Travaux dirigés

Exercice 1 — Encodage TLV

Q 1.1 À quoi correspond l'encodage TLV `40 04 C0 A8 FF FE` ?

Correction

On décompose cette suite d'octets :
- `40` ⇒ code du type IpAddress
- `04` ⇒ la longueur de cette adresse est de 4 octets
- `C0 A8 FF FE` ⇒ la valeur de cette adresse (sur 4 octets)

On décode l'adresse IP :
- $\texttt{C0}_{16} = \underbrace{12}_{\texttt{C}} \times 16^1 = 192$
- $\texttt{A8}_{16} = \underbrace{10}_{\texttt{A}} \times 16^1 + 8 = 168$
- $\texttt{FF}_{16} = \underbrace{15}_{\texttt{F}} \times 16^1 + \underbrace{15}_{\texttt{F}} = 255$
- $\texttt{FE}_{16} = \underbrace{15}_{\texttt{F}} \times 16^1 + \underbrace{14}_{\texttt{E}} = 254$

C'est donc l'encodage de l'adresse IP 192.168.255.254. ◆

Q 1.2 Donner l'encodage TLV de la chaîne de caractère `acba`. Les codes ASCII (en décimal) des lettres a, b et c sont 97, 98 et 99.

Correction

Pour passer de décimal en hexadécimal on décompose le nombre en puissances de 16 :
- $\texttt{97} = 6 \times 16^1 + 1 = 61_{16}$
- $\texttt{98} = 6 \times 16^1 + 2 = 62_{16}$
- $\texttt{99} = 6 \times 16^1 + 3 = 63_{16}$ ◆

On obtient donc l'encodage $\underbrace{\texttt{04}}_{\textit{type String}} \ \underbrace{\texttt{04}}_{\textit{sur 4 octets}} \ \underbrace{\texttt{61 63 62 61}}_{\texttt{acba}}$.

Q 1.3 Donner l'encodage TLV de l'OID `1.3.6.1.2.1.4.1`.

Correction

Les deux premiers nombres *x.y* d'un OID sont toujours codés selon la méthode suivante : on calcule $z = 40 \times x + y$ et on code z *sur un seul octet* (voir page 42 du cours). Ici $z = 40 \times 1 + 3 = 43 = 2 \times 16^1 + 1 = 2B_{16}$.
La valeur de l'OID codé est donc : `2B 06 01 02 01 04 01`.
On obtient donc l'encodage `06 07 2B 06 01 02 01 04 01`. ◆

Q 1.4 À quoi correspond l'encodage TLV `30 0C 06 08 53 06 01 02 82 00 01 00 05 00` ?

Correction

On décompose cette suite d'octets :
- `30` ⇒ code du type séquence
- `0C` ⇒ la valeur de cette séquence est codée sur 12 octets
- `06 08 53 06 01 02 82 00 01 00 05 00` ⇒ la valeur (sur 12 octets) de cette séquence

La valeur de cette séquence est elle-même encodée en TLV. Elle contient :
- `06 08 53 06 01 02 82 00 01 00` ⇒ un OID (code `06`) sur `08` octets dont la valeur est `53 06 01 02 82 00 01 00` (décodée plus bas) ;
- et une valeur nulle (code `05`) sur 0 octet.

On décode maintenant l'OID `53 06 01 02 82 00 01 00`.
Le premier octet code les deux premiers nombres *x.y* de l'OID sous la forme $40 \times x + y$ (voir page 42 du cours). $53_{16} = 83 = 40 \times \underbrace{2}_{x} + \underbrace{3}_{y}$. L'OID commence donc par 2.3.
Quand un octet de l'OID est supérieur ou égal à 80 (en hexadécimal) c'est que cet octet et le suivant encodent un seul nombre de l'OID. La règle (voir page 42 du cours) est donc de traduire cet octet et le suivant en binaire, puis de supprimer les bits de poids fort dans les deux octets avant de traduire en décimal. On obtient donc :

$$82\ 00_{16} = \not{1}0000010\ \not{0}0000000_2 = 256$$

L'OID encodé est donc :

$$\underbrace{\texttt{53}}_{2.3} \ \underbrace{\texttt{06}}_{.6} \ \underbrace{\texttt{01}}_{.1} \ \underbrace{\texttt{02}}_{.2} \ \underbrace{\texttt{82 00}}_{.256} \ \underbrace{\texttt{01}}_{.1} \ \underbrace{\texttt{00}}_{.0}$$

En résumé, c'est l'encodage d'une séquence composée de l'OID 2.3.6.1.2.256.1.0 et d'une valeur nulle. ◆

Q 1.5 Donner l'encodage TLV d'une séquence contenant l'adresse IP 10.0.0.1 suivi de l'entier 4096 et d'une sous-séquence contenant la chaîne de caractères "abc" et l'entier 10.

Correction

On encode d'abord les valeurs primitives :
- l'adresse IP 10.0.0.1 ⇒ `40 04 0A 00 00 01` (valeur de type IpAddress sur 4 octets)
- l'entier 4096 ⇒ `02 02 10 00` (valeur de type Integer sur 2 octets)
- la chaîne de caractères "abc" ⇒ `04 03 61 62 63` (valeur de type String sur 3 octets)
- l'entier 10 ⇒ `02 01 0A`

L'encodage de la sous-séquence est donc :

$$\underbrace{30}_{\text{séquence de 8 octets composée de "abc" et de l'entier 10}}\ \underbrace{08}\ \underbrace{04\ 03\ 61\ 62\ 63}\ \underbrace{02\ 01\ 0A}$$

Et la séquence complète est :

$$\underbrace{30}_{\text{séquence de 20 octets composée de l'adresse IP 10.0.0.1, de l'entier 4096 et de la sous-séquence de 8 octets composée de "abc" et de l'entier 10}}\ \underbrace{14}\ \underbrace{40\ 04\ 0A\ 00\ 00\ 01}\ \underbrace{02\ 02\ 10\ 00}\ \underbrace{30}\ \underbrace{08}\ \underbrace{04\ 03\ 61\ 62\ 63}\ \underbrace{02\ 01\ 0A}$$

◆

Exercice 2 — Analyse de trames SNMP

On considère les deux trames ci-dessous capturées avec wireshark.

```
b8 ac 6f 3e 67 dd 00 15 17 ef 57 59 08 00 45 00   ..............E.
00 6f 9f e1 40 00 40 11 fb 1b c0 a8 0f fe c0 a8   ................
0f 18 00 a1 bd bb 00 5b 9f ed 30 51 02 01 01 04   ................
07 70 72 69 76 61 74 65 a2 43 02 04 7f e4 41 e0   .private.C....A.
02 01 11 02 01 02 30 35 30 19 06 08 2b 06 01 02   ......050...+...
01 01 04 00 04 0d 6c 65 67 61 72 73 40 69 63 69   ......legars@ici
2e 66 72 30 18 06 08 2b 06 01 02 01 01 01 00 04   .fr0...+........
0c 6d 6f 6e 2d 70 6f 72 74 61 62 6c 65            .mon-portable
```

```
24 95 04 de c8 90 00 21 cc d3 70 78 08 00 45 00   .......!..px..E.
00 7a 29 15 40 00 40 11 45 8e 0a 00 00 0a 0a 00   .z).@.@.E.......
00 14 d4 f4 00 a2 00 66 cc 47 30 5c 02 01 01 04   .......f.G0\....
06 6d 61 43 6f 6d 6d a7 4f 02 04 54 90 2c 02 02   .maComm.O..T.,..
01 00 02 01 00 30 41 30 0e 06 08 2b 06 01 02 01   .....0A0...+....
01 03 00 43 02 09 29 30 17 06 0a 2b 06 01 06 03   ...C..)0...+....
01 01 04 01 00 06 09 2b 06 01 06 03 01 01 05 01   .......+........
30 16 06 08 2b 06 01 02 01 01 05 00 04 0a 6c 69   0...+.........li
70 6e 73 73 68 2e 66 72                           pnssh.fr
```

Q 2.1 Analyser ces deux trames.

Q 2.2 Donner les commandes snmp-net ou les événements qui ont pu générer ces trames.

Correction

Première trame

En-tête ethernet

- `b8 ac 6f 3e 67 dd` = MAC destination
- `00 15 17 ef 57 59` = MAC source
- `08 00` = etype d'IP

En-tête IP

- `45` = version 4 d'IP + l'en-tête IP fait 5×4 octets (⇒ pas d'options dans l'en-tête IP)
- `11` = code d'UDP (⇒ le paquet IP contient de l'UDP)
- `c0 a8 0f fe` = IP source = 192.168.15.254
- `c0 a8 0f 18` = IP destination = 192.168.15.24

En-tête UDP

— `00 a1` = port source = 161 = port utilisé par les agents SNMP
— `bd bb` = port destination

Message SNMP

Pour pouvoir décoder le contenu du message SNMP il faut consulter la structure (toujours la même) qui est donnée dans les pages 43 à 45 du cours. La page 43 nous dit qu'un message SNMP est toujours codé comme une séquence (au sens TLV) contenant la version, la communauté et le PDU SNMP. La page 44 décrit la structure d'un PDU SNMP : un identifiant de requête, un code d'erreur et ainsi de suite. Si on l'applique sur le message on obtient :

`30 51` un message SNMP (séquence de 81 octets) contenant
  `02 01 01` un numéro de version (l'entier 1 ⇒ SNMPv2c)
  `04 07 70 72 69 76 61 74 65` une communauté (chaîne de 7 octets, `private` codé en ASCII)
  `a2 43` et une réponse SNMP (code `a2`) de 67 octets contenant
    `02 04 7f e4 41 e0` un identifiant de requête (entier sur 4 octets)
    `02 01 11` un code d'erreur (notWritable ⇔ tentative de modification d'un objet en lecture seule, voir page 46 du cours)
    `02 01 02` un index d'erreur (l'entier 2 ⇒ c'est le deuxième objet qui a causé l'erreur)
    `30 35` et une séquence de 43 octets contenant
      `30 19` une séquence de 25 octets contenant
        `06 08 2b 06 01 02 01 01 04 00` l'OID 1.3.6.1.2.1.1.4.0
        `04 0d 6c 65 67 61 72 73 40 69 63 69 2e 66 72` et une chaîne de 13 octets (`legars@ici.fr` codé en ASCII)
      `30 18` et une séquence de 24 octets contenant
        `06 08 2b 06 01 02 01 01 01 00` l'OID 1.3.6.1.2.1.1.1.0
        `04 0c 6d 6f 6e 2d 70 6f 72 74 61 62 6c 65` et une chaîne de 12 octets (`mon-portable` codé en ASCII)

C'est donc une réponse SNMP envoyée suite à une requête SNMPSet qui a tenté de modifier deux objets de la MIB, 1.3.6.1.2.1.1.4.0 et 1.3.6.1.2.1.1.1.0, en leur donnant respectivement les valeurs `legars@ici.fr` et `mon-portable`. Le code d'erreur et l'index d'erreur nous indiquent que le deuxième objet ne peut pas être modifié car il est accessible en lecture seule. La requête a donc échoué et aucun des deux objets n'a été modifié. Les valeurs des objets contenues dans un SNMPSet sont toujours recopiées à l'identique par l'agent dans sa réponse (même si la requête a échoué). De même l'agent recopie dans sa réponse l'identifiant contenu dans la requête (qui est tiré aléatoirement par l'émetteur de la requête).

En résumé cette réponse a été envoyée suite à la requête suivante :

```
$ snmpset -v2c -cprivate 192.168.15.254 1.3.6.1.2.1.1.4.0 s legars@ici.fr
    1.3.6.1.2.1.1.1.0 s mon-portable
```

Deuxième trame

En-tête ethernet

— `24 95 04 de c8 90` = MAC destination
— `00 21 cc d3 70 78` = MAC source
— `08 00` = etype d'IP

En-tête IP

— `45` = version 4 d'IP + l'en-tête IP fait 5×4 octets (⇒ pas d'options dans l'en-tête IP)
— `11` = code d'UDP (⇒ le paquet IP contient de l'UDP)
— `0a 00 00 0a` = IP source = 10.0.0.10
— `0a 00 00 14` = IP destination = 10.0.0.20

En-tête UDP

— `d4 f4` = port source
— `00 a2` = port destination = 162 = port sur lequel sont envoyés les traps/notifications SNMP

Message SNMP

`30 5c` un message SNMP (séquence de 92 octets) contenant
  `02 01 01` un numéro de version (l'entier 1 ⇒ SNMPv2c)
  `04 06 6d 61 43 6f 6d 6d` une communauté (chaîne de 6 octets, `maComm` codé en ASCII)
  `a7 4f` et une notification SNMP (code `a7`) de 79 octets contenant
    `02 04 54 90 2c 02` un identifiant de requête (entier sur 4 octets)
    `02 01 00` un code d'erreur (l'entier 0 ⇒ pas d'erreur)
    `02 01 00` un index d'erreur (0 puisqu'il n'y a pas d'erreur)
    `30 41` et une séquence de 65 octets contenant
      `30 0e` une séquence de 14 octets contenant

`06 08 2b 06 01 02 01 01 03 00` l'OID 1.3.6.1.2.1.1.3.0
`43 02 09 29` un temps (code 43 = timeticks) de 2345 centièmes de secondes
`30 17` une séquence de 23 octets contenant
`06 0a 2b 06 01 06 03 01 01 04 01 00` l'OID 1.3.6.1.6.3.1.1.4.1.0
`06 09 2b 06 01 06 03 01 01 05 01` et l'OID 1.3.6.1.6.3.1.1.5.1
`30 16` et une séquence de 22 octets contenant
`06 08 2b 06 01 02 01 01 05 00` l'OID 1.3.6.1.2.1.1.5.0
`04 0a 6c 69 70 6e 73 73 68 2e 66 72` et une chaîne de 10 octets (`lipnssh.fr` codé en ASCII)

C'est donc une notification SNMP. Les notifications ont toujours un code et un index d'erreur valant 0. Elles contiennent *toujours* :
— en premier objet : 1.3.6.1.2.1.1.3.0 (sysUpTime) qui est le temps (exprimé en $\frac{1}{100}$ de sec.) écoulé depuis que l'agent est en marche ;
— et en deuxième objet : 1.3.6.1.6.3.1.1.4.1.0 (snmpTrapOID) qui contient l'OID de la notification envoyée. Ici cet OID est 1.3.6.1.6.3.1.1.5.1 (coldstart, voir page 26 du cours).

En plus de ces deux objets, une notification peut également contenir d'autres objets. C'est le cas ici puisqu'elle contient également l'objet 1.3.6.1.2.1.1.5.0 (sysName).

En résumé cette notification a été envoyée suite au démarrage (car c'est une notification de type coldstart) de l'agent se trouvant sur l'équipement 10.0.0.10 et nommé `lipnssh.fr`. C'est une notification SNMPv2c utilisant la communauté `maComm`. ◆

Exercice 3 — Série statistique

Soit la suite de valeurs suivante qui correspond à des délais aller-retour en milli-secondes :

$$X = 7, 9, 5, 8, 18, 19, 4$$

Q 3.1 Donnez sa variance.

Correction

La moyenne de la série est $\overline{X} = 10$.
La variance est donc :

$$V(X) = \frac{(7-\overline{X})^2 + (9-\overline{X})^2 + (5-\overline{X})^2 + (8-\overline{X})^2 + (18-\overline{X})^2 + (19-\overline{X})^2 + (4-\overline{X})^2 +}{7} \approx 31.43$$

◆

Q 3.2 Donnez son écart-type.

Correction

$$\sigma(x) = \sqrt{V(x)} \approx 5.61$$

◆

Q 3.3 Donnez sa médiane.

Correction

La série ordonnée est $X' = 4, 5, 7, 8, 9, 18, 19$. La médiane est donc 8 (valeur centrale). ◆