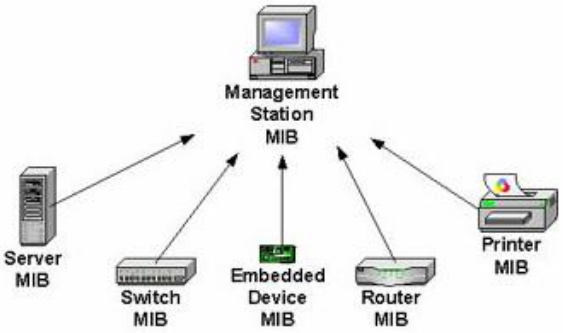
MOC

MIB

TRAP

GET

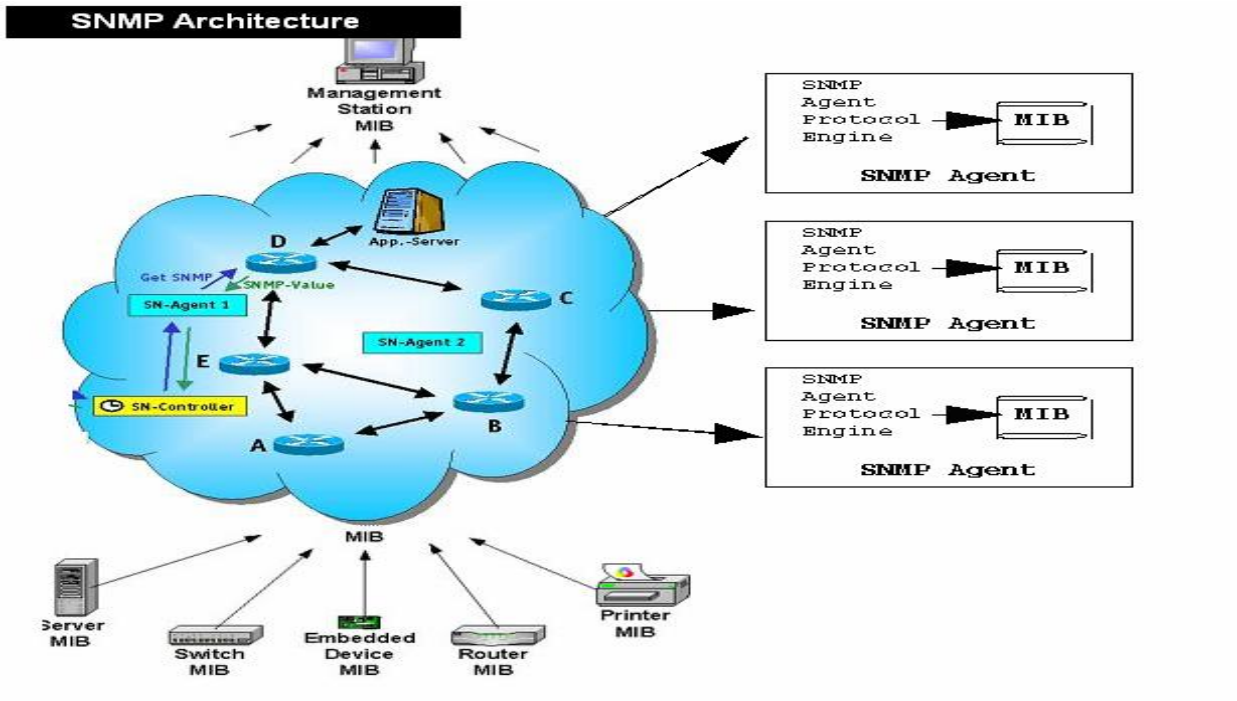
SNMP



NAME : Kifayat Ullah
REGISTRATION NO : 029807695
TITLE : Applications of SNMP
ASSIGNMENT NO : 2
MODULE CODE : COMM3D
MODULE TITLE : Network Managment

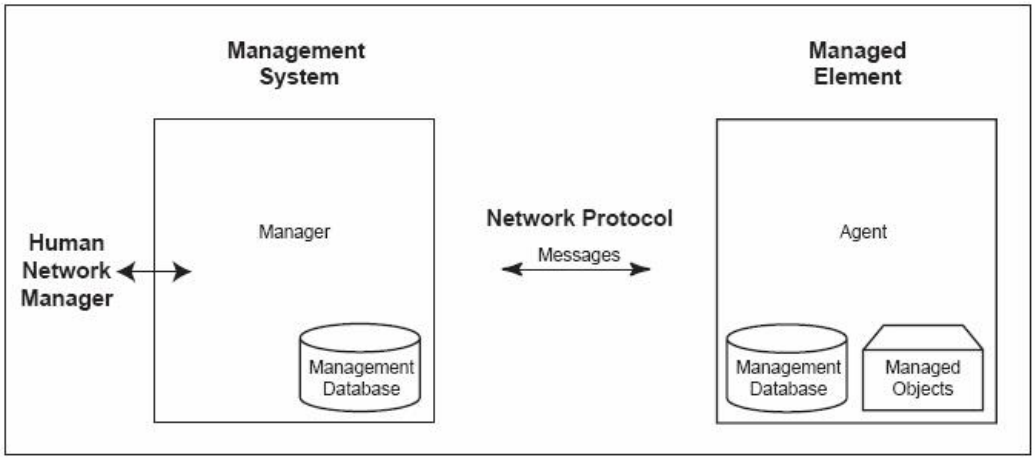
Abstract:
The assignment will elaborately explain the basic application of the SNMP standards for network management. As the networks of today's world are increasing in size and operation the need for proper network management is more crucial. SNMP was first introduced as a short term solution for network management in the late 80's but its easy structure and good management capabilities made it an industrial standard mechanism for network management. SNMP is based on an agent/manager model with database of those managed object in combination with some network protocols. We are given some scenarios with routers, switches, bridges and connecting different LANs, with some additional topics such as web management techniques and other related issues. Some description of MOC trees and IPv6 implementation has been discussed as will to explain the basic concepts of SNMP.

Introduction
SNMP (Simple Network Management Protocol) is an open source industrial standard protocol implementation for network management. SNMP is of the Agent/Manager architecture. The Manager is usually connected to a human network manger console and performs most of the administrative tasks. The agents are the objects that are required to be managed by the Manager. The information between the agent and the manager are sending in a controlled manners using UDP transport level protocol. These managed objects (Agents) are bridges, hubs, routers, and other network servers. The manager can be uses to manage agent's configurations, parameters, performance statistics and lots more. All these information are stored in logical information database called Management Information Base (MIB).



Simple Network Management Protocol provides simple methods of interacting with network devices. This open standard was defined by IETF RFC 1157 in May 1990.

Basic Elements:
There are few major elements of SNMP operation environment, which are explained below.



Agent:
Agent is a device that runs a demon service listening for SNMP manager's Get, GetNext, or Set messages. The agent provides a large number of *Object Identifiers* (OIDs) which are unique key-value pair to represent object in the network. These OIDs are populated by the agent and are made available to the Managers. The Managers queries these OIDs to get details about some attributes of the agent. The manager can read or write OIDs depending on the message type (Get or Set). SNMP also integrates some authentication features for setting OIDs of objects to make network more secure. SNMP communicates on UDP transport layer protocol using access ports 161 and 162.

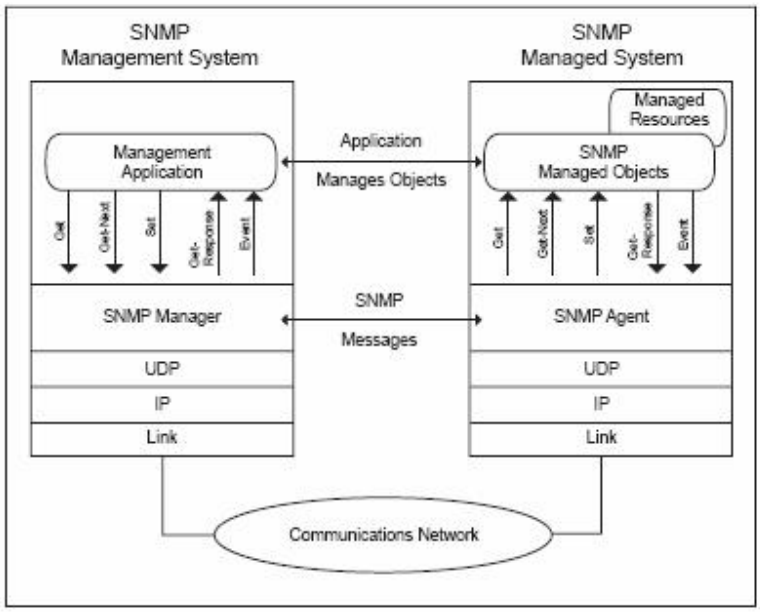
Manager:
Manager is a major administration unit of the SNMP environment. Usually Manager is connected to the human network administrator's console. All the SNMP information is gathered from the Agents via ***traps*** or ***GetResponse*** methods.

MIB:
Management Information Base (MIB) is used by both the manager and agent for the exchange of information with a relatively small set of commands for fast access. The organization of MIB is a structural tree with individual variables (attributes) such as OID (object Identifier), status, description etc. The numeric OID is used for the unique representation of variables in the MIB and SNMP messages. MIB lists the unique OID of each managed element in the network. The manager compiles the MIB files of every managed device in order to monitor and manage that object. MIB is like a database or code book containing information about the devices, such as their capabilities and other unique identifications on the network.

SNMP Messages:

SNMP handles five basic messages for information exchange between Manager and Agent.

1. **Get** (Manager request information from Agent for a specific Variable)
2. **GetNext** (same as Get)
3. **GetResponse** (Agents sends response to manager in reply to Get and GetNext, sends error code if requested information is not found)
4. **Set** (The manager request the agent to change the value of a specific variable)
5. **Trap**(It allows the agent to instantly inform the manger of and important event)

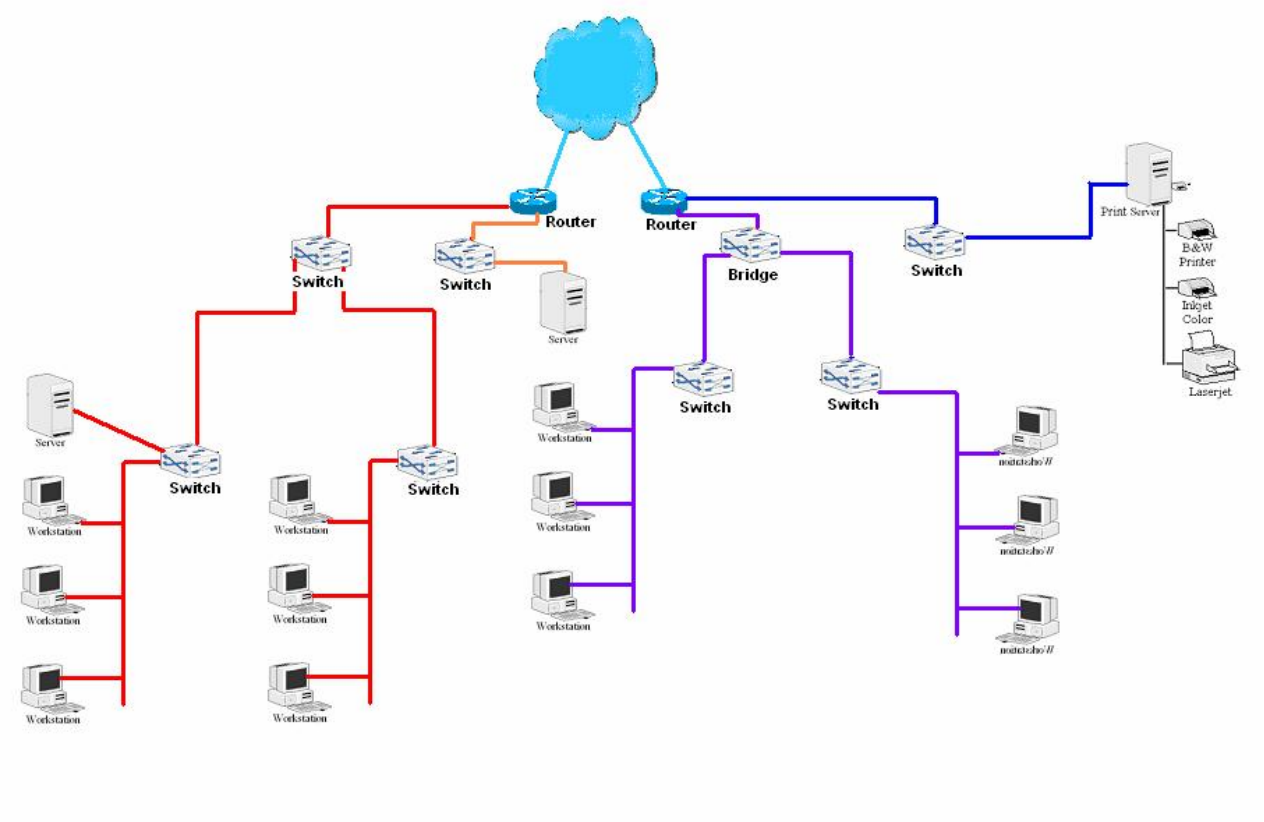


Layers Protocols:

SNMP is using the same layered DOD model TCP/IP architecture for its operation. The SNMP services and demons run on the application layer and use the UDP as a transport protocol on the transport layer. SNMP is using transport layer in order to reduce the communication load on the network.

Application Layer	SNMP	SMTP	Telnet	HTTP	FTP
Transport Layer	UDP		TCP		
Internet Layer	IP			ICMP	
Network Access	PPP		SLIP	ARP	
Physical Layer	Modem		USART	Ethernet	

Network Schematic diagram:
The Diagram given is consist of 4 major nodes, two routers, a switch and a bridge. But can be extended to the following structure. We can have some more different structure for the same network.



Managed Object Classes (MOC):

To draw the MIB of the network diagram first the MOC is needed to point out the specified attributes we are going to managed for the agents. The devices we are going to manage in this network are bellow.

- Router A
- Router B
- Switch A
- Switch B
- Switch AA
- Switch BB
- Switch C
- Bridge R
- Switch RBA
- Switch RBB
- Servers

Router MOC

Attribute	Description
DeviceID	**It used as a Unique name of the router device.**
TotalPacketsSend	**Total Number of Packet Send by the router through a specific interface.**
TotalPacketsReceived	**It is the total number of packets received through an interface.**
UpTime	**It the amount of time the device is switched on since then.**
DropPackets	The Number of packets drop due to network congestion or queue fill up.

Switch MOC

Attribute	Description
DeviceID	**It used as a Unique name of the switch device.**
TotalPacketsSend	**Total Number of Packet Send by the router through a specific interface.**
TotalPacketsReceived	**It is the total number of packets received through an interface.**
UpTime	**It the amount of time the device is switched on since then.**
DropPackets	**The Number of packets drop due to network congestion or queue fill up.**
DevCapabilities	Device Capabilities list, for example which IOS, VLAN can be made on this switch or not etc.

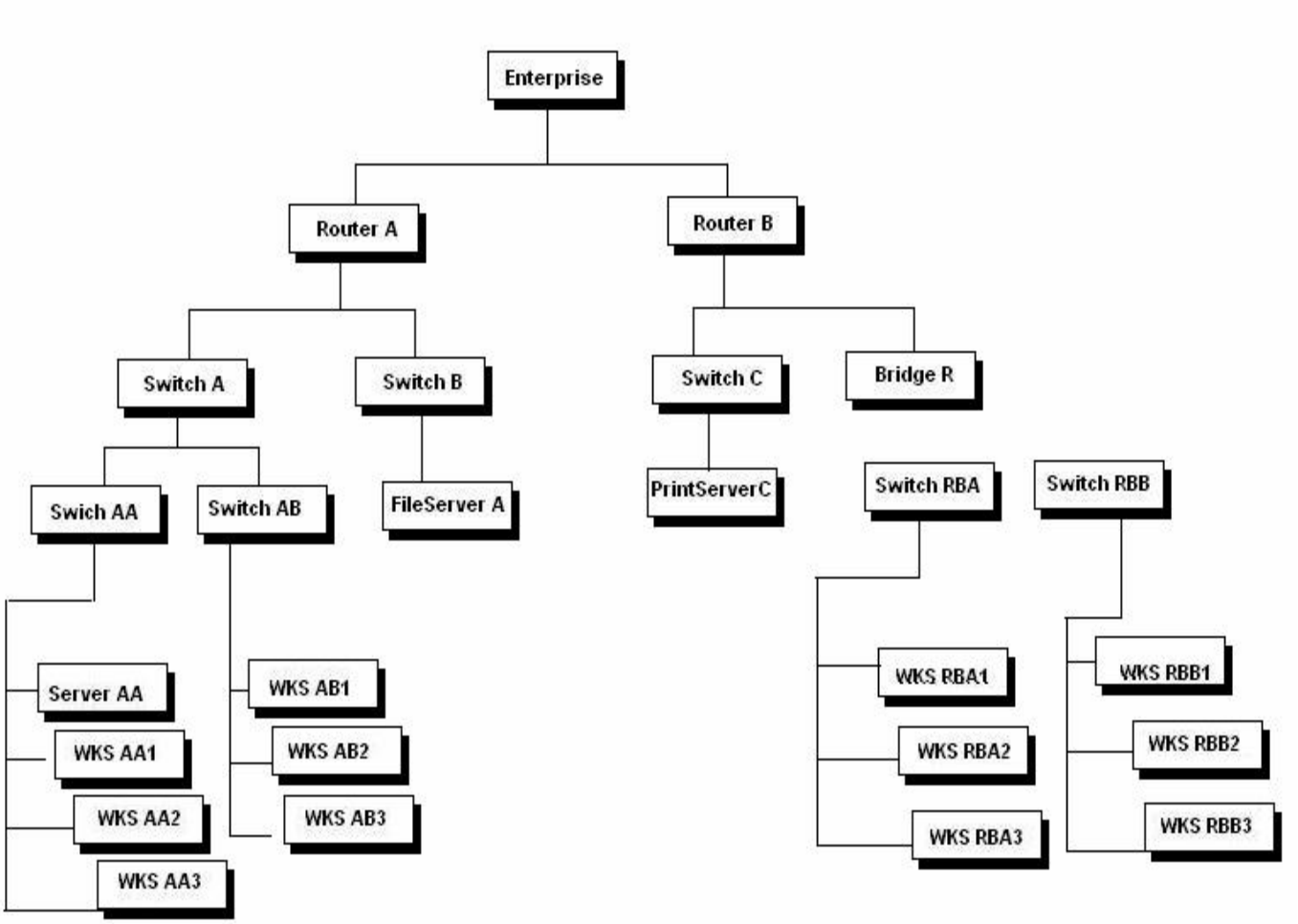
Bridge MOC

Attribute	Description
DeviceID	**It used as a Unique name of the Bridge device.**
TotalPacketsSend	**Total Number of Packet Send by the router through a specific interface.**
TotalPacketsReceived	**It is the total number of packets received through an interface.**
UpTime	**It the amount of time the device is switched on since then.**
DropPackets	The Number of packets drop due to network congestion or queue fill up.

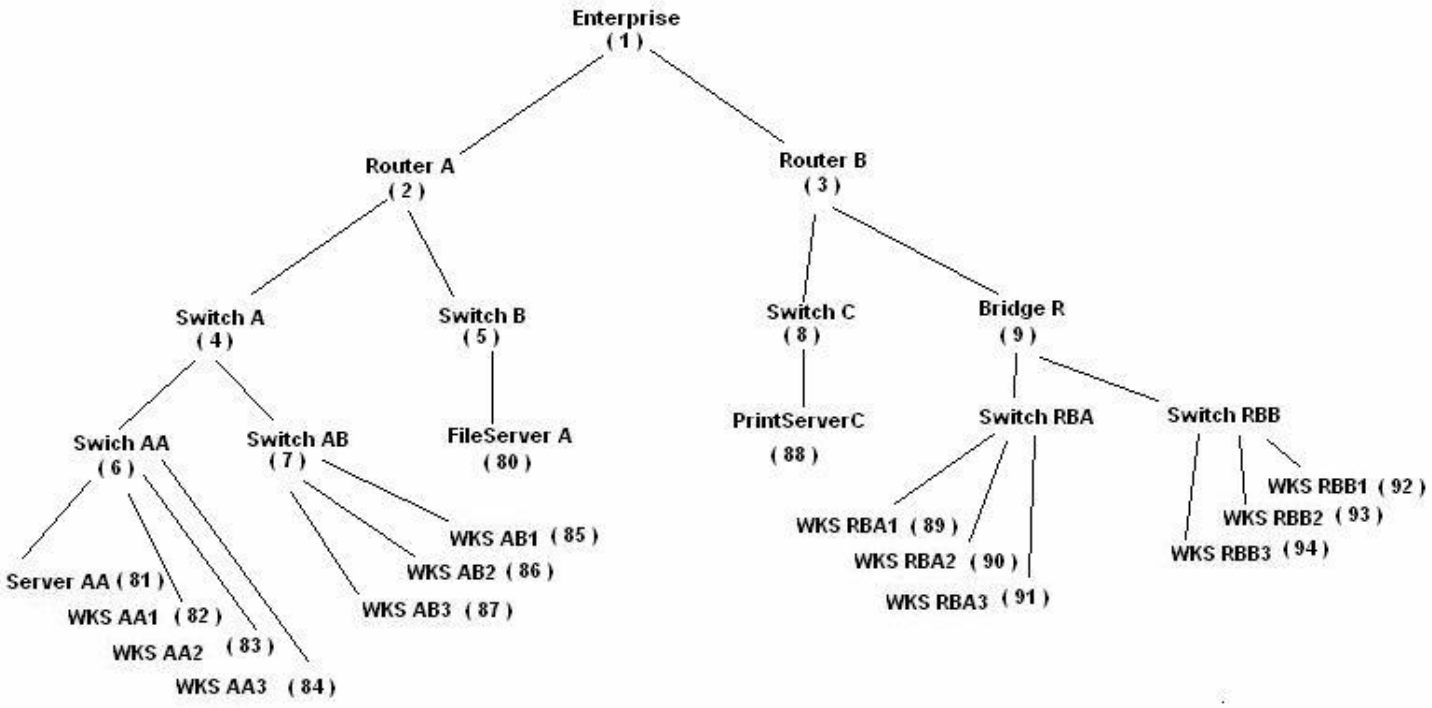
Server MOC

Attribute	Description
ServerId	**It used as a Unique name of the Server Machine**
TotalPacketsSend	**Total Number of Packet Send by the router through a specific interface.**
TotalPacketsReceived	**It is the total number of packets received through an interface.**
UpTime	**It the amount of time the device is switched on since then.**
DropPackets	**The Number of packets drop due to network congestion or queue fill up.**
ServicesList	**Show the list of the Services available on this machine.**
OSLevel	Show which operating system is running on the Server.

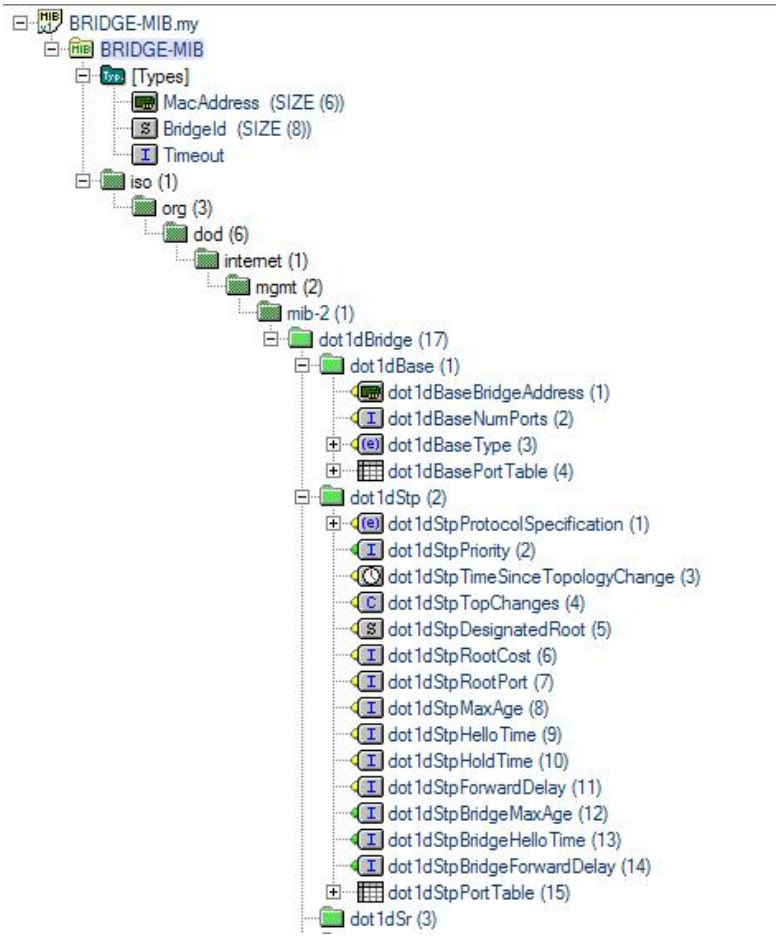
Inheritance Tree:



Naming Tree:
Following is the naming tree of the managed object in our network diagram.



BRIDGE MIB:




```
BRIDGE-MIB DEFINITIONS ::= BEGIN

IMPORTS
            Counter, TimeTicks
                    FROM RFC1155-SMI
            mib-2
                    FROM RFC1213-MIB
            OBJECT-TYPE
                    FROM RFC-1212
            TRAP-TYPE
                    FROM RFC-1215;

MacAddress ::= OCTET STRING (SIZE (6))

BridgeId ::= OCTET STRING (SIZE (8

Timeout ::= INTEGER
   dot1dBridge    OBJECT IDENTIFIER ::= { mib-2 17 }
```

```
dot1dBase      OBJECT IDENTIFIER ::= { dot1dBridge 1 }

dot1dStp       OBJECT IDENTIFIER ::= { dot1dBridge 2 }

dot1dSr        OBJECT IDENTIFIER ::= { dot1dBridge 3 }

dot1dTp        OBJECT IDENTIFIER ::= { dot1dBridge 4 }

dot1dStatic    OBJECT IDENTIFIER ::= { dot1dBridge 5 }


dot1dBaseBridgeAddress OBJECT-TYPE
    SYNTAX  MacAddress
    ACCESS  read-only
    STATUS  mandatory
    ::= { dot1dBase 1 }

dot1dBaseNumPorts OBJECT-TYPE
    SYNTAX  INTEGER
    ACCESS  read-only
    STATUS  mandatory
    ::= { dot1dBase 2 }

dot1dBaseType OBJECT-TYPE
    SYNTAX  INTEGER {
                unknown(1),
                transparent-only(2),
                sourceroute-only(3),
                srt(4)
            }
    ACCESS  read-only
    STATUS  mandatory
    ::= { dot1dBase 3 }

dot1dBasePortTable OBJECT-TYPE
    SYNTAX  SEQUENCE OF Dot1dBasePortEntry
    ACCESS  not-accessible
    STATUS  mandatory
    ::= { dot1dBase 4 }

dot1dBasePortEntry OBJECT-TYPE
    SYNTAX  Dot1dBasePortEntry
    ACCESS  not-accessible
    STATUS  mandatory
    INDEX   { dot1dBasePort }
    ::= { dot1dBasePortTable 1 }

Dot1dBasePortEntry ::=
    SEQUENCE {
        dot1dBasePort
            INTEGER,
        dot1dBasePortIfIndex
            INTEGER,
        dot1dBasePortCircuit
            OBJECT IDENTIFIER,
        dot1dBasePortDelayExceededDiscards
            Counter,
        dot1dBasePortMtuExceededDiscards
            Counter
```

```
    }

dot1dBasePort OBJECT-TYPE
    SYNTAX  INTEGER (1..65535)
    ACCESS  read-only
    STATUS  mandatory
    ::= { dot1dBasePortEntry 1 }

dot1dBasePortIfIndex OBJECT-TYPE
    SYNTAX  INTEGER
    ACCESS  read-only
    STATUS  mandatory
    ::= { dot1dBasePortEntry 2 }

dot1dBasePortCircuit OBJECT-TYPE
    SYNTAX  OBJECT IDENTIFIER
    ACCESS  read-only
    STATUS  mandatory
    ::= { dot1dBasePortEntry 3 }

dot1dBasePortDelayExceededDiscards OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    ::= { dot1dBasePortEntry 4 }

dot1dBasePortMtuExceededDiscards OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    ::= { dot1dBasePortEntry 5 }

dot1dStpProtocolSpecification OBJECT-TYPE
    SYNTAX  INTEGER {
                unknown(1),
                decLb100(2),
                ieee8021d(3)
            }
    ACCESS  read-only
    STATUS  mandatory
    ::= { dot1dStp 1 }

dot1dStpPriority OBJECT-TYPE
    SYNTAX  INTEGER (0..65535)
    ACCESS  read-write
    STATUS  mandatory
    ::= { dot1dStp 2 }

dot1dStpTimeSinceTopologyChange OBJECT-TYPE
    SYNTAX  TimeTicks
    ACCESS  read-only
    STATUS  mandatory
    ::= { dot1dStp 3 }

dot1dStpTopChanges OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    ::= { dot1dStp 4 }

dot1dStpDesignatedRoot OBJECT-TYPE
    SYNTAX  BridgeId
    ACCESS  read-only
```

```
    STATUS  mandatory
    ::= { dot1dStp 5 }

dot1dStpRootCost OBJECT-TYPE
    SYNTAX  INTEGER
    ACCESS  read-only
    STATUS  mandatory
    ::= { dot1dStp 6 }

dot1dStpRootPort OBJECT-TYPE
    SYNTAX  INTEGER
    ACCESS  read-only
    STATUS  mandatory
    ::= { dot1dStp 7 }

dot1dStpMaxAge OBJECT-TYPE
    SYNTAX  Timeout
    ACCESS  read-only
    STATUS  mandatory
    ::= { dot1dStp 8 }

dot1dStpHelloTime OBJECT-TYPE
    SYNTAX  Timeout
    ACCESS  read-only
    STATUS  mandatory
    ::= { dot1dStp 9 }

dot1dStpHoldTime OBJECT-TYPE
    SYNTAX  INTEGER
    ACCESS  read-only
    STATUS  mandatory
    ::= { dot1dStp 10 }

dot1dStpForwardDelay OBJECT-TYPE
    SYNTAX  Timeout
    ACCESS  read-only
    STATUS  mandatory
    ::= { dot1dStp 11 }

dot1dStpBridgeMaxAge OBJECT-TYPE
    SYNTAX  Timeout (600..4000)
    ACCESS  read-write
    STATUS  mandatory
    ::= { dot1dStp 12 }

dot1dStpBridgeHelloTime OBJECT-TYPE
    SYNTAX  Timeout (100..1000)
    ACCESS  read-write
    STATUS  mandatory
    ::= { dot1dStp 13 }

dot1dStpBridgeForwardDelay OBJECT-TYPE
    SYNTAX  Timeout (400..3000)
    ACCESS  read-write
    STATUS  mandatory
    ::= { dot1dStp 14 }

dot1dStpPortTable OBJECT-TYPE
    SYNTAX  SEQUENCE OF Dot1dStpPortEntry
    ACCESS  not-accessible
    STATUS  mandatory
    ::= { dot1dStp 15 }
```

```
dot1dStpPortEntry OBJECT-TYPE
    SYNTAX  Dot1dStpPortEntry
    ACCESS  not-accessible
    STATUS  mandatory
    INDEX   { dot1dStpPort }
    ::= { dot1dStpPortTable 1 }

Dot1dStpPortEntry ::=
    SEQUENCE {
        dot1dStpPort
            INTEGER,
        dot1dStpPortPriority
            INTEGER,
        dot1dStpPortState
            INTEGER,
        dot1dStpPortEnable
            INTEGER,
        dot1dStpPortPathCost
            INTEGER,
        dot1dStpPortDesignatedRoot
            BridgeId,
        dot1dStpPortDesignatedCost
            INTEGER,
        dot1dStpPortDesignatedBridge
            BridgeId,
        dot1dStpPortDesignatedPort
            OCTET STRING,
        dot1dStpPortForwardTransitions
            Counter
    }

dot1dStpPort OBJECT-TYPE
    SYNTAX  INTEGER (1..65535)
    ACCESS  read-only
    STATUS  mandatory
     ::= { dot1dStpPortEntry 1 }

dot1dStpPortPriority OBJECT-TYPE
    SYNTAX  INTEGER (0..255)
    ACCESS  read-write
    STATUS  mandatory
    ::= { dot1dStpPortEntry 2 }

dot1dStpPortState OBJECT-TYPE
    SYNTAX  INTEGER {
                disabled(1),
                blocking(2),
                listening(3),
                learning(4),
                forwarding(5),
                broken(6)
            }
    ACCESS  read-only
    STATUS  mandatory
    ::= { dot1dStpPortEntry 3 }

dot1dStpPortEnable OBJECT-TYPE
    SYNTAX  INTEGER {
                enabled(1),
                disabled(2)
            }
    ACCESS  read-write
    STATUS  mandatory
```

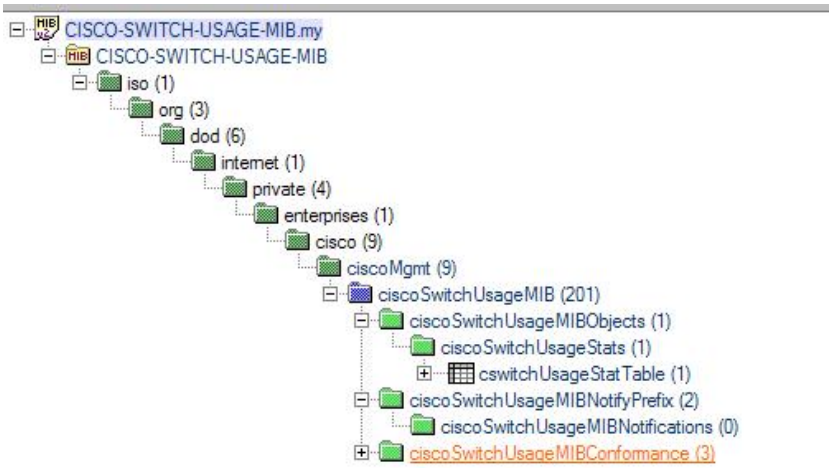
```
    ::= { dot1dStpPortEntry 4 }

dot1dStpPortPathCost OBJECT-TYPE
    SYNTAX  INTEGER (1..65535)
    ACCESS  read-write
    STATUS  mandatory
    ::= { dot1dStpPortEntry 5 }

dot1dStpPortDesignatedRoot OBJECT-TYPE
    SYNTAX  BridgeId
    ACCESS  read-only
    STATUS  mandatory
    ::= { dot1dStpPortEntry 6 }

dot1dStpPortDesignatedCost OBJECT-TYPE
    SYNTAX  INTEGER
    ACCESS  read-only
    STATUS  mandatory
    ::= { dot1dStpPortEntry 7 }

dot1dStpPortDesignatedBridge OBJECT-TYPE
    SYNTAX  BridgeId
    ACCESS  read-only
    STATUS  mandatory
    ::= { dot1dStpPortEntry 8 }

dot1dStpPortDesignatedPort OBJECT-TYPE
    SYNTAX  OCTET STRING (SIZE (2))
    ACCESS  read-only
    STATUS  mandatory
    ::= { dot1dStpPortEntry 9 }

dot1dStpPortForwardTransitions OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    ::= { dot1dStpPortEntry 10 }

dot1dTpLearnedEntryDiscards OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    ::= { dot1dTp 1 }

dot1dTpAgingTime OBJECT-TYPE
    SYNTAX   INTEGER (10..1000000)
    ACCESS   read-write
    STATUS   mandatory
    ::= { dot1dTp 2 }



dot1dTpFdbTable OBJECT-TYPE
    SYNTAX  SEQUENCE OF Dot1dTpFdbEntry
    ACCESS  not-accessible
    STATUS  mandatory
    ::= { dot1dTp 3 }

dot1dTpFdbEntry OBJECT-TYPE
    SYNTAX  Dot1dTpFdbEntry
    ACCESS  not-accessible
    STATUS  mandatory
    INDEX   { dot1dTpFdbAddress }
```

```
    ::= { dot1dTpFdbTable 1 }

Dot1dTpFdbEntry ::=
    SEQUENCE {
        dot1dTpFdbAddress
            MacAddress,
        dot1dTpFdbPort
            INTEGER,
        dot1dTpFdbStatus
            INTEGER
    }

dot1dTpFdbAddress OBJECT-TYPE
    SYNTAX  MacAddress
    ACCESS  read-only
    STATUS  mandatory
    ::= { dot1dTpFdbEntry 1 }

dot1dTpFdbPort OBJECT-TYPE
    SYNTAX  INTEGER
    ACCESS  read-only
    STATUS  mandatory
    ::= { dot1dTpFdbEntry 2 }

dot1dTpFdbStatus OBJECT-TYPE
    SYNTAX  INTEGER {
                other(1),
                invalid(2),
                learned(3),
                self(4),
                mgmt(5)
            }
    ACCESS  read-only
    STATUS  mandatory
    ::= { dot1dTpFdbEntry 3 }


dot1dTpPortTable OBJECT-TYPE
    SYNTAX  SEQUENCE OF Dot1dTpPortEntry
    ACCESS  not-accessible
    STATUS  mandatory
    ::= { dot1dTp 4 }

dot1dTpPortEntry OBJECT-TYPE
    SYNTAX  Dot1dTpPortEntry
    ACCESS  not-accessible
    STATUS  mandatory
    INDEX   { dot1dTpPort }
    ::= { dot1dTpPortTable 1 }

Dot1dTpPortEntry ::=
    SEQUENCE {
        dot1dTpPort
            INTEGER,
        dot1dTpPortMaxInfo
            INTEGER,
        dot1dTpPortInFrames
            Counter,
        dot1dTpPortOutFrames
            Counter,
        dot1dTpPortInDiscards
            Counter
    }
```

```
dot1dTpPort OBJECT-TYPE
    SYNTAX  INTEGER (1..65535)
    ACCESS  read-only
    STATUS  mandatory
    ::= { dot1dTpPortEntry 1 }

dot1dTpPortMaxInfo OBJECT-TYPE
    SYNTAX  INTEGER
    ACCESS  read-only
    STATUS  mandatory
    ::= { dot1dTpPortEntry 2 }

dot1dTpPortInFrames OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    ::= { dot1dTpPortEntry 3 }

dot1dTpPortOutFrames OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    ::= { dot1dTpPortEntry 4 }

dot1dTpPortInDiscards OBJECT-TYPE
    SYNTAX   Counter
    ACCESS   read-only
    STATUS   mandatory
    ::= { dot1dTpPortEntry 5 }


dot1dStaticTable OBJECT-TYPE
    SYNTAX  SEQUENCE OF Dot1dStaticEntry
    ACCESS  not-accessible
    STATUS  mandatory
    ::= { dot1dStatic 1 }

dot1dStaticEntry OBJECT-TYPE
    SYNTAX  Dot1dStaticEntry
    ACCESS  not-accessible
    STATUS  mandatory
    INDEX   { dot1dStaticAddress, dot1dStaticReceivePort }
    ::= { dot1dStaticTable 1 }

Dot1dStaticEntry ::=
    SEQUENCE {
        dot1dStaticAddress
            MacAddress,
        dot1dStaticReceivePort
            INTEGER,
        dot1dStaticAllowedToGoTo
            OCTET STRING,
        dot1dStaticStatus
            INTEGER
    }

dot1dStaticAddress OBJECT-TYPE
    SYNTAX  MacAddress
    ACCESS  read-write
    STATUS  mandatory
    ::= { dot1dStaticEntry 1 }
```

```
        dot1dStaticReceivePort OBJECT-TYPE
            SYNTAX  INTEGER
            ACCESS  read-write
            STATUS  mandatory
            ::= { dot1dStaticEntry 2 }

        dot1dStaticAllowedToGoTo OBJECT-TYPE
            SYNTAX  OCTET STRING
            ACCESS  read-write
            STATUS  mandatory
            ::= { dot1dStaticEntry 3 }

        dot1dStaticStatus OBJECT-TYPE
            SYNTAX  INTEGER {
                        other(1),
                        invalid(2),
                        permanent(3),
                        deleteOnReset(4),
                        deleteOnTimeout(5)
                    }
            ACCESS  read-write
            STATUS  mandatory
            ::= { dot1dStaticEntry 4 }


        newRoot TRAP-TYPE
            ENTERPRISE  dot1dBridge
            ::= 1

        topologyChange TRAP-TYPE
            ENTERPRISE  dot1dBridge
             ::= 2

END
```

SWITCH:



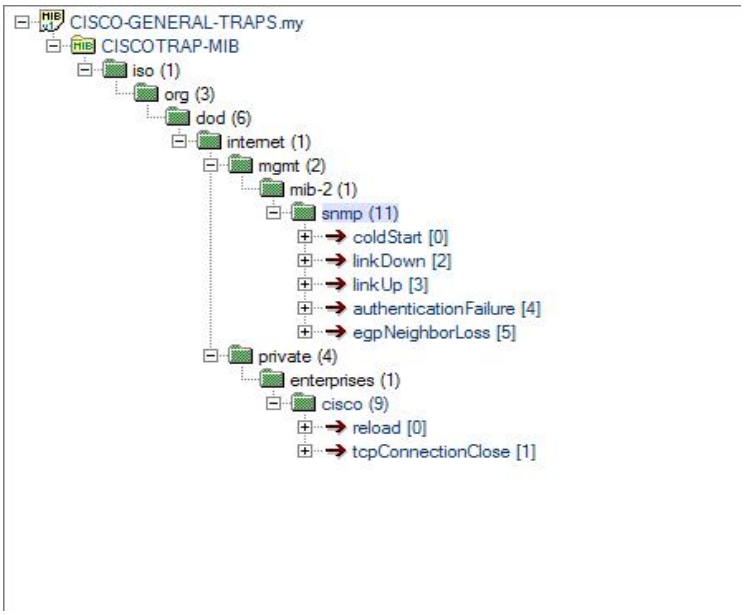

```
CISCO-SWITCH-USAGE-MIB DEFINITIONS ::= BEGIN

IMPORTS
```

```
        MODULE-IDENTITY, Counter32, Counter64,
        OBJECT-TYPE             FROM SNMPv2-SMI
        MODULE-COMPLIANCE,
        OBJECT-GROUP            FROM SNMPv2-CONF
        ifIndex
                                FROM IF-MIB
        ciscoMgmt               FROM CISCO-SMI;


ciscoSwitchUsageMIB   MODULE-IDENTITY
        ::= { ciscoMgmt 201 }


ciscoSwitchUsageMIBObjects OBJECT IDENTIFIER
                        ::= { ciscoSwitchUsageMIB 1 }

ciscoSwitchUsageStats      OBJECT IDENTIFIER
                        ::= { ciscoSwitchUsageMIBObjects 1 }

cswitchUsageStatTable OBJECT-TYPE
        SYNTAX          SEQUENCE OF CswitchUsageStatEntry
        MAX-ACCESS      not-accessible
        STATUS          current
        ::= { ciscoSwitchUsageStats 1 }

cswitchUsageStatEntry OBJECT-TYPE
        SYNTAX          CswitchUsageStatEntry
        MAX-ACCESS      not-accessible
        STATUS          current
        INDEX           { ifIndex }
        ::= { cswitchUsageStatTable 1 }

CswitchUsageStatEntry ::=
        SEQUENCE {
                cswitchUsageByIngrsIntfPkts        Counter32,
                cswitchUsageByIngrsIntfHCPkts      Counter64,
                cswitchUsageByIngrsIntfOctets      Counter32,
                cswitchUsageByIngrsIntfHCOctets    Counter64
        }

cswitchUsageByIngrsIntfPkts OBJECT-TYPE
        SYNTAX          Counter32
        MAX-ACCESS      read-only
        STATUS          current
        ::= { cswitchUsageStatEntry  1 }

cswitchUsageByIngrsIntfHCPkts OBJECT-TYPE
        SYNTAX          Counter64
        MAX-ACCESS      read-only
        STATUS          current
        ::= { cswitchUsageStatEntry  2 }

cswitchUsageByIngrsIntfOctets OBJECT-TYPE
        SYNTAX          Counter32
        MAX-ACCESS      read-only
        STATUS          current
        ::= { cswitchUsageStatEntry  3 }

cswitchUsageByIngrsIntfHCOctets OBJECT-TYPE
        SYNTAX          Counter64
        MAX-ACCESS      read-only
        STATUS          current
        ::= { cswitchUsageStatEntry  4 }
```

```
ciscoSwitchUsageMIBNotifyPrefix  OBJECT IDENTIFIER
                              ::= { ciscoSwitchUsageMIB 2 }
ciscoSwitchUsageMIBNotifications  OBJECT IDENTIFIER
                  ::= { ciscoSwitchUsageMIBNotifyPrefix 0 }


ciscoSwitchUsageMIBConformance    OBJECT IDENTIFIER
                              ::= { ciscoSwitchUsageMIB 3 }
ciscoSwitchUsageMIBCompliances    OBJECT IDENTIFIER
                              ::= { ciscoSwitchUsageMIBConformance 1 }
ciscoSwitchUsageMIBGroups         OBJECT IDENTIFIER
                              ::= { ciscoSwitchUsageMIBConformance 2 }


ciscoSwitchUsageMIBCompliance  MODULE-COMPLIANCE
        STATUS  current
        MODULE  -- this module
                MANDATORY-GROUPS { ciscoSwitchUsageMIBGroup }
        ::= { ciscoSwitchUsageMIBCompliances 1 }


ciscoSwitchUsageMIBGroup    OBJECT-GROUP
        OBJECTS {
                cswitchUsageByIngrsIntfPkts,
                cswitchUsageByIngrsIntfHCPkts,
                cswitchUsageByIngrsIntfOctets,
                cswitchUsageByIngrsIntfHCOctets
        }
        STATUS    current
        ::= { ciscoSwitchUsageMIBGroups 1 }

END
```

ROUTER:

ROUTER MIB DESIGN




```
CISCOTRAP-MIB DEFINITIONS ::= BEGIN
```

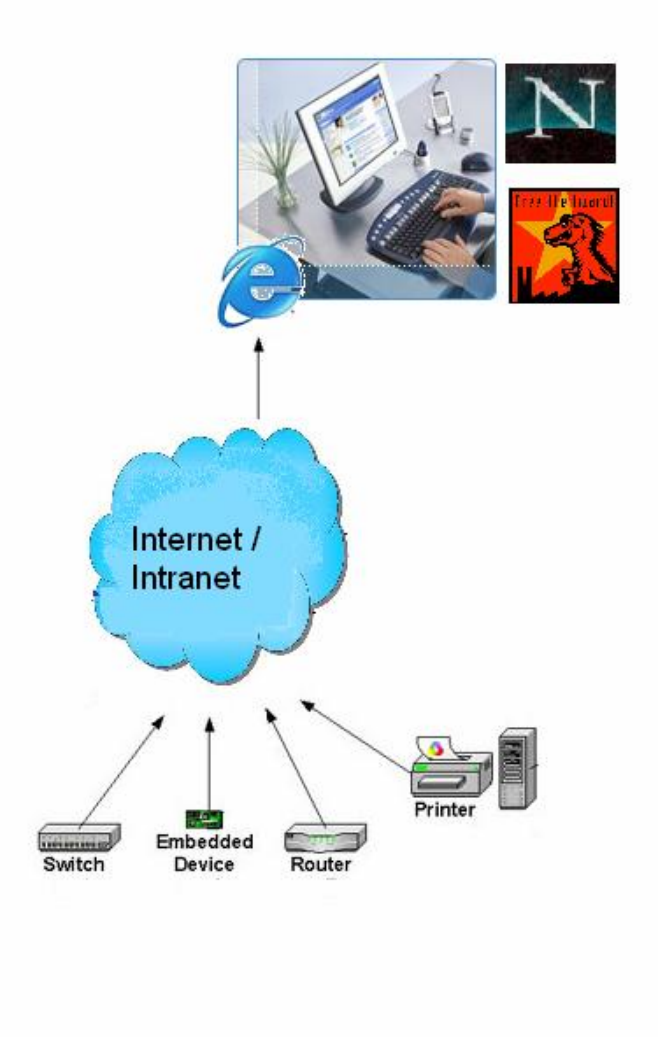
```
IMPORTS
    sysUpTime, ifIndex, ifDescr, ifType, egpNeighAddr,
    tcpConnState
        FROM RFC1213-MIB
    cisco
        FROM CISCO-SMI
    whyReload, authAddr
        FROM OLD-CISCO-SYSTEM-MIB
    locIfReason
        FROM OLD-CISCO-INTERFACES-MIB
    tslineSesType, tsLineUser
        FROM OLD-CISCO-TS-MIB
    loctcpConnElapsed, loctcpConnInBytes, loctcpConnOutBytes
        FROM OLD-CISCO-TCP-MIB
    TRAP-TYPE
        FROM RFC-1215;


coldStart TRAP-TYPE
    ENTERPRISE  snmp
    VARIABLES   { sysUpTime, whyReload }
    ::= 0

linkDown TRAP-TYPE
    ENTERPRISE  snmp
    VARIABLES   { ifIndex, ifDescr, ifType, locIfReason }
    ::= 2

linkUp TRAP-TYPE
    ENTERPRISE  snmp
    VARIABLES   { ifIndex, ifDescr, ifType, locIfReason }
    ::= 3

authenticationFailure TRAP-TYPE
    ENTERPRISE  snmp
    VARIABLES   { authAddr }
    ::= 4

egpNeighborLoss TRAP-TYPE
    ENTERPRISE  snmp
    VARIABLES   { egpNeighAddr }
    ::= 5


reload TRAP-TYPE
    ENTERPRISE  cisco
    VARIABLES   { sysUpTime, whyReload }
    ::= 0

tcpConnectionClose TRAP-TYPE
    ENTERPRISE  cisco
    VARIABLES   { tslineSesType, tcpConnState,
                  loctcpConnElapsed, loctcpConnInBytes,
                  loctcpConnOutBytes, tsLineUser }

    ::= 1

END
```

Web Based Management:
The idea of the web management is to provide such a way the is platform independent and more user friendly. Another good reason for web based management is that as we can see that the almost every operating system has some type of browser and a vital support for HTTP protocol. Web based management is getting more and more popular because its easy to implement and the client is only need to be using a URL to the device we are managing.

WBM merges Web functionalities with network management to provide administrators with capabilities beyond traditional tools. The administrators is using WBM can monitor and control enterprise networks with any Web browser at any node, they are no longer tied down to management workstations and can eliminate many interoperability issues that arise with multiplatform structures.

WBM provides graphical interfaces that present information in a more visual and useful fashion than conventional, command-driven telnet screens. Browser operation and Web page interfaces are known paradigms for today's users of the World Wide Web. As a result, both reduce the costs of training MIS personnel and enable a wider range of users to utilize network status information. In addition, WBM is an ideal means for distributing information about network operation.

Web based management is a new approach which provides new feature with today's networked devices, such as Cisco switch, wireless access points and routers. The device provides a small version of a web server that makes the user able to put a URL in his browser's address bar, and access the device via HTTP protocol.



Web based Management

The web based managed enabled device has its own IP on the network and all we need to do is access that URL from the browser. Web Management is good in other sense as well and that is the vendors do not care about the client operating system and they do not need to provide separate versions of their management application to the client operating system. For example normally a vendor provide a software CD containing software packages that runs only on Windows XP and if the client is using Linux or Unix then the CD is useless to them, but if the vendor provides the web based management system then all the vendor need is a good implementation of standard HTML documents with HTTP protocol. The device runs a mini web server and the client can access only the port 80 of that device for web access. This also increase security of the device as only http access is allowed to the device and no other protocols can gain access to the device.

A Web based management system can have either of the following forms.

- Web-Enabled agents: Agents that have a web server in built. These can be directly managed using a web browser.

- Web-Enabled managers: The managers reside on a web-server. The browsers connect to the manager which in turn queries the agents.

Enhance Security and easy portability:

- An assembly presents evidence at load time.
- An assembly can match more than one code group.
- Assembly evidence is matched against a code group to gain permissions.
- The CLR examines evidence about code to determine if it is trustworthy
- A code group has 2 attributes
  - Membership condition.
  - Permission set.

A few minor drawbacks are

- Hacking like to hack web traffic
- Virus threats are there for most of the web servers
- Proper encryption is required in order to make the communication between the browser and the device more secure.

Most of these devices are provide firmware to provide web based management features. And even the vendor provides the upgrades for those firm ware for free download most of them time as I had an experience with D-Link Wireless Access points.

Desktop Management:

Desktop management provides the basic user interface to interact with most of the devices to manage them. Most of desktop management system support use of mouse drag and drop features, high security techniques implemented in application programs. Desktop managements system also provides a feature like plotting statistical graphs of the network behavior though the same feature is can also be embedded in to web based system.

Most of management resource/information is accessible from the desktop, providing feature for daily, weekly and monthly reports of the network behavior of the network. The entire devices are available for management on one single desktop which help the administrator in easy administration of the network devices. The main difference between

the web based management system and desktop based management system is that the network management function are available only from the web browser in web based management while in the desktop implementation a software tools is used to administrator and manage the device.

The major draw back of the desktop base management system is the version of the software tool to be available for every operating system, such as a software tool that suns on Windows XP to manage D-Link Access point can not be install to run on a FreeBSD, and there fore most of the people prefer to use web based management tools instead of desktop tools, because they do not want them self to be restricted to a specific operating system desktop.

Why IPv6?

IPv6 is the new generation of the Internet Protocol. The addressing scheme of the IPv4 was very good, in the being providing very huge ranges of IPs available host on the large networks. IPv4 is using 32bit address space providing $\mathbf{2^{32}}$ IP address. The class full and classless division helps the routing algorithms to maintain larger IPv4 networks. But the number of networked devices is increasing day by day, and in a very near future IPv4 address stack will be filled, and no more address will be available. There fore the designer of the internet and network specification proposed a new version of addressing scheme on the IP layer called IPv6 which is 128bit addressing scheme. Providing a very huge stack probably will take thousands of years to fill. IPv6 is a very good solution to the address shortage of IPv4 but it is still not widely implemented. And the major reason is the network devices made so far were implemented to support IPv4 only. Most of the vendors were using IPv4 as the primary protocol on network devices such as routers and switches. The Internet backbone and other major backbones of the major network providers around the world had spent million of dollars on there IPv4 infrastructure. And therefore IPv6 was not welcomed very warmly but will take over the internet and network industry in the near coming future. The new network devices and operating systems are providing full support for IPv6.

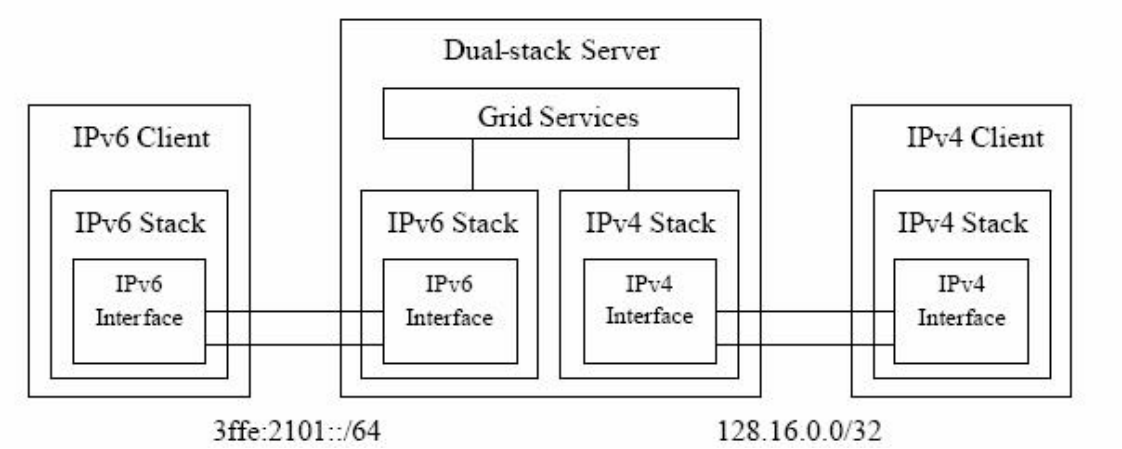
The IPv6 standards were ratified by Internet Engineering Task Force (IETF) in 1998. IPv6 provide support for a number of additional data format features. Most of the protocols are redesigned in the IPv6 suit. IPv6 provide bigger address space, mobility and security of network data.

Bigger Address Space means the range of IPs available to be assigned to devices on the network. As IPv6 is using 128bit address space therefore the range is $\mathbf{2^{128}}$, which is a very huge figure, containing billions of IPs. IPv6 makes a huge scale of grid networking possible. We can also reduce or remove the need of NAT services due to a large number of IPs available for use on the network. IP address can be assigned to almost any device such as PDAs, cell phone, and may be even electric meters, satellite receiver devices in houses etc. we can also assigned more than one IP to a single interface make the use of

multiple IPs more useful. And the network routing and security of end to end devices on the network can very easily be managed.

Mobility Support of the IPv6 makes is a very good choice for Grid computing and a lot of research is done on it in this field. IPv6 is mandated by the 3rd generation Partnership project of mobile networks. Most of the vendors provide IPv6 API to be accessible programmatically for better statistical results and remote management.

Built-in Security is also provided in the IPv6 for increasing scalability of networks and improving performance in homo or heterogeneous networks. As we can see that the Grid Systems must more secure because of its distributed nature, IPv6 provides the basic support to achieve certain level of security. IPSec is implemented into the stack of IPv6 with feature of authentication and encryption of IP traffic. This feature provides the basic security level even if the application itself is a lack security consideration which is a very good feature.
We can also provide a network in which both IPv6 and IPv4 is implemented, but for this some sort of address mapping technique must be using.



Following are some of the challenges faced in IPv6

- How to have both IPv4 and IPv6 traffic.
- Using which address scheme as backbone, IPv4 or IPv6 network.
- Enabling communication IPv6 without replacing IPv4 devices.
- Overhead associated with tunneling, translation or dual stacks.
- Addressing simplicity for routing and mobility.
- Hop limit.

References

[1]. http://wwwmnmteam.informatik.uni-muenchen.de/common/Literatur/MNMPub/Publikationen/kell98a/kell98as.pdf

[2] http://www2.nict.go.jp/is/t822/108/pdf-data/thesis/24-7th-wcits/7th-WC-ITS-05.pdf

[3] http://www.ee.surrey.ac.uk/Personal/G.Pavlou/Publications/Book-chapters/Pavlou-98e.pdf

[4] http://data.goahead.com/sr/RoleOfAvailabilityMiddlewareWP.pdf

[5] http://www2.rad.com/networks/1995/snmp/snmp.htm#

[6] http://www.icir.org/fenner/mibs/mib-index-cisco.html