Standard de l'Administration de Réseaux

SNMP

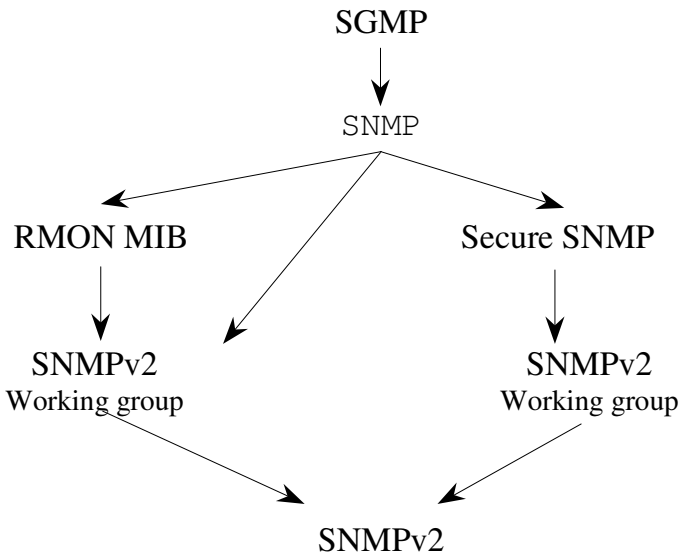
Simple Network Management Protocol

Université François Rabelais de Tours
Faculté des Sciences et Techniques
Antenne Universitaire de Blois

Institut Universitaire Professionnalisé
Informatique et Télécommunication

SNMP v2

Historique de SNMPv2



SGMP : Simple Gateway-Monitoring Protocol
RMON : Remote Network Monitoring
CMOT : CMIP au dessus de TCP/IP

Ce qui change par rapport à SNMP :

- → SNMPv2 est capable de gérer de manière distribuée un réseau: opérations entre stations d'administration
- → sécurité renforcée
- → nouvelles opérations

SMI : Structure de l'information d'administration

Quelques changements mineurs :

- → redéfiniton de certains types.
  Counter devient Counter32 ou Counter64.

- → La clause **ACCESS** devient **MAX-ACCESS.**
  Permet d'indiquer que c'est un niveau maximum d'accès.

- → Quatre possibilités : pas d'accès, lecture seule, lecture-écriture, lecture-création.

- → Introduction de nouveaux mots-clés (**Unit**).

- → La clause **STATUS** n'inclut plus les catégories optionnel et obligatoire.

Les tables

Les droits de création, de destruction et d'accès :

→ Les tables protégées :
Elles ne peuvent être ni crées ni détruites par une station de gestion. Ces tables sont contrôlées par l'agent. Le maximum d'un type d'accès alloué pour cette table et Read-write. Ces tables sont pratiques lorsqu'elles correspondent à un nombre fixe d'attributs comme le nombre d'interfaces physiques par exemple.

→ Les tables non protégées :
Certaines tables peuvent être crées ou détruites. Elles peuvent être initialisées avec un nombre de rangs égal à 0.

snmpORTable OBJECT_TYPE
SYNTAX SEQUENCE OF SnmpOREntry
MAX_ACCESS not-accessible
STATUS current
DESCRIPTION
"the conceptual table listing the dynamically-configurable objet resources in a SNMPv2 entity acting in an agent role. SNMPv2 entities which do not support dynamically-configurable object resources will never have any instances of the columnar objetc in this table"
::= {snmpOR 2}

snmpOREntry OBJECT-TYPE
SYNTAX SnmpOREntry
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
"An entry (conceptual row) in the snmpORTable"
INDEX {snmpORIndex}
::= {snmpORTable 1}

Création et destruction d'un rang dans un tableau

La méthode createAndWait

- → La station de gestion commence à ordonner à l'agent de créer un nouveau rang dans la table avec une instance d'identification ("index value").

- → Si l'agent accepte, il crée le rang et assigne des valeurs par défaut aux objets du rang.

- → Si tous les objets de type read-write possèdent des valeurs par défaut, le rang est placé dans l'état notInservice.

- → S'il existe des objets de type read-write qui n'ont pas des valeurs par défaut, le rang est placé dans l'état notready.

- → Le gestionnaire envoie une requête de type "Get" pour déterminer l'état de chaque objet dont le type d'accès est read-create dans le rang.

- → L'agent envoie chaque valeur de chaque objet. Si l'objet ne possède pas de valeur, il envoie NoSuchInstance.

- → La station d'administration doit alors envoyer un SetRequest pour assigner des valeurs aux objets. Elle peut ensuite envoyer une requête de type "Set" pour activer les objets non actifs.

Création et destruction d'un rang dans un tableau

La méthode createAndGo

→ Plus simple, mais plus restrictive car elle permet de travailler sur des tables dont les objets sont contenus dans une seule PDU.

→ De plus, la station d'administration ne connaît pas les valeurs par défaut attribuées aux différentes colonnes.

→ La station d'administration envoie un Get-PDU pour déterminer les objets de type "read-create" possédant le type noSuchInstance.

→ Elle envoie ensuite un Set-PDU pour créer un nouveau rang et assigner des valeurs aux objets ayant le type d'accès "read-create" dans ce rang.

→ Si le Set réussit, l'agent active ces objets.

Exemple de création de ligne d'une table

La commande "ping" qui fournit un echo distant
Les messages utilisés dans ICMP sont echo et echo_reply

La station d'administration peut mettre à jour un rang pour dire à l'agent de faire un ping sur un autre système à intervalle régulier:

L'agent possède initialement la table :

```
Index IpAddress    Delai Remanient Total  Received  Rtt Status
1     128.2.13.21  1000  0         10     9         3active
```

La station d'administration souhaite ajouter un nouveau rang en utilisant la méthode **createAndWait**.
Elle détermine que le prochain index est 2 et souhaite que le nouveau rang ait les valeurs suivantes :

```
Index IpAddress    Delai Remanient Total Received   Rtt   Status
1     128.2.13.99  1000 20         20     0                active
```

Pour ajouter cette dernière entrée, la station de gestion commence par envoyer une commande Set à l'agent :

```
SetRequest(pingStatus.2=createAndWait)
```

En cas d'acceptation, l'agent répond :

```
Response(pingStatus.2,=notInService)
```

La station de gestion envoie un Get pour lire le nouveau rang :

```
GetRequest(pingIpAdress.2, pingDelay.2, ping.Remaining.2,
pingStatus.2, pingSize.2)
```

L'agent répond :

```
Response ((pingIpAdress.2=noSuchInstance), (pingDelay.2=1000),
(ping.Remaining.2=5), (pingStatus.2=UnderModification),
(pingSize.2=noSuchObject))
```

Certaines valeurs ont été affectées par défaut. Il faut alors compléter....par un

```
SetRequest((pingIpAddress.2=128.2.13.99),....)
```

Le protocole

Quelques modifications

Type PDU	Request-id	0	0	Partie Variable

GetRequest, GetNextRequest, SetRequest, Trap, InformRequest PDU

Type PDU	Request-id	Error-status	Error-index	Partie Variable

Response PDU

Type PDU	Request-id	nonrepeaters	maxrepetitions	Partie Variable

GetBulkRequest

- GetBulkRequest

But → minimiser le nombre d'échange à travers le réseau.

Permet à une station d'administration de solliciter de la part d'un agent une réponse contenant le maximum d'information pouvant être contenu dans un message (limitation par la taille du message).

Possibilité de spécifier des successeurs multiples lexicographiques.

Fonctionnement →

GetBulkRequest inclut une liste de (N+R) variables dans le champ "partie variable".

Pour les N noms, la récupération est faite comme dans GetNextRequest.

Pour chaque variable de la liste, la variable suivante dans l'ordre lexicographique ainsi que sa valeur sont retournées.

Si il n'y a pas de suivant lexicographique, la variable nommée et la valeur "endOfMibView" sont retournées.

Les champs "non-repeaters" et "max-repetition" indiquent le nombre de variables contenu dans la liste "partie variable" et le nombre de successeurs devant être retournés pour les variables restantes.

Exemple

Soient :

L : nombre de variables dans le champ "partie variable".
N : nombre de variables dans le champ "partie variable" avec demande(variable) = un seul successeur.
R : nombre de variables, succédant les N premières variables pour lesquelles de multiples successeurs lexicographiques sont demandées.
M : nombre de successeurs lexicographiques sollicités pour chacune des dernières R variables.

N = MAX (MIN (non-repeaters, L), 0)
M = MAX (max_repetitions, 0)
R = L - N

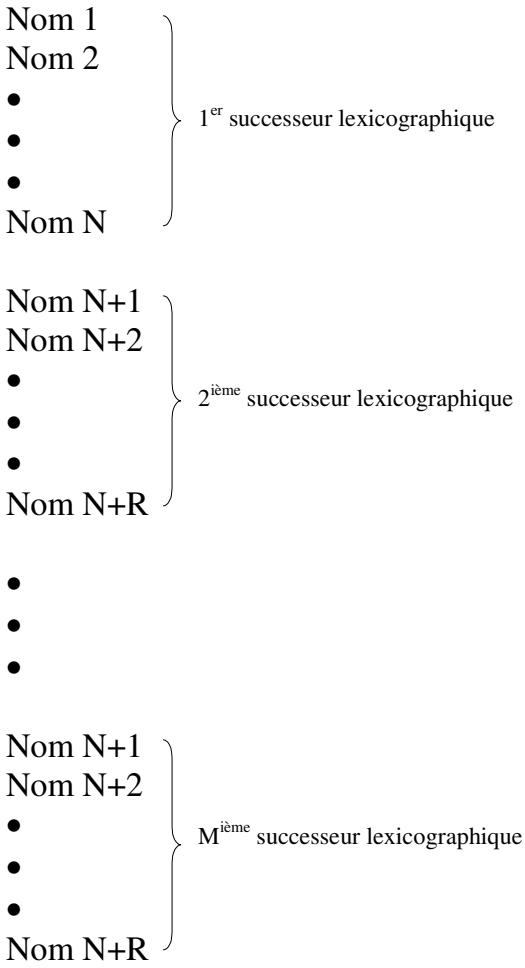
Si N > 0 , alors les N premières variables son traitées comme pour un GetNextRequest.

Si R > 0 et M > 0, alors pour chacun des R dernières variables, ces M successeurs lexicographiques sont renseignés.

Pour chaque variable, cela signifie :

- obtenir la valeur du successeur lexicographique de la variable considérée,
- obtenir la valeur du successeur lexicographique de l'instance objet obtenu à l'étape précédente,
- ainsi de suite, jusqu'à ce que M instances objets soient extraites.

Exemple



Ordonnancement des variables-bindings dans la réponse GetBulkrequest

Soit la table suivante :

Interface-Number	Network-Address	Physical-Address	Type
1	10.0.0.51	00.00.10.01.23.4	static
1	9.2.3.4	00.00.10.54.32.10	dynamic
2	10.0.0.15	00.00.10.98.76.54	dynamic

La station de gestion envoie :

```
GetBulkRequest [non-repeaters=1, max-repetitions=2]
     (sysUpTime, ipNetToMediaPhysAddress, ipNetToMediaType)
```

L'agent répond :

```
Response  ((sysUpTime.0="123456"),
     (ipNetToMediaPhysAddress.1.9.2.3.4="000010543210"),
     (ipNetToMediaType.1.9.2.3.4="dynamic"),
     (ipNetToMediaPhysAddress.1.10. 0.0.51="00001012345"),
     (ipNetToMediaType.1.10.0.0.51="static"))
```

La station de gestion envoie :

```
GetBulkRequest [non-repeaters=1, max-repetitions=2]
     (sysUpTime, ipNetToMediaPhysAddress.1.10.0.0.51,
     ipNetToMediaType.1.10.0.0.51)
```

L'agent répond :

```
Response ((sysUpTime.0="123466"),
     (ipNetToMediaPhysAddress.2.10.0.0.51="0000109887654"),
     (ipNetToMediaType.2.10.0.0.51="dynamic"),
     (ipNetToMediaNetAddress.1 .9.2.3.4="9.2.3.4"),
     (ipRoutingDiscards.0="2"))
```

Possibilité de station d'administration à station d'administration

- InformRequest-PDU

Permet à une station de gestion d'envoyer des informations vers une station d'administration qui centralise des informations contenues dans la MIB "manager-to-manager".

Le message a le même format que Get, Set,...

La MIB permet de spécifier des paramètres tels que :

- → l'intervalle de temps devant séparer 2 "InformRequest_PDU",
- → le nombre d'"InformRequest-PDU" voulues,
- → description de l'événement à rapporter,
- → la date de l'événement,
- → …

Cette PDU étend le mécanisme de Trap de SNMP1.

La MIB

2 nouvelles MIB sont définies :

• SNMPv2 Management Information Base

permet de décrire le comportement des agents SNMP du réseau.
Composée de 5 groupes :

1. SNMPv2 Statistics group : contient des informations relatives au protocole SNMPv2 comme le nombre total de paquets reçus au niveau transport, le nombre de paquets mal codés, le nombre de requêtes PDU GetRequest, GetNextRequest, …

2. SNMPv1 Statistics group : informations relatives au protocole SNMPv1. Par exemple, le nombre de messages ayant un mauvais nom de communauté, nombre de messages demandant une opération non autorisée, …

3. Object resource group : utilisé par l'agent SNMPv2 pour décrire les objets susceptibles d'être configurés par une station d'administration. On y trouve le nom de l'objet, sa description, …

4. Traps group : gère les "traps" générés par un agent,

5. Set group : se compose d'un seul objet qui permet de résoudre 2 problèmes : la sérialisation des opérations de type Set émises par une station de gestion et la gestion de la concurrence d'accès par de multiples stations de gestion.

• Manager-To-Manager MIB

1. Alarm group : permet de spécifier les paramètres de configuration des alarmes (intervalles entre les alarmes, instances ou objet ayant provoqué l'alarme, …),

2. Event group : permet de renseigner une station de gestion sur un ensemble d'événements choisis, sur l'instant où ils se produisent, …

La compatibilité entre SNMP et SNMPv2

La coexistence des 2 versions est facilitée par le fait que SNMPv2 est un sur-ensemble de SNMPv1.

→ La manière la plus simple de gérer le passage de V1 à V2 est de passer la station d'administration à la version 2, qui peut ainsi gérer à la fois des stations en V2 (en cas de gestion répartie) et des agents en V1 et V2.

Il est nécessaire des équivalences dans :

- ⇨ la manière dont sont gérées les informations (SMI)
- ⇨ le protocole

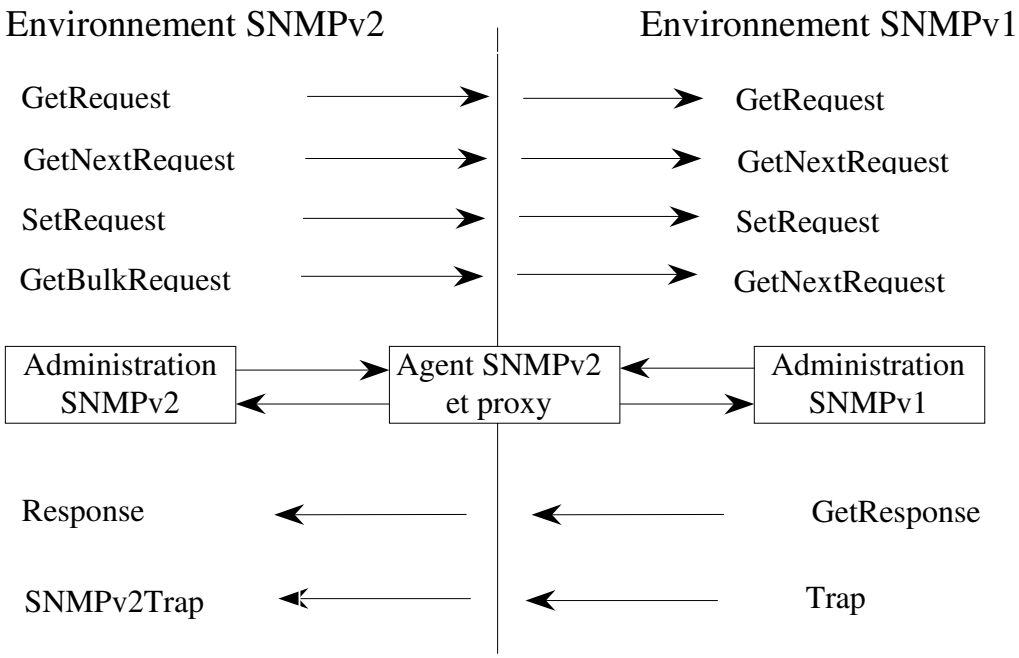
• Le SMI

Pour assurer la compatibilité, les correspondances suivantes sont nécessaires :

- → INTEGER défini sans restriction devient Integer32,
- → Counter devient Counter32,
- → Gauge devient Gauge32,
- → ACCESS devient MAX-ACCESS,
- → …

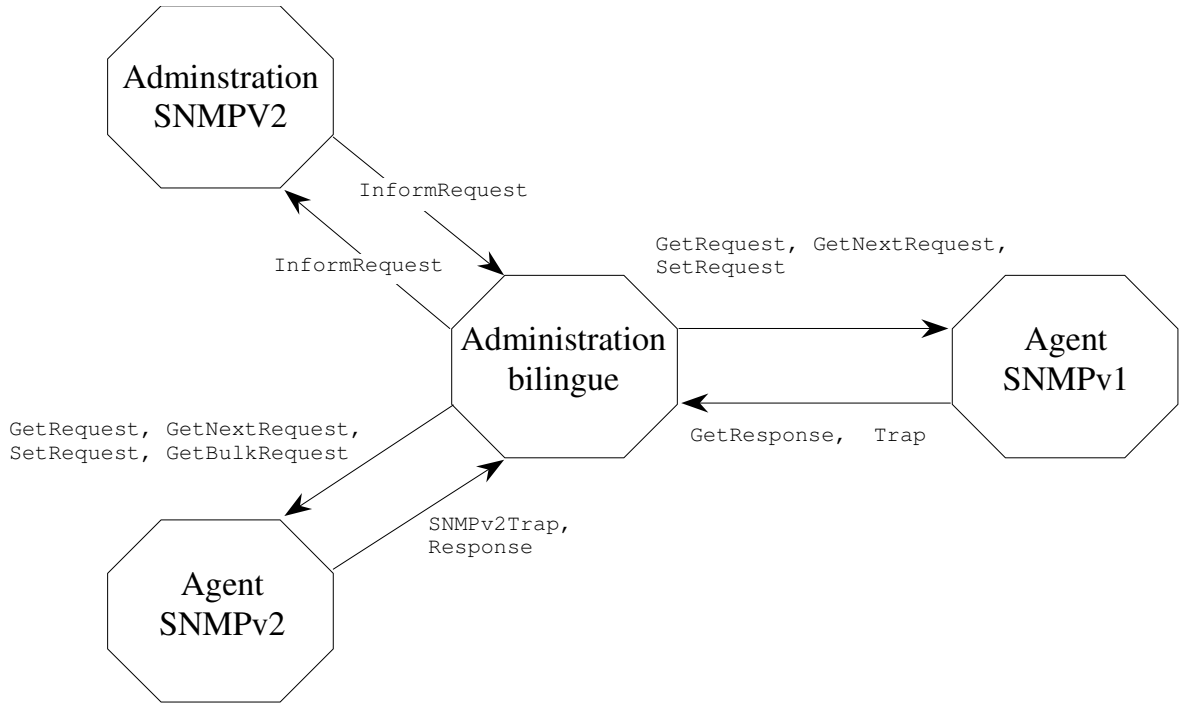
• Le protocole

SNMPv2 gère des PDU supplémentaires.
On prévoit l'utilisation d'un agent proxy qui assure la traduction des PDU entre les 2 versions.



→ noSuchName, readOnly et badValue ne sont pas utilisables par un agent en version 2 mais interprétables par une station d'administration,
→ l'agent proxy assure la gestion des messages ne pouvant pas être contenus dans une seule PDU.

- Possibilité de faire cohabiter 2 versions SNMP. Le gestionnaire utilise au choix le protocole 1 ou 2 :



La sécurité dans SNMP 2

version 1 → utilisation de la notion de communauté pour définir la visibilité accordée à une station par un agent.

version 2 → notion de groupe.

```
SnmpParty ::= SEQUENCE {
  partyIdentify OBJECT IDENTIFIER,      -- identifiant du groupe
  partyDomain OBJECT IDENTIFIER,        -- type de couche transport
  partyAddress OCTET STRING,            -- adresse de niveau transport
  partyMaxMessageSize INTEGER,          -- taille max des messages
  partyAuthProtocol OBJECT IDENTIFIER, -- nomme le protocole d'authentification
                                           utilisé
  partyAuthClock INTEGER,               -- période valide pour le groupe
  partyAuthPrivate OCTET STRING,        -- clé privée d'authentification
  partyAuthPublic OCTET STRING,         -- clé publique d'authentification
  partyAuthLifeTime INTEGER,            -- durée de vie des messages
  partyPrivProtocol OBJECT IDENTIFIER, -- identification du protocole utilisé
                                           (PGP par exemple)
  partyPrivPrivate OCTET STRING,        -- clé privée
  partyPrivPublic OCTET STRING,         -- clé publique
  }
```

Un élément actif sur le réseau agit de la manière suivante :

- exécute uniquement les opérations permises par le groupe,
- maintient une petite base de données qui contient tous les groupes reconnus par l'entité, les opérations pouvant s'effectuer directement et celles qui font appel à un agent de proxy, les ressources accessibles (notion de contexte).

→ Chaque entité maintient donc l'ensemble des données définissant le concept de "politique d'accès".

Le Contexte

Se définit comme étant l'ensemble des ressources accessibles (objets) par une entité SNMPv2.

Il existe deux types de contexte :

- local

Le gestionnaire accède directement aux informations dans l'agent

Le gestionnaire envoie une opération de gestion qui contient :

- un groupe source (srcParty) (le gestionnaire),
- un groupe destination (dstParty) (agent),
- un contexte,
- PDU (Get, Set, …),

L'agent consulte l'entité ACL (Access Control List) et détermine si les opérations sont permises.

- distant

L'agent intervient comme médiateur entre une station d'administration et une entité distante.

L'agent agit comme un proxy qui gère les droits d'accès.

Format des messages sécurisés

privDest	authInfo	dstParty	srcParty	contexte	PDU

Format général

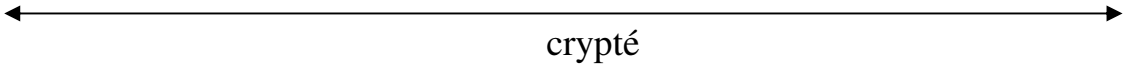
privDest	Octet string	dstParty	srcParty	contexte	PDU

Message non sécurisé

privDest	digest	dsttimestamp	srctimestamp	dstParty	srcParty	contexte	PDU

Authentifié mais non privé

privDest	OctetString	dstParty	srcParty	contexte	PDU



Privé et authentifié

privDst : désigne le groupe pour lequel le message est destiné,
authInfo: protocole d'authentification utilisé.

Émission d'une requête sécurisée

Construction d'un message

srcParty ← groupe d'émission
dstParty ← groupe de réception
contexte ← contexte voulu
PDU ← Get, set,...

La base de données locale de l'entité émettrice est consultée pour récupérer entre autre le type de protocole authentification utilisé

- un message authentifié est construit
  authInfo ← type de protocole

La base de données locale de l'entité émettrice est consultée pour récupérer les caractéristiques du protocole

- un message privé est construit
  privDest ← identifie le destinataire
  message crypté

- transmission au destinataire

Exemples d'agents

Configuration d'un agent non sécurisé

Identity	gracie (agent)	george (manager)
Domain	snmpUDPDomain	snmpUDPDomain
Address	1.2.3.4, 161	1.2.3.5, 2001
AuthProt	noAuth	noAuth
Auth Priv Key	""	""
Auth Pub Key	""	""
Auth clock	0	0
Auth lifetime	0	0
PrivProt	noPriv	noPriv
PrivPrivKey	""	""
PrivPubKey	""	""

Base de données de l'agent

Target	Subject	Context	Privileges
Gracie	george	local	35 (Get,GetNext & GetBulk)
george	gracie	local	132(Response et SNMPv2-Trap)

Configuration d'un agent sécurisé

Identity	ollie (agent)	stan (manager)
Domain	snmpUDPDomainsnmp	UDPDomain
Address	1.2.3.4, 161	1.2.3.5, 2001
Auth Prot	v2md5AuthProtocol	v2md5AuthProtocol
Auth Priv Key	"0123456789AZ"	"GHIJKLM45"
Auth Pub Key	""	""
Auth clock	0	0
Auth lifetime	300	300
PrivProt	desPrivProtocol	desPrivProtocol
PrivPrivKey	"MNOPIU89"	"BNJIUY78"
PrivPubKey	""	""

Base de données de l'agent

Target	Subject	Context	Privileges
ollie	stan	local	35 (Get,GetNext & GetBulk)
stan	ollie	local	132 (Response et SNMPv2-Trap)

Algorithme de synchronisation des horloges

Pour cet algorithme on utilise un nouvel objet :

```
AuthInformation ::= [2] IMPLICIT SEQUENCE {
    AuthDigest OCTET STRING,
    authDstTimestamp UInteger32,
    authSrcTimestamp UInteger32
    }
```

Lorsqu'un message est transmis, il inclut les valeurs d'horloges de l'émetteur et du récepteur.

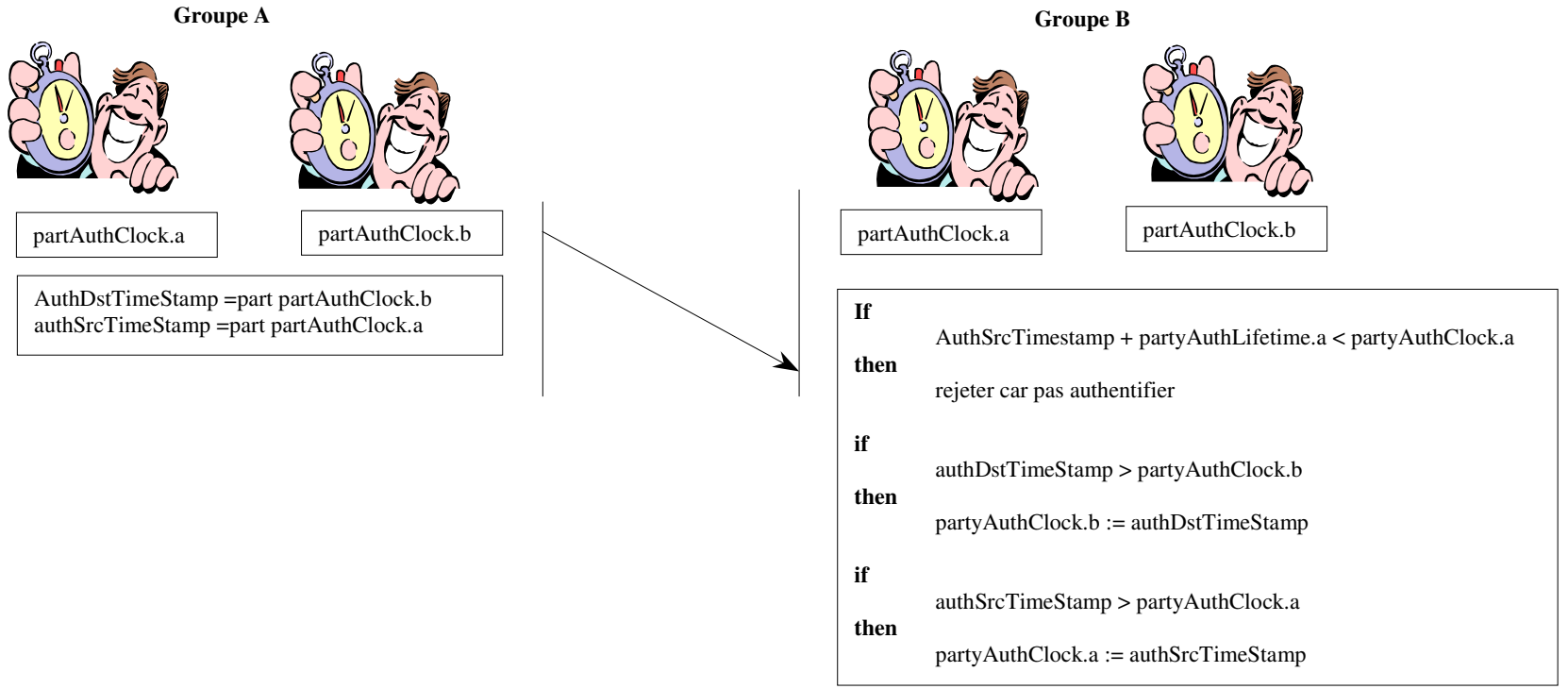
Ces horloges sont synchronisées de telle manière que l'horloge la plus lente soit égale à l'horloge la plus rapide.

Considérons deux groupes : "AgentParty" et "MgrParty" contenant respectivement un agent et une station de gestion.

4 cas de figure sont envisageables :

- → l'estimation de l'horloge "AgentParty" qu'a la station de gestion dépasse la valeur qu'en a l'agent,
- → l'estimation de l'horloge "MgrParty" qu'a la station de gestion dépasse la valeur qu'en a l'agent,
- → l'estimation de l'horloge "AgentParty" qu'a l'agent dépasse la valeur qu'en a la station de gestion,
- → l'estimation de l'horloge "MgrParty" qu'a l'agent dépasse la valeur qu'en a la station de gestion.

Algorithme de synchronisation des horloges(2)



Conclusion

- Snmpv2 : plus efficace, plus sécurisé que SNMPv1 mais pas encore de migrations complètes de tous les sites.

- Approche OSI marginale, non à cause des concepts, mais du fait des investissements déjà réalisés par les administrateurs sur SNMP.

  → recherche d'un protocole compatible avec la v1 de SNMP.

- Quelques inconnues :

  - le devenir du modèle OSI (CMISE/CMIP),
  - la forte évolution des réseaux et l'émergence de nouveaux protocoles,
  - problèmes législatifs concernant le cryptage.

- Trois phénomènes qui devraient aussi influencer l'administration de réseau :
  - la technologie orientée objet,
  - la notion d'agent intelligent (sachant prendre des décisions sans en référer à la station de gestion),
  - l'administration à travers le WEB.

Bibliographie

Les divers RFC sur SNMP accessibles sur le serveur web de l'Urec ou à Pasteur (www.urec.fr, www.pasteur.fr).

SNMPv1
RFCs 1089,1140, 1147, 1155, 1156, 1157, 1158, 1161, 1212, 1213, 1215, 1298.

SNMPv2
RFCs 1441, 1442, 1443, 1444, 1445, 1446, 1447, 1448, 1449, 1450, 1451, 1452.

Les livres

The Simple Book : An Introduction to management of TCP/IP-based Internets by Marshall Rose, Prentice Hall, 2nd edition, 1994

SNMP, SNMPv2, CMIP, The pratical Guide to Network-Management Standards, William Stallings, Addison Wesley, 1995

Rapport de valeur C 94-95 "SNMP", O. Porte, M. Izadpahan

Mémoire d'ingénieur
"Mise en oeuvre du protocole SNMP pour un outil de gestion hétérogène",
G. Ndjeudji, 1992

Les sites qui offrent des logiciels SNMP

lancaster.andrew.cmu.edu:/pub/snmp-dist/*
snmp2.1.2.tar
CMU SNMPv2 source library agent, mid-level agent, net management routines

ftp.ics.uci.edu:mrose/isode-snmpv2/isode-snmpv2.tar.Z
4BSD/ISODE 8.0 SNMPv2 package

dnpap.et.tudelft.nl:/pub/btng
contient RMON agent pour OS/2, SUN OS 4.1.x &Ultrix
Tricklet (perl based SNMP tool pour Unix ou OS/2)
…