Administration et sécurité des réseaux

TD SNMP

Exercice 1 :

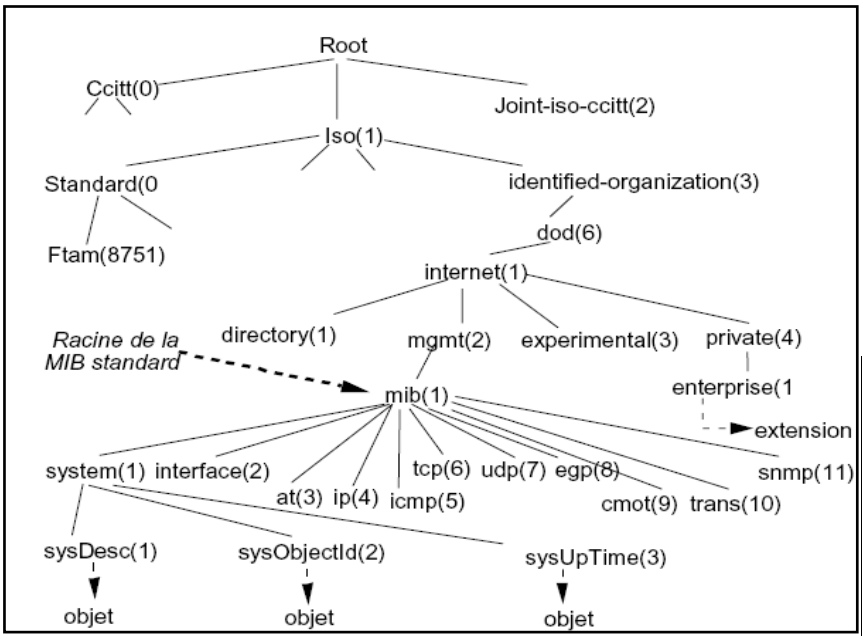
Répondre par vrai ou faux aux assertions suivantes en justifiant votre réponse.

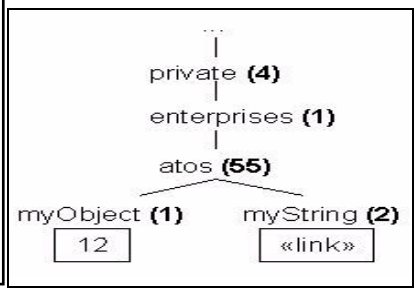
1- Un manager qui a les droits de lecture et d'écriture ne peut pas modifier la valeur de n'importe quel objet de la MIB
2- Tout objet de la MIB est lui associé une valeur.
3- SNMP peut utiliser le même message pour lire la valeur d'un objet et modifier la valeur d'un autre objet puisqu'il permet un accès multiple.
4- La taille d'un message SNMP est fixe
5- Dans SNMPv2, les messages GET, GETNEXT et SET ne sont envoyé que par le manager.
6- La station d'administration communique directement avec les objets administrés.
7- SNMP peut utiliser le même message pour lire plusieurs objets de la MIB.
8- Un hub ne peut pas être administrable car il agit au niveau physique

Exercice 2 :

On considère une partie de la MIB d'un agent SNMP représentée ci dessous.

1- Donner les OID des objets « myString » et « myObject »
2- Le manager désire récupérer la valeur de l'objet « myString » puis modifier la valeur de l'objet « myObject » à 20. Donner, l'échange des messages (entre le manager et l'agent) nécessaires pour réaliser les deux opérations en indiquant les attributs pertinents de chaque message.
3- Peut-on modifier la variable ipInReceives par SNMP?





Exercice 3 :

Partie1 : Soient les deux messages SNMP suivants représentant une requête et sa réponse :

```
⊟ Simple Network Management Protocol
    Version: 1
    Community: public
    PDU type: G .....
    Request Id: 0x25
    Error Status: NO ERROR
    Error Index: 0
    Object identifier 1: 1.3.6.1.4.1.9.2.1.58.0 (SNMPv2-SMI::enterprises.9.2.1.58.0)
    Value: NULL
    Object identifier 2: 1.3.6.1.4.1.9.2.1.57.0 (SNMPv2-SMI::enterprises.9.2.1.57.0)
    Value: NULL
    Object identifier 3: 1.3.6.1.2.1.1.3.0 (SNMPv2-MIB::sysUpTime.0)
    Value: NULL
⊟ Simple Network Management Protocol
    Version: 1
    Community: public
    PDU type: RESPONSE
    Request Id: 0x25
    Error Status: NO ERROR
    Error Index: 0
    Object identifier 1: 1.3.6.1.4.1.9.2.1.58.0 (SNMPv2-SMI::enterprises.9.2.1.58.0)
    Value: INTEGER: 16
    Object identifier 2: 1.3.6.1.4.1.9.2.1.57.0 (SNMPv2-SMI::enterprises.9.2.1.57.0)
    Value: INTEGER: 16
    Object identifier 3: 1.3.6.1.2.1.1.3.0 (SNMPv2-MIB::sysUpTime.0)
    Value: Timeticks: (11915034) 1 day, 9:05:50.34
```

1- Déterminer la version de SNMP utilisé.
2- Préciser le type de la requête (simple ou multiple). Expliquer.
3- Préciser le « PDU type » du premier message (GET ou GET-NEXT ou SET). Expliquer.
4- Quelle est la signification de la valeur 11915034 dans le deuxième message? a-t-elle une relation avec le texte qui le suit ? Expliquer.

Partie2: Soient la requête SNMP suivante:

```
⊟ Simple Network Management Protocol
    Version: 1
    Community: public
    PDU type:
    Request Id: 0x25
    Error Status: NO ERROR
    Error Index: 0
    Object identifier 3: 1.3.6.1.2.1.1.3.0 (SNMPv2-MIB::sysUpTime.0)
    Value: 15246874
```

5- Préciser le type de la requête (simple ou multiple). Expliquer?
6- Préciser le « PDU type » (GET ou GET-NEXT ou SET). Expliquer ?
7- A votre avis, que sera la réponse de l'agent (en vos propres mots) ? Expliquer?

Exercice 4:

1) Un agent au niveau d'un nœud doit remplir et maintenir à jour les objets da sa MIB indépendamment du manager. Expliquer comment cet agent obtient les valeurs des objets gérés afin de les sauvegarder dans sa MIB ?
2) Préciser l'utilité de la requête GET-NEXT par rapport à la requête GET ?
3) Expliquer pourquoi le manager ne peut pas modifier la valeur de n'importe quel objet de la MIB ?
4) En faisant appel aux messages SNMP (GET, SET, GET-NEXT…etc)
   a) Ecrire un algorithme permettant de fixer le TTL des paquets envoyés à 64 ?
   b) Ecrire un algorithme permettant de déterminer si un nœud joue le rôle d'une passerelle ou non ?
   c) Ecrire un algorithme permettant de déterminer le nombre d'interfaces d'un nœud et le débit de chaque interface ?

Exercice 5 :

On considère la SMI suivante d'une MIB propriétaire d'un Switch de marque SMC.

```
smc               OBJECT IDENTIFIER ::= { iso(1) org(3) dod(6) internet(1)
private(4) enterprises(1) 202 }
smcSwitches         OBJECT IDENTIFIER ::= { smc 20 }
smc6724L2    MODULE-IDENTITY
        LAST-UPDATED    "0010300000Z"
ORGANIZATION    "SMC Networks Inc."
        CONTACT-INFO
                "SMC Networks, Inc.
                 Customer Service

                 Postal: No. 1 Creation Rd. III,
                         Science-based Industrial Park,
Hsinchu 300-77, Taiwan, R.O.C.

Tel:  3-5770270

E-mail: support@accton.com.tw"
DESCRIPTION
"The MIB module for SMC6724L2."
REVISION  "0010300000Z"
        DESCRIPTION
"Initial version of this MIB."
::= { smcSwitches 11}

lanModule       OBJECT IDENTIFIER ::= { smc6724L2 1 }
-- wanModule    OBJECT IDENTIFIER ::= { smc6724L2 2 }

vniSystemAdmin  OBJECT IDENTIFIER ::= { lanModule 2 }

vniSystemAdminTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF VniSystemAdminEntry
    ACCESS      not-accessible
    STATUS      mandatory
    DESCRIPTION
        "System Administration Configuration Settings Table"
::= { vniSystemAdmin 1 }

vniSystemAdminEntry OBJECT-TYPE
    SYNTAX      VniSystemAdminEntry
    ACCESS      not-accessible
    STATUS      mandatory
    DESCRIPTION
        "System Administration Configuration Settings Entry"
    INDEX       { vniSystemAdminUserID }
::= { vniSystemAdminTable 1 }

VniSystemAdminEntry ::=
    SEQUENCE {
vniSystemAdminUserID
            INTEGER,
vniSystemAdminPassword
DisplayString,
vniSystemAdminName
DisplayString
    }

vniSystemAdminUserID    OBJECT-TYPE
```

```
    SYNTAX      INTEGER {
admin(1),
guest(2)
                }
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "User ID"
::= { vniSystemAdminEntry 1 }

vniSystemAdminPassword   OBJECT-TYPE
    SYNTAX      DisplayString(SIZE(0..17))
    ACCESS      read-write
    STATUS      mandatory
    DESCRIPTION
        "User password"
::= { vniSystemAdminEntry 2 }

vniSystemAdminName   OBJECT-TYPE
    SYNTAX      DisplayString(SIZE(0..17))
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "User name"
::= { vniSystemAdminEntry 3 }
```

1. Tracer l'arbre d'objets de cette MIB et déterminer l'OID des variables vniSystemAdminUserID, vniSystemAdminPassword, vniSystemAdminName.

2. Quelles sont les requêtes SNMP (avec leurs arguments) qu'on peut appliquer à ces trois variables dans le cas où l'accès à l'Agent du switch est en lecture seule et dans le cas où l'accès est en lecture-écriture.

3. Le switch possède un seul compte administrateur ayant un vniSystemAdminName="admin", un vniSystemAdminPassword="smcadmin". Appliquer les requêtes nécessaires pour accèder à ces informations sachant que le manager ne connait que la structure de cette MIB.

Exercice 6 :

Note : On trouvera en annexe les extraits utiles de la MIB-II ; on a supprimé certaines variables pour simplifier, considérez que seules les variables indiquées existent. Pour les Objet Identifier (OID), donner le nom simple et aussi la forme numérique, complète à partir de la racine.

1) On veut regarder par SNMP si une machine « machine » est un routeur ou non. Quelle variable SNMP consulter ? Donner le nom, l'OID. Indiquer une requête SNMP pour lire cette variable; quels sont les arguments de cette requête et le résultat ?

2) Donner l'appel SNMP (avec ses arguments) modifiant *« machine »* pour qu'elle devienne un routeur.

- On considère pour *« machine »* la table de routes suivante (cas d'une machine d'interface Ethernet d'adresse 210.1.1.10, netmask 255.255.255.0) :

Destination/prefixe	Passerelle	Type	Index interface
195.1.1.0/24	210.1.1.1	Gateway	1
default	210.1.1.3	Gateway	1
210.1.1.0/24	210.1.1.10	Direct	1

3) Quelles sont les variables SNMP associées à la route 195.1.1.0/24 (donner les OID et les valeurs) ?

4) Quel est le nom, et l'OID numérique de la variable donnant la « passerelle » pour la route par défaut ?
5) Donner l'appel SNMP (avec ses arguments) mettant hors service l'interface de la route 195.1.1.0/24
6) On supposant que l'administrateur ne connaît pas le contenu de cette table de routage, donner toutes les requêtes permettant de lire les objets de la MIB relatifs à cette table de routage sous forme *nom_requête(OID, valeur).*

Annexes

```
        RFC1213-MIB DEFINITIONS ::= BEGIN

IMPORTS
mgmt, NetworkAddress, IpAddress, Counter, Gauge,
TimeTicks
                FROM RFC1155-SMI
              OBJECT-TYPE
                   FROM RFC-1212;

--  This MIB module uses the extended OBJECT-TYPE macro as
--  defined in [14];


--  MIB-II (same prefix as MIB-I)

        mib-2      OBJECT IDENTIFIER ::= { iso(1) org(3) dod(6) internet(1) mgmt (2) 1 }

       -- textual conventions

DisplayString ::=
           OCTET STRING
        -- This data type is used to model textual information taken
        -- from the NVT ASCII character set.  By convention, objects
        -- with this syntax are declared as having
        --
        --      SIZE (0..255)

PhysAddress ::=
           OCTET STRING
        -- This data type is used to model media addresses.  For many
        -- types of media, this will be in a binary representation.
        -- For example, an ethernet address would be represented as
        -- a string of 6 octets.

        -- groups in MIB-II

interfaces   OBJECT IDENTIFIER ::= { mib-2 2 }
ip           OBJECT IDENTIFIER ::= { mib-2 4 }

        -- the Interfaces group

        -- Implementation of the Interfaces group is mandatory for
        -- all systems.

ifNumber OBJECT-TYPE
SYNTAX  INTEGER
ACCESS  read-only
STATUS  mandatory
           DESCRIPTION
                "The number of network interfaces (regardless of
their current state) present on this system."
::= { interfaces 1 }

        -- the Interfaces table
```

```
        -- The Interfaces table contains information on the entity's
-- interfaces.  Each interface is thought of as being
        -- attached to a `subnetwork'.  Note that this term should
        -- not be confused with `subnet' which refers to an
        -- addressing partitioning scheme used in the Internet suite
        -- of protocols.

ifTable OBJECT-TYPE
SYNTAX  SEQUENCE OF IfEntry
ACCESS  not-accessible
STATUS  mandatory
            DESCRIPTION
"A list of interface entries.  The number of
entries is given by the value of ifNumber."
::= { interfaces 2 }

ifEntry OBJECT-TYPE
SYNTAX  IfEntry
ACCESS  not-accessible
STATUS  mandatory
            DESCRIPTION
                "An interface entry containing objects at the
subnetwork layer and below for a particular
interface."
            INDEX   { ifIndex }
::= { ifTable 1 }

IfEntry ::=
            SEQUENCE {
ifIndex
                    INTEGER,
ifDescr
DisplayString,
-- NOTE: plusieurs champs ont été supprimés pour simplifier le texte
ifAdminStatus
                    INTEGER,
ifOperStatus
            }

ifIndex OBJECT-TYPE
SYNTAX  INTEGER
ACCESS  read-only
STATUS  mandatory
            DESCRIPTION
"A unique value for each interface.  Its value
ranges between 1 and the value of ifNumber.  The
value for each interface must remain constant at
least from one re-initialization of the entity's
network management system to the next re-
initialization."
::= { ifEntry 1 }

ifDescr OBJECT-TYPE
SYNTAX  DisplayString (SIZE (0..255))
ACCESS  read-only
STATUS  mandatory
            DESCRIPTION
                "A textual string containing information about the
interface.  This string should include the name of
the manufacturer, the product name and the version
of the hardware interface."
::= { ifEntry 2 }

ifAdminStatus OBJECT-TYPE
SYNTAX  INTEGER {
up(1),       -- ready to pass packets
```

```
down(2),
testing(3)   -- in some test mode
                }
ACCESS  read-write
STATUS  mandatory
        DESCRIPTION
"The desired state of the interface.The
testing(3) state indicates that no operational
packets can be passed."
::= { ifEntry 7 }

ifOperStatus OBJECT-TYPE
SYNTAX  INTEGER {
up(1),       -- ready to pass packets
down(2),
testing(3)   -- in some test mode
                }
ACCESS  read-only
STATUS  mandatory
        DESCRIPTION
"The current operational state of the interface.
                The testing(3) state indicates that no operational
packets can be passed."
::= { ifEntry 8 }

        -- the IP group
        -- Implementation of the IP group is mandatory for all
        -- systems.

ipForwarding OBJECT-TYPE
SYNTAX  INTEGER {
forwarding(1),    -- acting as a gateway
                  not-forwarding(2) -- NOT acting as a gateway
                }
ACCESS  read-write
STATUS  mandatory
        DESCRIPTION
                "The indication of whether this entity is acting
as an IP gateway in respect to the forwarding of
datagrams received by, but not addressed to, this
entity.  IP gateways forward datagrams.  IP hosts
do not (except those source-routed via the host).

                Note that for some managed nodes, this object may
take on only a subset of the values possible.
                Accordingly, it is appropriate for an agent to
return a `badValue' response if a management
station attempts to change this object to an
inappropriate value."
::= { ip 1 }

ipDefaultTTL OBJECT-TYPE
SYNTAX  INTEGER
ACCESS  read-write
STATUS  mandatory
        DESCRIPTION
                "The default value inserted into the Time-To-Live
field of the IP header of datagrams originated at
this entity, whenever a TTL value is not supplied
by the transport layer protocol."
::= { ip 2 }

ipInReceives OBJECT-TYPE
SYNTAX  Counter
ACCESS  read-only
STATUS  mandatory
```

DESCRIPTION
"The total number of input datagrams received from interfaces, including those received in error."
::= { ip 3 }
……..

-- the IP routing table

-- The IP routing table contains an entry for each route
-- presently known to this entity.

ipRouteTable OBJECT-TYPE
SYNTAX SEQUENCE OF IpRouteEntry
ACCESS not-accessible
STATUS mandatory
DESCRIPTION
"This entity's IP Routing table."
::= { ip 21 }

ipRouteEntry OBJECT-TYPE
SYNTAX IpRouteEntry
ACCESS not-accessible
STATUS mandatory
DESCRIPTION
"A route to a particular destination."
INDEX { ipRouteDest }
::= { ipRouteTable 1 }

IpRouteEntry ::=
SEQUENCE {
ipRouteDest
IpAddress,
ipRouteIfIndex
INTEGER,
ipRouteNextHop
IpAddress,
ipRouteType
INTEGER,
ipRouteMask
IpAddress,
}

ipRouteDest OBJECT-TYPE
SYNTAX IpAddress
ACCESS read-write
STATUS mandatory
DESCRIPTION
"The destination IP address of this route. An entry with a value of 0.0.0.0 is considered a default route. Multiple routes to a single destination can appear in the table, but access to such multiple entries is dependent on the table-access mechanisms defined by the network management protocol in use."
::= { ipRouteEntry 1 }

ipRouteIfIndex OBJECT-TYPE
SYNTAX INTEGER
ACCESS read-write
STATUS mandatory
DESCRIPTION
"The index value which uniquely identifies the local interface through which the next hop of this route should be reached. The interface identified by a particular value of this index is the same interface as identified by the same value of

ifIndex."
::= { ipRouteEntry 2 }

ipRouteNextHop OBJECT-TYPE
SYNTAX IpAddress
ACCESS read-write
STATUS mandatory
        DESCRIPTION
"The IP address of the next hop of this route.
        (In the case of a route bound to an interface
which is realized via a broadcast media, the value
of this field is the agent's IP address on that
interface.)"
::= { ipRouteEntry 7 }

ipRouteType OBJECT-TYPE
SYNTAX INTEGER {
other(1),   -- none of the following

invalid(2),   -- an invalidated route

                        -- route to directly
direct(3),   -- connected (sub-)network

 -- route to a non-local
indirect(4)   -- host/network/sub-network
        }
ACCESS read-write
STATUS mandatory
        DESCRIPTION
"The type of route. Note that the values
direct(3) and indirect(4) refer to the notion of
direct and indirect routing in the IP
architecture.

        Setting this object to the value invalid(2) has
the effect of invalidating the corresponding entry
in the ipRouteTable object. That is, it
effectivelydissasociates the destination
identified with said entry from the route
identified with said entry. It is an
implementation-specific matter as to whether the
agent removes an invalidated entry from the table.
        Accordingly, management stations must be prepared
to receive tabular information from agents that
corresponds to entries not currently in use.
        Proper interpretation of such entries requires
examination of the relevant ipRouteType object."
::= { ipRouteEntry 8 }

ipRouteMask OBJECT-TYPE
SYNTAX IpAddress
ACCESS read-write
STATUS mandatory
        DESCRIPTION
            "Indicate the mask to be logical-ANDed with the
destination address before being compared to the
value in the ipRouteDest field. For those systems
that do not support arbitrary subnet masks, an
agent constructs the value of the ipRouteMask by
determining whether the value of the correspondent
ipRouteDest field belong to a class-A, B, or C
network, and then using one of:

mask   network
                  255.0.0.0   class-A

255.255.0.0 class-B
255.255.255.0 class-C
If the value of the ipRouteDest is 0.0.0.0 (a default route), then the mask value is also
0.0.0.0. It should be noted that all IP routing subsystems implicitly use this mechanism."
::= { ipRouteEntry 11 }