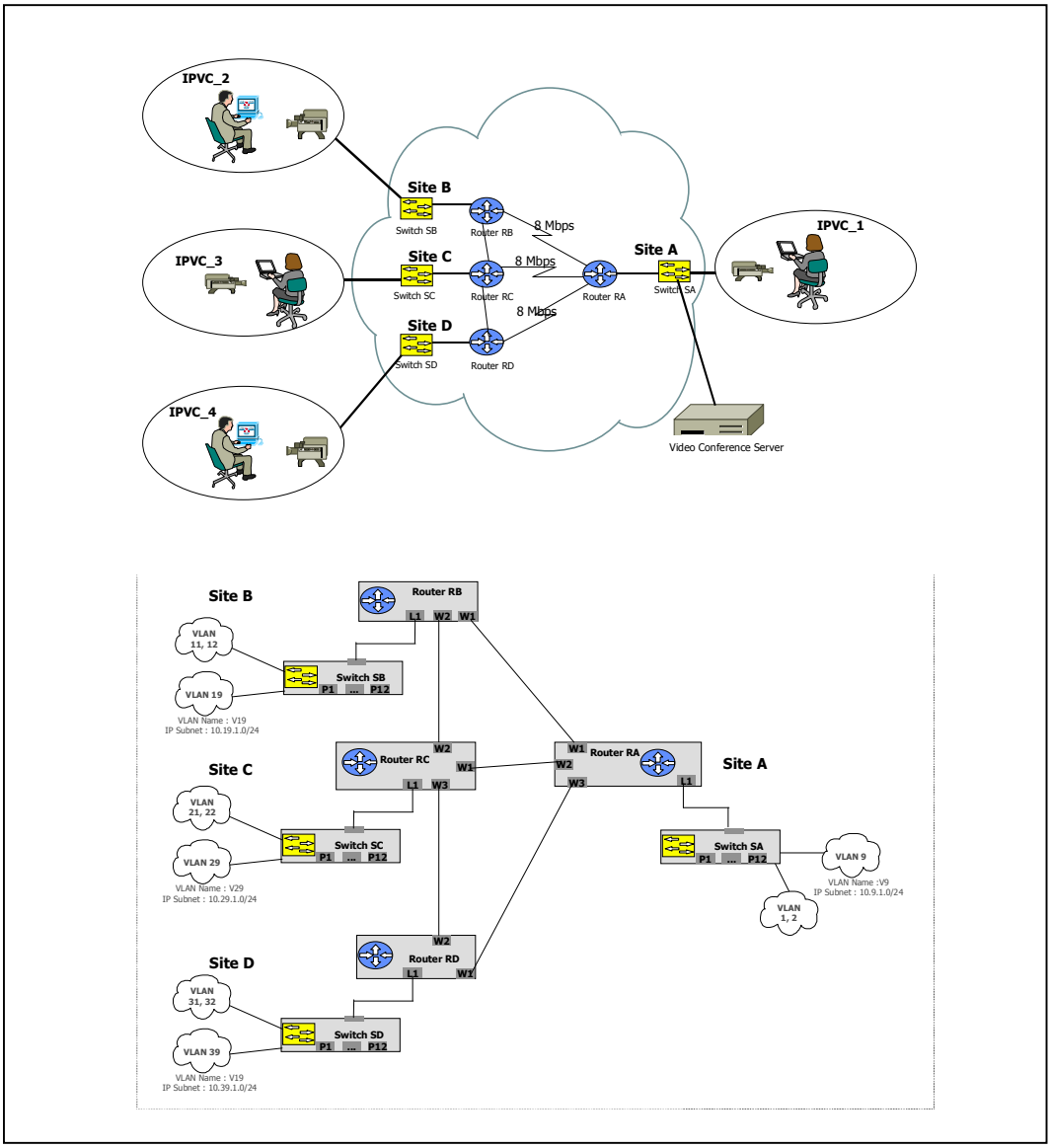


Figure 2: IP video conferencing system and its network configuration.

For example, the two participants, IPVC_2 at site B and IPVC_1 at site A, speak to one another. The IPVC_2 client is connected to a port of LAN Switch SB and configures the computer with IP address of 10.19.1.1/24. And the IPVC_1 client is connected to a port of LAN Switch SA and configures the computer with IP address of 10.9.1.1/24. The traffic flow of this session is from Switch SB and Router RB to Router RA and Switch SA.

As a result, the administrator needs to enable RSVP and to configure its parameters at Switch SB, Router RB, Router RA and Switch SA. Moreover, he/she needs to check the status of available capability of all related network devices and bandwidth of link in order to

operate most efficiently and effectively. Repeatedly, if also other users employ this system, the administrator has to carry out again this routine task.

4.2. Network Facts : Static and Dynamic Network Capabilities

In order to create an open data exchange environment for network management, any parties exchanging information must agree on the definition of network system entities and their interrelationships. Common Information Model (CIM) will be employed to facilitate the integration of independently developed components into the network management system. CIM, which specifies common semantics for network resources, their attributes and relationships, is a document - a set of class diagrams using the Unified Modeling Language (UML), which specifies it in an abstract manner with due allowance for open implementations (i.e., without restriction to relational or object oriented or other modeling techniques). Figure 3 shows a fragment of the CIM class diagram in UML notation, related to the above example.

The ComputerSystem class is used to model a network device such as router and switch. It contains a list of source and destination IP addresses from IPHeaderFilter class with FilterEntryInSystem relationship. The router interfaces and switch ports are modeled as network services with ServiceAccessPoint, an abstract class representing a service. ProtocolEndpoint is a derived class of ServiceAccessPoint, used to model router interfaces and switch ports. A communication endpoint, e.g., computer interfaces connecting to a switch, is represented by LANEndpoint and connecting Switchable relationship to SwitchPort. The base class of the SwitchPort and LANEndpoint is the ProtocolEndpoint class, defined to represent a communication point from which data may be sent or received. The ActiveConnection association defines a connection which currently carries traffic, e.g., unicast or multicast applications, between two ProtocolEndpoints. StatisticalData is a class for any arbitrary collection of statistical data and/or metrics applicable to one or more ManagedElement. Some parts of network capability status information, e.g., traffic usage, are derived from this instance.

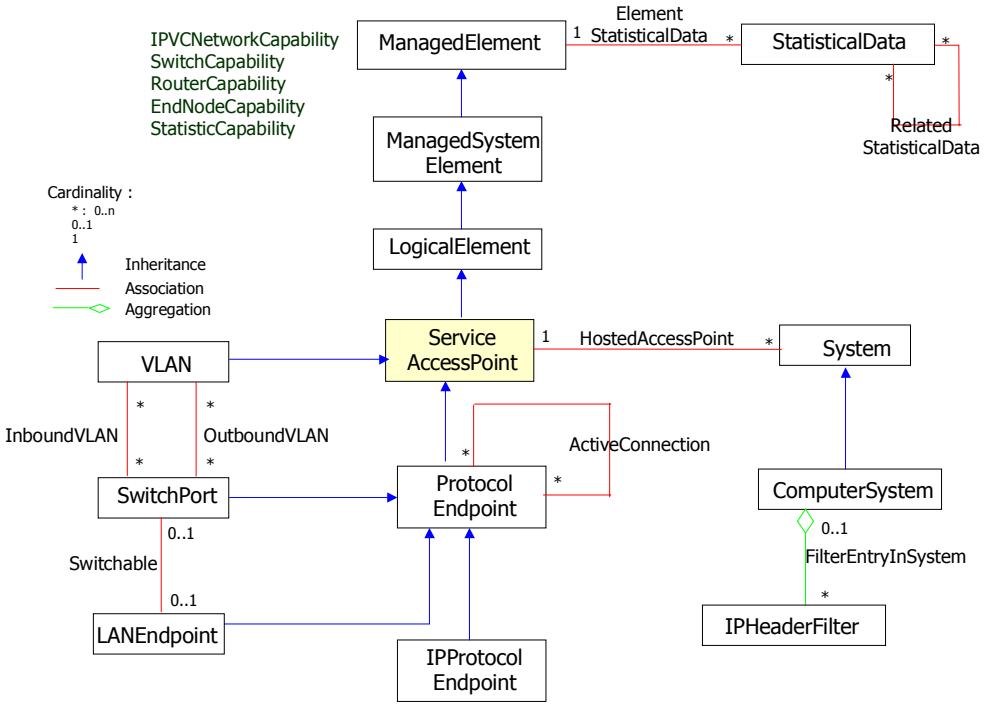


Figure 3: A fragment of the CIM network information model.

This example involves five basic classes for the representation of network capabilities: IPVCNetworkCapability, SwitchCapability, RouterCapability, EndNodeCapability, and StatisticCapability. IPVCNetworkCapability, inherited from the VLAN class, represents the IPVC network system. SwitchCapability and RouterCapability, inherited from the ComputerSystem class, represent a switch and a router, respectively. EndNodeCapability, inherited from the LANEndpoint class, represents a computer connected to the network, and StatisticCapability, inherited from the StatisticalData class, represents statistical data.

In the proposed approach, a description of network information knowledge, encoded in xmlCIM [18] language, becomes immediately an XDD description comprising solely ground XML unit clauses. For example, consider the given xmlCIM class and its instance of Figure 4, called XDD description, relating to information of Figures 2 and 3.

C_1:

```
<CLASS NAME="SwitchCapability" SUPERCLASS="CIM_ComputerSystem">
<PROPERTY NAME="NameFormat" CLASSORIGIN="CIM_ComputerSystem"
TYPE="string">
        <VALUE>NameFormat</VALUE>
</PROPERTY>
<PROPERTY.ARRAY NAME="Dedicated" TYPE="uint16"
CLASSORIGIN="CIM_ComputerSystem">
<QUALIFIER NAME="Values" TYPE="string" OVERRIDABLE="false"
TOSUBCLASS="false">
        <VALUE.ARRAY>
        <VALUE/>
        </VALUE.ARRAY>
</QUALIFIER>
</PROPERTY.ARRAY>
<PROPERTY.ARRAY NAME="HdrSrcAddress" CLASSORIGIN="CIM_IPHeadersFilter"
TYPE="uint8">
<VALUE/>
</PROPERTY.ARRAY>
<PROPERTY.ARRAY NAME="HdrDestAddress" CLASSORIGIN="CIM_IPHeadersFilter"
TYPE="uint8">
<VALUE/>
</PROPERTY.ARRAY>
…
</CLASS> ←.
```

```
<INSTANCE CLASSNAME="SwitchCapability" SUPERCLASS="CIM_ComputerSystem">
<PROPERTY NAME="NameFormat" CLASSORIGIN="CIM_ComputerSystem"
TYPE="string">
        <VALUE>Switch SB</VALUE>
</PROPERTY>
<PROPERTY.ARRAY NAME="Dedicated" TYPE="uint16"
CLASSORIGIN="CIM_ComputerSystem">
<QUALIFIER NAME="Values" TYPE="string" OVERRIDABLE="false"
TOSUBCLASS="false">
        <VALUE.ARRAY>
        <VALUE>Layer 3 Switch</VALUE>
        </VALUE.ARRAY>
</QUALIFIER>
</PROPERTY.ARRAY>
<PROPERTY.ARRAY NAME="HdrSrcAddress" CLASSORIGIN="CIM_IPHeadersFilter"
TYPE="uint8">
<VALUE>10.10.19.1</VALUE>
</PROPERTY.ARRAY>
<PROPERTY.ARRAY NAME="HdrDestAddress"
CLASSORIGIN="CIM_IPHeadersFilter" TYPE="uint8">
<VALUE>10.10.9.1</VALUE>
</PROPERTY.ARRAY>
…
</INSTANCE> ←.
```

(1) An xmlCIM example – definitions of the class **SwitchCapability** and its properties in the left column and an instance example of **switch SB** in the right column

C_2:

```
<CLASS NAME="EndNodeCapability" SUPERCLASS="CIM_LANEndpoint">
<PROPERTY NAME="MACAddress" CLASSORIGIN="CIM_LANEndpoint">
        <VALUE/>
</PROPERTY>
<PROPERTY NAME="Address" CLASSORIGIN="CIM_IPProtocolEndpoint"
TYPE="string">
        <VALUE/>
</PROPERTY>
<PROPERTY NAME="TrafficType" CLASSORIGIN="CIM_ActiveConnection">
        <VALUE/>
</PROPERTY>
<PROPERTY NAME="NameFormat" CLASSORIGIN="CIM_ComputerSystem"
TYPE="string">
        <VALUE>NameFormat</VALUE>
</PROPERTY>
<PROPERTY NAME="SlotNumber" CLASSORIGIN="CIM_SwitchPort">
        <VALUE/>
</PROPERTY>
<PROPERTY NAME="PortNumber" CLASSORIGIN="CIM_SwitchPort">
        <VALUE/>
</PROPERTY>
</CLASS> ←.
```

```
<INSTANCE CLASSNAME="EndNodeCapability" SUPERCLASS="CIM_LANEndpoint">
<PROPERTY NAME="MACAddress" CLASSORIGIN="CIM_LANEndpoint">
        <VALUE>00:01:01:0A:0A:01</VALUE>
</PROPERTY>
<PROPERTY NAME="Address" CLASSORIGIN="CIM_IPProtocolEndpoint"
TYPE="string">
        <VALUE>10.10.19.1</VALUE>
</PROPERTY>
<PROPERTY NAME="TrafficType" CLASSORIGIN="CIM_ActiveConnection">
        <VALUE>Multicast</VALUE>
</PROPERTY>
<PROPERTY NAME="NameFormat" CLASSORIGIN="CIM_ComputerSystem"
TYPE="string">
        <VALUE>Switch SB</VALUE>
</PROPERTY>
<PROPERTY NAME="SlotNumber" CLASSORIGIN="CIM_SwitchPort">
        <VALUE>3</VALUE>
</PROPERTY>
<PROPERTY NAME="PortNumber" CLASSORIGIN="CIM_SwitchPort">
        <VALUE>1</VALUE>
</PROPERTY>
</INSTANCE> ←.
```

(2) An xmlCIM example – definitions of the class **EndNodeCapability** and its properties in the left column and an instance example of **Endnode address 10.10.19.1** in the right column

C_3:

```
<CLASS NAME="StatisticCapability" SUPERCLASS="CIM_StatisticalData">
<PROPERTY NAME="InstanceID" CLASSORIGIN="CIM_StatisticalData">
        <VALUE/>
</PROPERTY>
<PROPERTY NAME="ElementName" CLASSORIGIN="CIM_StatisticalData">
        <VALUE/>
</PROPERTY>
<PROPERTY NAME="NumberOfPacketsLost" CLASSORIGIN="CIM_StatisticCapability">
        <VALUE/>
</PROPERTY>
<PROPERTY NAME="Delay" CLASSORIGIN="CIM_StatisticCapability">
        <VALUE/>
</PROPERTY>
<PROPERTY NAME="Availability" CLASSORIGIN="CIM_StatisticCapability">
        <VALUE/>
</PROPERTY>
</CLASS> ←.
```

```
<INSTANCE CLASSNAME="StatisticCapability" SUPERCLASS="CIM_StatisticalData">
<PROPERTY NAME="InstanceID" CLASSORIGIN="CIM_StatisticalData">
        <VALUE>Switch SB</VALUE>
</PROPERTY>
<PROPERTY NAME="ElementName" CLASSORIGIN="CIM_StatisticalData">
        <VALUE>Switch SB</VALUE>
</PROPERTY>
<PROPERTY NAME="NumberOfPacketsLost" CLASSORIGIN="CIM_StatisticCapability">
        <VALUE>0</VALUE>
</PROPERTY>
<PROPERTY NAME="Delay" CLASSORIGIN="CIM_StatisticCapability">
        <VALUE>10</VALUE>
</PROPERTY>
<PROPERTY NAME="Availability" CLASSORIGIN="CIM_StatisticCapability">
        <VALUE>40</VALUE>
</PROPERTY>
</INSTANCE> ←.
```

(3) An xmlCIM example – definitions of the class **StatisticCapability** and its properties in the left column and an instance example of **Switch SB Statistic** in the right column.

C_4:

```
<CLASS NAME="IPVCNetworkCapability" SUPERCLASS="CIM_VLAN">
  <PROPERTY NAME="VLANNumber" CLASSORIGIN="CIM_VLAN">
        <VALUE.ARRAY>
              <VALUE/>
        </VALUE.ARRAY>
  </PROPERTY>
  <PROPERTY NAME="Address" CLASSORIGIN="CIM_IPProtocolEndpoint">
        <VALUE.ARRAY>
              <VALUE/>
        </VALUE.ARRAY>
  </PROPERTY>
</CLASS> ←.
```

```
<INSTANCE CLASSNAME="IPVCNetworkCapability" SUPERCLASS="CIM_VLAN">
  <PROPERTY NAME="VLANNumber" CLASSORIGIN="CIM_VLAN">
        <VALUE.ARRAY>
              <VALUE>9</VALUE>
              <VALUE>19</VALUE>
              <VALUE>29</VALUE>
              <VALUE>39</VALUE>
        </VALUE.ARRAY>
  </PROPERTY>
  <PROPERTY NAME="Address" CLASSORIGIN="CIM_IPProtocolEndpoint">
        <VALUE.ARRAY>
              <VALUE>10.10.9.0</VALUE>
              <VALUE>10.10.19.0</VALUE>
              <VALUE>10.10.29.0</VALUE>
              <VALUE>10.10.39.0</VALUE>
        </VALUE.ARRAY>
  </PROPERTY>
</INSTANCE> ←.
```

(4) An xmlCIM example – definitions of the class **IPVCNetworkCapability** and its properties in the left column and an instance example of **IPVC network** in the right column.

Figure 4: XDD descriptions C_1, C_2, C_3, and C_4, modeling network facts in xmlCIM.

4.3. Management Service

Management service of this application deals with supporting network QoS using RSVP [20]. The management service based on this application involves:

1. *Monitoring a packet using IPVC application*

This service monitors a packet using IPVC application and detects the packet which has the address in IPVC sub-network and using the multicast traffic type. The example above is 10.10.19.1 for the source packet connected at Switch SB and 10.10.9.1 for the destination packet connected at Switch SA.

2. *Finding the traffic flow from source to destination*

This service finds the traffic flow. The result of this example is Switch SB – Router RB – Router RA - Switch SA.

3. *Checking available resources of all related network devices*

This service checks available resources before configuring the related devices. The system gets availability, loss and delay parameters from StatisticCapability instance.

4. *Configuring the RSVP at all related network devices*

This service configures at all related devices by enabling RSVP and setting related parameters.

Monitoring a packet using IPVC application will be employed as a demonstration. Implicit information derived from the XDD description C_5 of Figure 5: "monitor all packets which use the IPVC application" will be illustrated. The body of C_5 contains three parts and specifies the condition of the rule that TrafficType is equal "Multicast" and SrcAddress is a member of IPVC network. Its head derives a MonitoringIPVCPackets-element with inferred DeviceCapability-, Device-, SrcAddress-, DestAddress-, TrafficType-, SlotNumber- and PortNumber-sub-elements.

```
C5:   <MonitoringIPVCPackets>
          <INSTANCE CLASSNAME="$S:DeviceCapability">
            <PROPERTY NAME="NameFormat">
              <VALUE>$S:Device</VALUE>        </PROPERTY>>
            <PROPERTY.ARRAY NAME="HdrSrcAddress">
                <VALUE>$S:SrcAddress</VALUE>      </PROPERTY.ARRAY>
            <PROPERTY.ARRAY NAME="HdrDestAddress">
                <VALUE>$S:DestAddress</VALUE>    </PROPERTY.ARRAY>
            <PROPERTY NAME="TrafficType">
              <VALUE>Multicast</VALUE>        </PROPERTY>
            <PROPERTY NAME="SlotNumber">
                <VALUE>$S:SlotNumber</VALUE>      </PROPERTY>
            <PROPERTY NAME="PortNumber">
                <VALUE>$S:PortNumber</VALUE>      </PROPERTY>
          </INSTANCE>
      </MonitoringIPVCPackets>

          ← <INSTANCE CLASSNAME="EndNodeCapability" $P:p1>
                $E:properties1
              <PROPERTY NAME="Address" $P:p2>
                <VALUE>$S:SrcAddress</VALUE>        </PROPERTY>
              <PROPERTY NAME="NameFormat" $P:p3>
                <VALUE>$S:Device</VALUE>      </PROPERTY>
              <PROPERTY NAME="TrafficType" $P:p4>
                <VALUE>Multicast</VALUE>        </PROPERTY>
              <PROPERTY NAME="SlotNumber" $P:p5>
                <VALUE>$S:SlotNumber</VALUE>      </PROPERTY>
              <PROPERTY NAME="PortNumber" $P:p6>
                <VALUE>$S:PortNumber</VALUE>      </PROPERTY>
            </INSTANCE>,

            <INSTANCE CLASSNAME="$S:DeviceCapability" $P:p11>
              <PROPERTY NAME="NameFormat" $P:p12>
                <VALUE>$S:Device</VALUE>          </PROPERTY>
              $E:properties2
              <PROPERTY.ARRAY NAME="HdrSrcAddress" $P:p13>
                <VALUE>$S:SrcAddress</VALUE>      </PROPERTY.ARRAY>
              <PROPERTY.ARRAY NAME="HdrDestAddress" $P:p14>
                <VALUE>$S:DestAddress</VALUE>    </PROPERTY.ARRAY>
              $E:properties3
            </INSTANCE>,

            <INSTANCE CLASSNAME="IPVCNetworkCapability" $P:p21>
              $E:properties4
              <PROPERTY NAME="Address"  $P:p22>
                <VALUE.ARRAY>
                  $E:properties5
                </VALUE.ARRAY>
              </PROPERTY>
            </INSTANCE>,

          StrLength( <string>$S:SrcAddress</string>, <length>$S:Result1</length> ),
          Replace( <string>$S:SrcAddress</string>, <startAt>9</startAt>,
                <endAt>$S:Result1</endAt>, <replaceStr>0</replaceStr>,
                <result>$S:Result2<result> ),
          Member( <value> $S:Result2</value>,  $E:properties5 >.
```

```
/ The management service is "monitoring
/ a packet using IPVC application".
/
/ This rule is to get an IPVC application's
/ packet with the information of device,
/ source and destination addresses, as
/ well as slot and port numbers of the
/ device by checking IP address and traffic
/ type.
/ IP address must exist in the IPVC
/ network, e.g., 10.10.19.0,
/ and traffic type must be multicast.
```

Figure 5: An example of management service rule.

4.4. Self-Management Rule

In the proposed system, such restrictions and relations are expressed, respectively, as constraints and axioms on the values of one or more network objects.

The network constraints, technology-specific constraints, of this example involve:

- The bandwidth of a WAN link requires 384 kbps per an IPVC client.
- If the bandwidth is less than *n*x384 kbps; *n* = number of clients, the RSVP is not set.

The network configuration rules, usage-specific constraints, involve:

- IPVC networks consist of 10.10.9.0, 10.10.19.0, 10.10.29.0, and 10.10.39.0.

- The capability of a device is available, if availability of device is greater than 30%, loss is less than 1% and delay is less than 150.

Figure 6 shows an examples of XML clause C_6 for representing the self-management rule.

```
C6:   <DeviceCapabilityOK>
        <PROPERTY NAME="InstanceID">
          <VALUE>$S:Device</VALUE>   </PROPERTY>
      </DeviceCapabilityOK>

      ← <INSTANCE CLASSNAME="StatisticCapability" $P:p1>
          <PROPERTY NAME="InstanceID" $P:p2>
            <VALUE>$S:Device</VALUE>        </PROPERTY>
           $E:Property
          <PROPERTY NAME="Availability" $P:p3>
            <VALUE>$S:Availability</VALUE>      </PROPERTY>
          <PROPERTY NAME="NumberOfPacketsLost" $P:p4>
            <VALUE>$S:Loss</VALUE>        </PROPERTY>
          <PROPERTY NAME="Delay" $P:p5>
            <VALUE>$S:Delay</VALUE>       </PROPERTY>
        </INSTANCE>,

        GT(<VALUE>$S:Availability</VALUE>, <VALUE>30</VALUE>),
        LT(<VALUE>$S:Loss</VALUE>, <VALUE>1</VALUE>),
        LT(<VALUE>$S:Delay</VALUE>,  <VALUE>150</VALUE>).

/ This network configuration rule reads
/ "the capability of a device is available, if
/ availability of device is greater than
/ 30%, loss is less than 1% and delay is
/ less than 150".
```

Figure 6: An example of self-management rule.

4.5. Self-Management System Configuration

In this example, whenever an IPVC packet monitoring is detected, the system will check the self-management rules based on functions of the management service and yield a list of possible configurations. An example of a query aiming at obtaining configuration parameters for the RSVP and an example of possible resultant configuration for Switch SB are shown in Figures 7 and 8, respectively. Figure 9 depicts how Switch SB is actually configured using the resultant parameters.

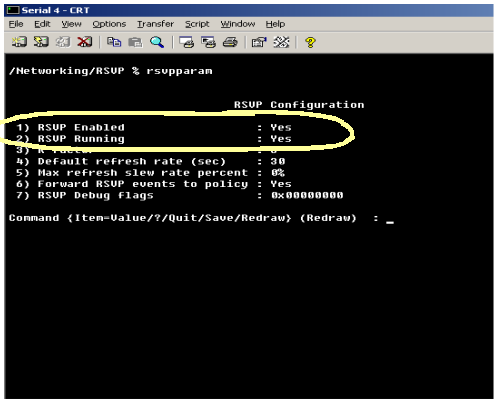
```
C7 :    <DeviceSetting>
          <INSTANCE CLASSNAME="$S:DeviceCapability">
            <PROPERTY NAME="NameFormat" >
              <VALUE>$S:Device</VALUE>
            </PROPERTY>>
            <RSVPEnabled>Yes</RSVPEnabled>
            <RSVPRunning>Yes</RSVPRunning>
            <QoSCondition>
              <SourceAddress>$S:SrcAddress</SourceAddress>
              <DestinationAddress>$S:DestAddress</DestinationAddress>
              <DeviceSlot>$S:SlotNumber</DeviceSlot>
              <DevicePort>$S:PortNumber</DevicePort>
            </QoSCondition>
            <QoSAction>
              <Bandwidth>
                <MinimumReservation>384</MinimumReservation>
                <MaximumReservation>512</MaximumReservation>
              </Bandwidth>
            </QoSAction>
          </INSTANCE>
        </ DeviceSetting>

          ←   <MonitoringIPVCPackets>
                <INSTANCE CLASSNAME="$S:DeviceCapability">
                  <PROPERTY NAME="NameFormat">
                    <VALUE>$S:Device</VALUE>       </PROPERTY>>
                  <PROPERTY.ARRAY NAME="HdrSrcAddress">
                    <VALUE>$S:SrcAddress</VALUE>      </PROPERTY.ARRAY>
                  <PROPERTY.ARRAY NAME="HdrDestAddress">
                    <VALUE>$S:DestAddress</VALUE>    </PROPERTY.ARRAY>
                  <PROPERTY NAME="TrafficType">
                    <VALUE>Multicast</VALUE>       </PROPERTY>
                  <PROPERTY NAME="SlotNumber">
                    <VALUE>$S:SlotNumber</VALUE>      </PROPERTY>
                  <PROPERTY NAME="PortNumber">
                    <VALUE>$S:PortNumber</VALUE>      </PROPERTY>
                </INSTANCE>
              </MonitoringIPVCPackets>,

              <DeviceCapabilityOK>
                <PROPERTY NAME="InstanceID">
                  <VALUE>$S:Device</VALUE>
                </PROPERTY>
              </DeviceCapabilityOK>.
```

/ This query reads "configuring the RSVP
/ at all related network devices".
/
/ This query finds the parameters :
/ **device**, **source and destination**
/ addresses, slot and **port numbers**
/ and **available device** .

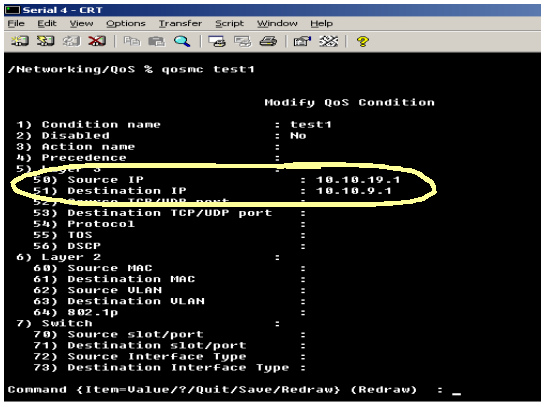
Figure 7: A query to obtain device configuration information.

```
<DeviceSetting>
  <INSTANCE CLASSNAME="SwitchCapability">
    <PROPERTY NAME="NameFormat">
      <VALUE>Switch SB</VALUE>
    </PROPERTY>>
    <RSVPEnabled>Yes</RSVPEnabled>                          / RSVP is enabled.
    <RSVPRunning>Yes</RSVPRunning>
    <QoSCondition>
      <SourceAddress>10.10.19.1</SourceAddress>             / Source and destination addresses of end nodes
      <DestinationAddress>10.10.9.1</DestinationAddress>
      <DeviceSlot>3</DeviceSlot>                            / Slot and port numbers of the switch
      <DevicePort>1</DevicePort>
    </QoSCondition>
    <QoSAction>
      <Bandwidth>
        <MinimumReservation>384</MinimumReservation>        / Bandwidth reservation (corresponding
        <MaximumReservation>512</MaximumReservation>        / sequence of values appears in Figure 9)
      </Bandwidth>
    </QoSAction>
  </INSTANCE>
</ DeviceSetting>
```

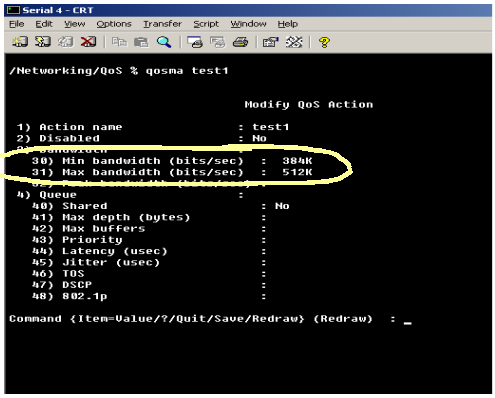
Figure 8: A possible configuration example of Switch SB.



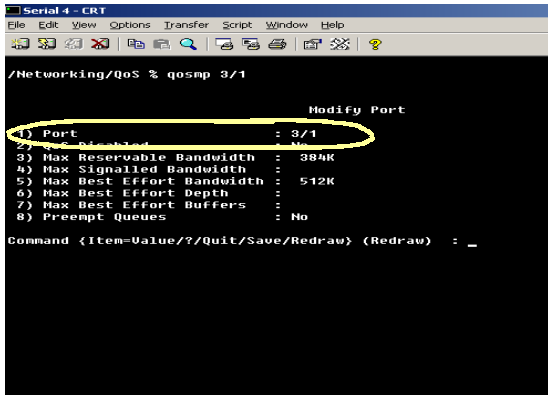
a) Configuration of RSVP parameters



b) Addition of QoS classification condition



c) Modification of physical port's QoS characteristics



d) Addition of QoS policy action

Figure 9: Parameter setting for RSVP at Switch SB.

5. EMPLOYMENT OF THE FRAMEWORK IN DISTRIBUTED NETWORK MANAGEMENT

The proposed approach to network management with self-management ability can be applied to distributed network management. The application together with required agent communication is presented.

5.1. An Architecture of Distributed Network Management Using the Framework

Figure 10 shows the proposed architecture comprising two layers and two types of communication. The two layers are for information structuring and classification, while the communication is for network management information exchange.

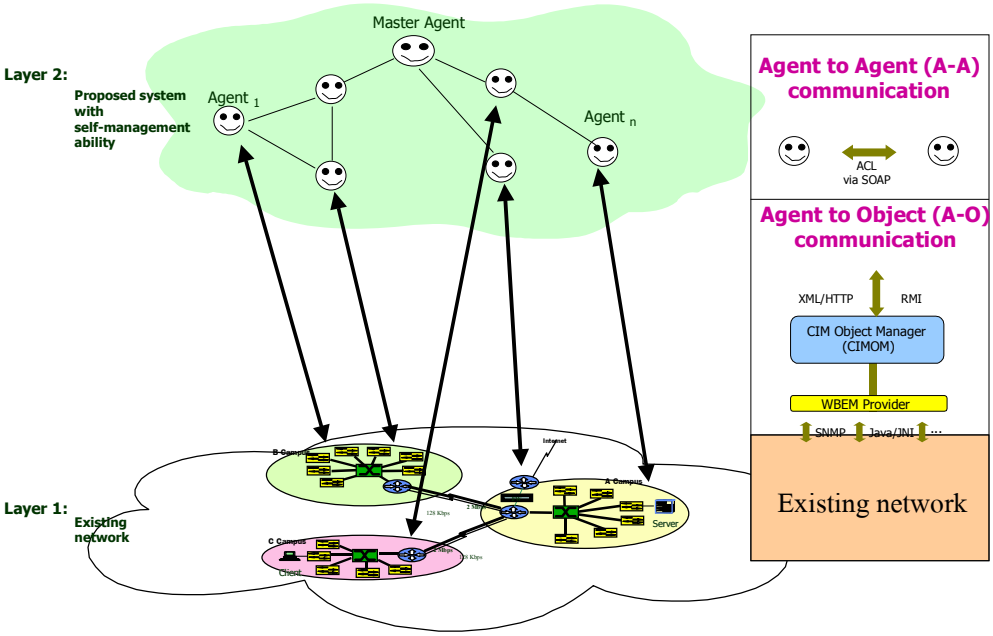


Figure 10: The proposed distributed network management architecture.

Layer 1 represents the existing physical network. There will be sub-networks inside the network. The network or sub-network can contain a single device or a group of devices. Layer 2 is viewed as a set of agents. There exists a single master agent acting as main controller and trusted agent for information exchange among agents. Each agent manages its physical network or domain of response, called Object. Objects can be a single device, group of devices, a sub-network or a group of sub-networks. In order to make a network manageable, there will be two types of communications - Agent to agent (A-A) and Agent to object (A-O).

5.2. Agent to Object Communication

The framework employs the CIM of DMTF. The DMTF runs the Web Based Enterprise Management (WBEM) initiative [22], which specifies an XML mapping for the CIM

(xmlCIM) [18] so that managed object classes and instances can be encoded in XML. Examples of this communication are presented with xmlCIM language in Section 4.2.

There are many advantages of this framework employed in Distributed Network Management. XML is being used for the definition and encoding of messages in data communications protocols [23]. Such messages could be transported in many different ways, including via TCP sockets or as parameters in simple method calls using distributed processing platforms such as CORBA, DCOM or RMI. Furthermore, XML and its transport using HTTP (XML/HTTP) does benefit from being readily deployable on the Internet, with accompanying proven availability, scalability and interoperability. In particular, with regard to interoperability, XML seems to be reached relatively easily, partially due to its flexibility, simplicity, and technology independence. The separation of information in XML from the mechanisms to present, transmit, and store that information also makes it a flexible solution for the specification and manipulation of corporate data.

The Agent to Object Communication will also use the WBEM, illustrated in Figure 10. The main WBEM components are the CIM Object Manager (CIMOM), the WBEM Providers and WBEM Clients. A WBEM Provider is the interface between a managed resource Object and the CIMOM via the existing management protocols such as SNMP, or Java/JNI. A WBEM Client is the interface between the manager Agent and the CIMOM using XML/HTTP or RMI.

5.3. Agent to Agent Communication

The unit of Agent to Agent (A-A) communication is called a Dialog. A single Dialog can contain a single message or a group of messages with parameters, and input-output can conform to function of agent. A-A communication provides a set of defined Dialogs to be used whenever appropriate. For instance, one agent sends a Dialog “request utilization of switch SB” to agent responsible for switch SB. Such a Dialog consists of “ask-one” message

with utilization of a switch SB as parameter. The received agent's reply - a "tell" message to the requested agent - consists of the content "50%".

In order to establish a support for management communication over the distributed network management and agent communication standards, one can employ SOAP to encode *XML-based ACL messages* [24] (either FIPA's ACL or KQML [25]), the contents of which can be, for example, an utilization request. Figure 11 gives an extended SOAP Envelope example which encodes a RequestUtilization request with a certain input of SwitchSB instance and Utilization property. After completion of the request execution, the output is returned.

```
<SOAP-ENV:Envelope xnlns:SOAP-                                  / SOAP envelope
ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body>                                               / SOAP body to carry RequestUtilization
    <RequestUtilization>
      <acl:aclMessages>                                         / List of ACL parameters
        <kqmlPerformative>
          <performativeName>ask-one</performativeName>
            <parameters>
              <sender>MasterAgent</sender>
              <receiver>SwitchAgentSB</receiver>
              <in-reply-to>msg</in-reply-to>
              <reply-with>RequestUtilization</reply-with>
              <language>SOAP</language>                         / Using SOAP to transport XML-based ACL
              <ontology>network_device</ontology>                / messages

              <content>                                         / xmlCIM content language
                <RequestUtilization>                            / for describing network information which
                  <INSTANCE CLASSNAME="SwitchSB" >              / contains the following input parameters :
                    <PROPERTY NAME="Utilization">               / "SwitchSB" instance and
                    </PROPERTY>                                 / "Utilization" property.
                  </INSTANCE>
                </RequestUtilization>
              </content>

            </parameters>
        </kqmlPerformative>
      </acl:aclMessages>
    </RequestUtilization>
  </SOAP-ENV:Body>                                              / End of SOAP body and
</SOAP-ENV:Envelope>                                            / SOAP envelope
```

Figure 11: An example of agent message.

6. CONCLUSIONS AND FURTHER WORK

Proper network management requires automatic and efficient mechanisms to aid in the design, deployment and operations of networks. This proposed network management framework employs XDD, an XML-based knowledge representation language which uses XML as its bare syntax and enhances XML expressive power by provision of mechanisms for succinct and uniform expression of knowledge, rules, conditional relationships, integrity constraints and axioms. Moreover, for the sake of interoperation with other existing network management protocols and standards, the Common Information Model (CIM) is employed. The components in the framework can be adapted and specialized to serve efficiently in several network management functions including configuration, fault, performance, security and accounting management.

As the representation and manipulation of all the information and knowledge in the proposed framework are based on XML, the framework is in line with the current trend of Web technology and hence can easily interoperate with other techniques and systems on the Internet. Therefore, it can readily be extended to deal with distributed network management in which communication, information interchange and remote procedure calls are carried out in terms of XML messaging. Such an extension has been illustrated.

It is also possible to employ the framework to handle policy-based network management where organizational policy, goals and service level agreements are encoded in XML. All these high-level objectives will then be broken down and transformed into low-level or device-level technical targets, leading to the development of a unified XML-based framework for the management of networks and services. This possibility is being explored.

REFERENCES

[1] William Stallings, *SNMP, SNMP2v2 and RMON: Practical Network Managem*ent, Addison-Wesley, 1999.

[2] D.L. Tennenhouse and D.J. Wetherall, Towards an Active Network Architecture, Computer Communication Review, 1996.

[3] D.L. Tennenhouse, J.M. Smith, W.D. Sincoskie, D.J. Wetherall, and G.J.Minden, A Survey of Active Network Research, IEEE Communications Magazine, Vol.35, No.1, pp.80-86, January 1997.

[4] Amit B. Kulkarni and Gary J.Minden, Active Networking Services for Wired/Wireless Networks, IEEE Special Issue on Active Networks, July 1998.

[5] Y. Yemini, A. V. Konstantinou, and D. Florissi, NESTOR: An Architecture for Network Self-Management and Organization, IEEE Journal on Selected Areas in Communications, Vol.18, No.5, pp.758-766, May 2000.

[6] Raouf Boutaba, Salima Omari, and Ajay Pal Singh Virk, SELFCON: An Architecture for Self-Configuration of Networks, Journal Of Communications And Networks, Vol.3, No.4, December 2001.

[7] Martin-Flatin J. P., Znaty S., Hubaux J.P., A Survey of Distributed Enterprise Network and Systems Management Paradigms, Journal of Network and Systems Management, Vol.7, No.1, 1999.

[8] Raouf Boutaba and Jin Xiao, Network Management: State of the Art, IFIP World Computer Congress, 2002.

[9] *Common Information Model (CIM) Version 2.7 Specification*, Distributed Management Task Force, April 2003.

[10] V. Wuwongse, C. Anutariya, K. Akama and E. Nantajeewarawat, XML Declarative Description (XDD). A Language for the Semantic Web, IEEE Intelligent Systems, 16, 3:54-65, May/June 2001.

[11] Heinz-Gerd Hegaring, Sebastian Abeck, and Bernhard Neumair, *Integrated Management of Networked Systems, Concepts, Architectures, and Their Operational Application*, Morgan Kaufmann Publishers, San Francisco, California,1998.

[12] William Stallings, *Local & Metropolitan Area Networks*, Sixth Edition, Prentice Hall, 2000.

[13] J. O. Kephart and D. M. Chess, The Vision of Autonomic Computing, IEEE Computer, 36(1), 2003.

[14] A. W. Jackson, J. P. G. Sterbenz, M. N. Condell, and R. R. Hain, Active Network Monitoring and Control: The SENCOMM Architecture and Implementation, presented at DARPA Active Networks Conference and Expositin (DANCE), California, 2002.

[15] Ricciulli, L., Porras, P., An Adaptable Network Control and Reporting System (ANCORS), Integrated Network Management, 1999, Distributed Management for the Networked Millennium. Proceedings of the Sixth IFIP/IEEE International Symposium, pp. 401 –414, 1999.

[16] A. Konstantinou, Y. Yemini, S. Bhatt, and S. Rajagopalan, Managing Security in Dynamic Networks, In 13th USENIX Systems Administration Conference (LISA'99), Seattle, WA, USA, 1999.

[17] A. Konstantinou, Y. Yemini, and D. Florissi, Towards Self Configuring Networks, In DARPA Active Networks Conference and Exposition (DANCE), IEEE Press, 2002.

[18] *XML CIM by Distributed Management Task Force (DMTF)*, Available from: http://www.dmtf.org/download/spec/xmls/CIM_XML _Mapping20.htm.

[19] *Cisco AVVID Network Infrastructure Enterprise Quality of Service Design*, Solutions Reference Network Design, August, 2002.

[20] R. Braden, L. Zhang, S. Berson, S. Herzog, S. Jamin, Resource ReSerVation Protocol (RSVP) Version 1 Functional Specification [Rfc2205], September 1997.

[21] Gilbert Held, *Virtual LANs Construction, Implementation, and Management*, John Wiley & Sons, Inc. 1997.

[22] Distributed Management Task Force, *Web Based Enterprise Management (WBEM) Initiative*, Available from: http://www.dmtf.org/standards/wbem .

[23] David Lewis and Jens D. Mouritzsen, The Role of XML in TMN Evolution, Proceedings of IM 2001, Seattle, USA, IEEE, 2001.

[24] S. Jindadamrongwech, An Agent Communication Language using XML Declarative Description, Master's Thesis, Computer Science and Information Management Program, Asian Institute of Technology, Thailand, 2000.

[25] *Knowledge Query Manipulation Language*, Available from: http://www.cs.umbc.edu/kqml/ .

APPENDIX

XML Declarative Description (XDD) [10] is an XML-based knowledge representation, which extends ordinary well-formed XML elements by incorporation of variables for an enhancement of expressive power and representation of implicit information into so called *XML expressions*. Ordinary XML elements – XML expressions without variable – are called *ground XML expressions*. Every component of an XML expression can contain variables, e.g., its expression or a sequence of sub-expressions (*E-variables*), tag names or attribute names (*N-variables*), strings or literal contents (*S-variables*), pairs of attributes and values (*P-variables*) and some partial structures (*I-variables*). Every variable is prefixed by '$*T*:', where *T* denotes its type; for example, $S:value and $E:expression are *S*- and *E*-variables, which can be specialized into a string or a sequence of XML expressions, respectively.

An *XDD description* is a set of *XML clauses* of the form:

$$H \leftarrow B_1, \ldots, B_m, \beta_1, \ldots, \beta_n,$$

where $m, n \geq 0$, H and the B_i are XML expressions, and each of the β_i is a predefined *XML constraint* – useful for defining a restriction on XML expressions or their components. The XML expression H is called the *head*, the set $\{ B_1, \ldots, B_m, \beta_1, \ldots , \beta_n\}$ the *body* of the clause. When the body is empty, such a clause is referred to as an *XML unit clause* and the symbol '$\leftarrow$' will often be omitted; hence, an XML element or document can be mapped directly onto *a ground XML unit clause*. Given an XDD description D, its meaning is the set of all XML elements which are directly described by and are derivable from the unit and non-unit clauses in D, respectively.