

# R502

## Supervision des réseaux

Sami Evangelista  
IUT de Villetaneuse  
Département Réseaux et Télécommunications  
2025–2026

<http://www.lipn.univ-paris13.fr/~evangelista/cours/R502>

Sources des images utilisées dans ce document:

<http://www-lipn.univ-paris13.fr/~evangelista/cours/credits.html>

Ce document est mis à disposition selon les termes de la licence Creative Commons "Attribution – Pas d'utilisation commerciale – Partage dans les mêmes conditions 3.0 non transposé".



## R502 — Supervision des réseaux

<http://www.lipn.univ-paris13.fr/~evangelista/cours/R502>

- ▶ Volume horaire :
  - ▶ 2×3h de cours/TD
  - ▶ 6×3h de TP
  - ▶ 2h de contrôle
- ▶ Évaluation :
  - ▶ TP notés
  - ▶ contrôle

1. Introduction à la supervision des réseaux
2. SNMP : protocole de supervision
3. Centralisation de journaux avec syslog
4. Éléments de qualité de service

*Ensemble des techniques permettant de s'assurer du bon fonctionnement des éléments d'un réseau.*

- ▶ superviser  $\Leftrightarrow$  surveiller
- ▶ tâche dévolue à l'administrateur du réseau
- ▶ éléments d'un réseau :
  - ▶ switches, routeurs
  - ▶ ordinateurs
  - ▶ imprimantes
  - ▶ services (HTTP, FTP, DHCP, ...)
  - ...
  - ▶ En gros, tout équipement qui a une @IP (plus éventuellement un port).

- ▶ Assurer la continuité des services.
  - ▶ Détecter les pannes pour remettre les équipements en service
- ▶ Mesurer les performances du réseau
  - ▶ Assurer une qualité de service
  - ▶ Analyser le trafic (reporting)
- ▶ Détecter/prévenir les attaques
  - ▶ ex : avec un système de détection des intrusions (IDS)

## 1. Surveillance d'un serveur de mail

- ▶ Un processus vérifie périodiquement l'état du disque dur du serveur de mail.
- ▶ espace disque libre  $< 1Go \Rightarrow$  il envoie un mail aux utilisateurs pour leur demander de faire le ménage dans leurs mails

## 2. Surveillance d'une imprimante

- ▶ Toutes les 10 sec. on envoie un ping à une imprimante.
- ▶ 5 pings successifs négatifs  $\Rightarrow$  mail envoyé à l'administrateur

## 3. Blocage d'un attaquant

- ▶ Sur la passerelle d'accès au réseau on journalise les tentatives de connexion SSH.
- ▶ Un processus inspecte périodiquement le journal.
- ▶ Il bloque les @IP (via l'ajout d'une règle de pare-feu) qui apparaissent trop fréquemment (p.ex., plus de 100 tentatives de connexion en une minute).

## 4. Prévion du trafic

- ▶ On enregistre dans une base de données les dates des requêtes envoyées au serveur Web.
- ▶ On peut ensuite en extraire des statistiques pour prévoir les pics d'affluence.

## Supervision par interrogation (polling)

- ▶ à l'initiative du système de supervision
- ▶ Il interroge périodiquement les équipements supervisés sur leur état.
- ▶ ex : envoyer un ping toutes les minutes à l'imprimante

## Supervision par remontée d'incidents

- ▶ à l'initiative des équipements supervisés
- ▶ En cas d'événement exceptionnel, l'équipement supervisé prévient le système de supervision.
- ▶ ex : si un service meurt, le système sur lequel il s'exécutait envoie un message au système de supervision

- ▶ La supervision par polling nécessite plus de bande passante et plus de traitements des deux côtés (superviseur et supervisé).
  - ▶ Elle peut passer à côté d'incidents brefs (p.ex., si un service plante puis redémarre sans que le superviseur n'ait interrogé le supervisé entre-temps).
  - ▶ Avec la méthode par remontée, le superviseur est prévenu dès que possible.
  - ▶ Mais la supervision par polling peut détecter davantage de problèmes (p.ex., problème de réseau (équipement supervisé inaccessible), si la machine supervisée s'est arrêtée brusquement, ...).
- ⇒ intérêt de combiner les deux

## ▶ Outils généralistes

- ▶ ping
- ▶ traceroute
- ▶ iperf
- ▶ netstat
- ▶ wireshark
- ...

## ▶ Protocoles de supervision

- ▶ CMIP (Common Management Information Protocol)
  - ▶ normalisé par l'OSI, peu utilisé
- ▶ SNMP\* (Simple Network Management Protocol)
  - ▶ simple à mettre en œuvre ⇒ très répandu

## ▶ Outils dédiés

- ▶ Nagios\*
- ▶ Centreon
- ▶ Zabbix
- ...

\* : vu dans ce module

<https://www.nagios.com/>

- ▶ logiciel libre
- ▶ première version en 1996
- ▶ 2 composants :
  - ▶ un moteur qui exécute périodiquement des tâches de supervision ;
  - ▶ et une interface web pour visualiser les résultats collectés par le moteur.
- ▶ quelques fonctionnalités importantes :
  - ▶ possibilité de superviser
    - ▶ des équipements (p.ex., vérifier qu'une machine répond au pings)
    - ▶ des services (p.ex., vérifier qu'apache est actif sur un PC)
    - ▶ des ressources (espace disque, charge du processeur, ...)
  - ▶ cartographie du réseau
  - ▶ mise en places d'alerte (envoi de mail/SMS en cas de problème)
- ▶ possibilité de développer des plugins réalisant des tâches de supervision
- ▶ plusieurs forks : icinga, shinken, naemon

Plus de détails en TP.

Vue en temps réel des services disponibles ou à l'arrêt.

**Current Network Status**  
 Last Update: Sun Jul 15 14:09:00 CDT 2001  
 Updated every 75 seconds  
 Nagios™ - [www.nagios.org](http://www.nagios.org)  
 Logged in as guest  
 - Monitoring process is running  
 - Notifications cannot be sent out  
 - Service checks are being executed

**Host Status Totals**

Up	Down	Unreachable	Pending
3	4	0	0
All Problems		All Types	
7		35	

**Service Status Totals**

OK	Warning	Unknown	Critical	Pending
14	2	0	14	19
All Problems		All Types		
16		137		

**Service Details For All Hosts**

Host	Service	Status	Last Check	Duration	Attempt	Service Information
asa01	PN0	Up	07-15-2001 14:04:09	44 4b 7m 13s	5/2	PN0 ok - Packet loss = 0%, RTA = 0.80 ms
asa01	PN0	CRITICAL	07-15-2001 14:04:39	44 3s 46m 13s	5/2	CRITICAL - Plugin timed out after 10 seconds
asa01	Something...	CRITICAL	07-15-2001 14:00:38	44 4b 1m 46s	5/2	(Service Check Timed Out)
asa01	PN0	CRITICAL	07-15-2001 14:02:36	44 4b 1m 46s	5/2	CRITICAL - Plugin timed out after 10 seconds
asa01	PN0	CRITICAL	07-15-2001 14:04:09	44 3s 47m 23s	5/2	CRITICAL - Plugin timed out after 10 seconds
asa01	Something...	CRITICAL	07-15-2001 14:04:39	44 3s 46m 22s	5/2	(Service Check Timed Out)
asa01	PN0	CRITICAL	07-15-2001 14:05:38	44 3s 46m 3s	5/2	CRITICAL - Plugin timed out after 10 seconds
asa01	Something...	CRITICAL	07-15-2001 14:02:36	44 3s 33m 31s	5/2	(Service Check Timed Out)
asa01	PN0	CRITICAL	07-15-2001 14:04:09	44 3s 46m 31s	5/2	CRITICAL - Plugin timed out after 10 seconds
asa01	Something...	CRITICAL	07-15-2001 14:04:39	44 3s 46m 22s	5/2	(Service Check Timed Out)
asa01	PN0	CRITICAL	07-15-2001 14:05:40	44 3s 46m 3s	5/2	CRITICAL - Plugin timed out after 10 seconds
asa01	Something...	CRITICAL	07-15-2001 14:02:36	44 3s 23m 21s	5/2	(Service Check Timed Out)
asa01	Log_Anomalies	PENDING	N/A	44 3s 38m 29s	0/1	Service check is not scheduled for execution...
asa01	TCP_Winapp	PENDING	N/A	44 3s 38m 29s	0/1	Service check is not scheduled for execution...
asa01	Security_Audit	PENDING	N/A	44 3s 38m 29s	0/1	Service check is not scheduled for execution...
asa01	PN0	Up	07-15-2001 14:02:35	44 4b 6m 14s	5/2	PN0 ok - Packet loss = 0%, RTA = 0.80 ms
asa01	PN0	Up	07-15-2001 14:04:01	44 3s 47m 34s	5/2	PN0 ok - Packet loss = 0%, RTA = 0.80 ms
asa01	Security_Audit	PENDING	N/A	44 3s 38m 29s	0/1	Service check is not scheduled for execution...
asa01	TCP_Winapp	PENDING	N/A	44 3s 38m 29s	0/1	Service check is not scheduled for execution...
asa01	Log_Anomalies	PENDING	N/A	44 3s 38m 29s	0/1	Service check is not scheduled for execution...

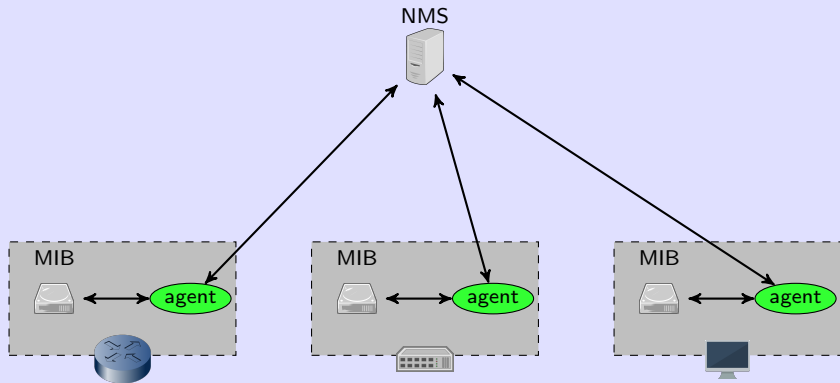
Source : <https://fr.wikipedia.org/wiki/Nagios>, consultée le 10/09/2023.

1. Introduction à la supervision des réseaux
2. SNMP : protocole de supervision
3. Centralisation de journaux avec syslog
4. Éléments de qualité de service

- ▶ SNMP = Simple Network Management Protocol
- ▶ protocole défini par l'IETF
- ▶ 3 versions définies dans différentes RFC :
  - v1 RFC 1065–1067 (1988) — toujours utilisée
  - v2c RFC 1901–1908 (1996) — version la plus répandue  
(Plusieurs v2 ont été définies : v2u, v2p, ... v2c est celle utilisée actuellement.)
  - v3 RFC 3411–3418 (2002)
- ▶ SNMP utilise (généralement) UDP pour le transport.
- ▶ Ports utilisés (par défaut) :
  - ▶ 161 = port utilisé par les agents SNMP
  - ▶ 162 = port de destination des notifications
- ▶ **Dans ce cours, on se basera sur la v2c.**  
(cf. section 2.5 pour une description des différences entre les trois versions)

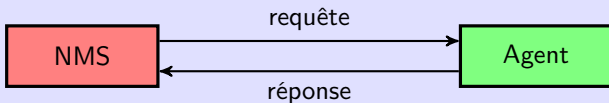
- ▶ SNMP fait intervenir :
  - ▶ un **NMS** (Network Management System) — la plate-forme de supervision ;
  - ▶ et des **agents** — processus s'exécutant sur les équipements supervisés.
- ▶ Chaque agent a en mémoire un ensemble d'objets qui peuvent être lus/modifiés par le NMS.
- ▶ Ces objets donnent des informations sur l'état de l'équipement sur lequel s'exécute l'agent et peuvent être utilisés pour sa supervision.
- ▶ Ils forment une **MIB** : une Management Information Base.
- ▶ Les objets de cette MIB sont décrits dans le langage SMI (cf. les exemples dans la suite).

SMI = Structure of Management Information
- ▶ Exemples d'objets :
  - ▶ états des interfaces réseau (activée/désactivée)
  - ▶ température interne
  - ▶ espace disque libre



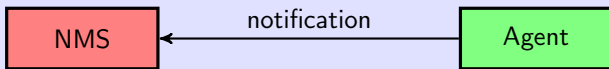
SNMP est mis en œuvre sur la plupart des équipements réseau (routeurs, switches, ...) et systèmes d'exploitation.

## ▶ les requêtes



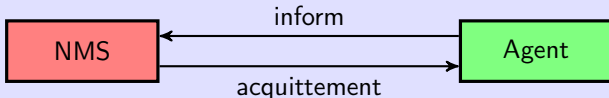
- ▶ permet de lire/modifier la (les) valeur(s) d'un (de plusieurs) objet(s)

## ▶ les notifications (ou traps)



- ▶ permet de prévenir le NMS d'un événement particulier  
Exemple : l'OS va s'arrêter.
- ▶ pas d'acquiescement

## ▶ les informs

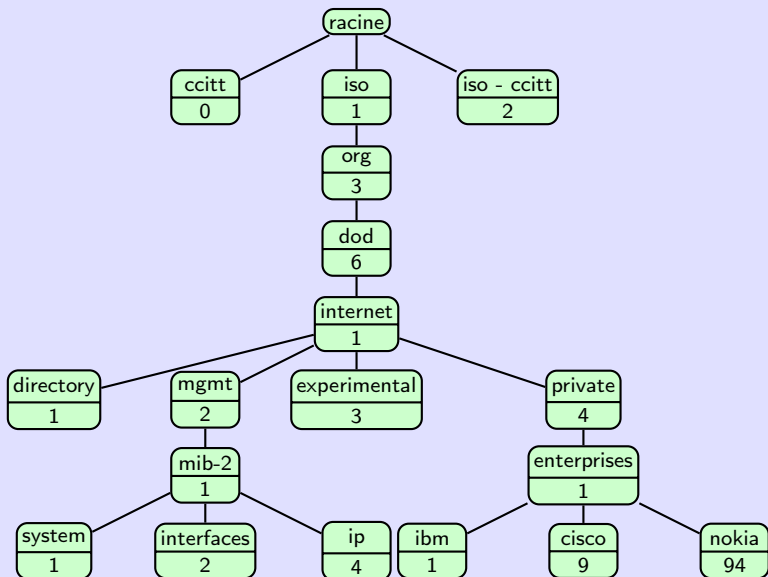


- ▶ notification avec acquiescement

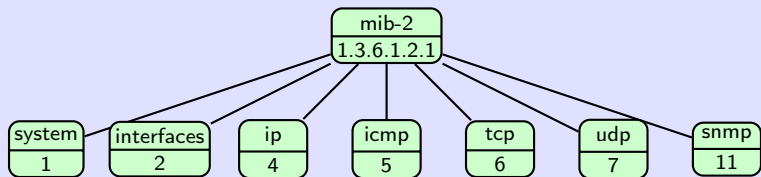
- ▶ **getRequest**  
récupère la valeur d'un (de plusieurs) objet(s)
- ▶ **setRequest**  
modifie la valeur d'un (de plusieurs) objet(s)
- ▶ **getNextRequest**  
récupère la valeur de l'objet suivant dans la MIB
- ▶ **getBulkRequest**  
récupère les valeurs d'un groupe d'objets consécutifs de la MIB

- ▶ Une communauté désigne un ensemble d'agents/NMS auxquels sont associés des droits.
- ▶ communauté  $\approx$  mot de passe
- ▶ Trois types de droits sont associés à une communauté :
  - ▶ readOnly — droit en lecture sur les objets de la MIB
  - ▶ readWrite — droit en lecture/écriture sur les objets de la MIB
  - ▶ trap — droit d'envoyer des traps
- ▶ La communauté est présente dans tous les messages SNMP.
- ▶ L'échange se fait en clair (non chiffré) sur le réseau.
  - ⇒ En voyant passer des messages SNMP on peut découvrir les communautés.
  - ⇒ à changer régulièrement

- ▶ MIB = base de données d'objets présente sur un équipement supervisé par un agent SNMP
- ▶ Les MIB ont une structure arborescente.
- ▶ Cette structure est **normalisée** (en partie) et **extensible** (en partie).
- ▶ Un objet est un nœud de l'arbre.
- ▶ Chaque objet
  - ▶ a un nom symbolique et un numéro ;
  - ▶ et est identifié par un **OID** (object identifier) unique.
- ▶ OID d'un objet O = liste des numéros des objets sur le chemin de la racine à O séparés par des '.'
- ▶ Les agents ne possèdent pas tous la même MIB !

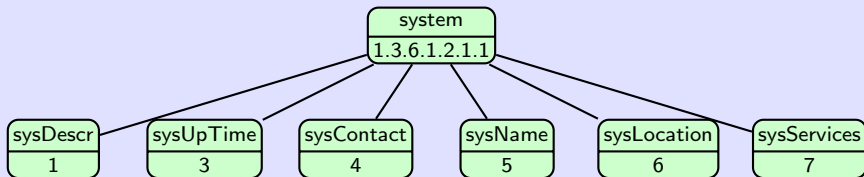


OID = 1.3.6.1.2.1.1

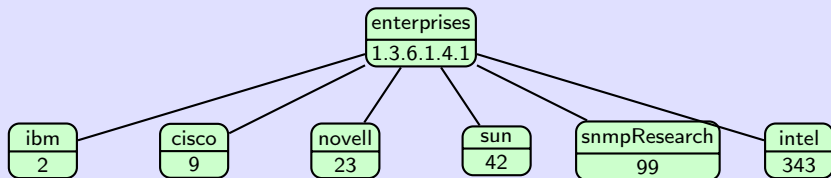


branche **normalisée** et **présente sur tous les agents SNMP** (en théorie!)

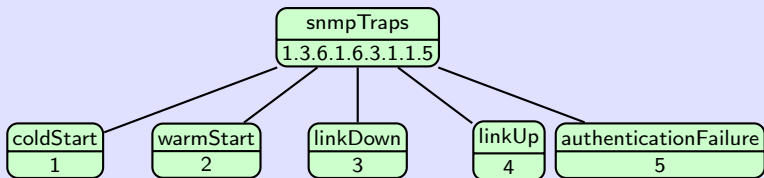
- ▶ system (1.3.6.1.2.1.1)  
informations générales sur l'équipement sur lequel se trouve l'agent
- ▶ interfaces (1.3.6.1.2.1.2)  
informations sur les interfaces réseau (ex : statuts des interfaces)
- ▶ ip (1.3.6.1.2.1.4)  
informations sur la pile IP (ex : contenu de la table de routage)
- ▶ tcp (1.3.6.1.2.1.6), udp (1.3.6.1.2.1.7), snmp (1.3.6.1.2.1.11)  
statistiques TCP, UDP et SNMP (ex : connexions TCP ouvertes)



- ▶ sysDescr (1.3.6.1.2.1.1.1)  
description textuelle
- ▶ sysUpTime (1.3.6.1.2.1.1.3)  
temps (exprimé en  $\frac{1}{100}$  de sec.) depuis lequel l'équipement est en marche
- ▶ sysContact (1.3.6.1.2.1.1.4)  
adresse mail de l'administrateur de l'équipement
- ▶ sysName (1.3.6.1.2.1.1.5)  
nom de l'équipement
- ▶ sysLocation (1.3.6.1.2.1.1.6)  
emplacement physique de l'équipement
- ▶ sysServices (1.3.6.1.2.1.1.7)  
services rendus par l'équipement



- ▶ Branche dans laquelle les entreprises peuvent insérer des objets spécifiques à leurs équipements.
- ▶ Chaque objet fils de enterprises est géré par la compagnie correspondante.
- ▶ Ces MIB sont dites privées.



branche contenant la définition des notifications SNMPv2c standard  
(Les notifications sont aussi définies comme des objets de la MIB.)

- ▶ `coldStart` (1.3.6.1.6.3.1.1.5.1)  
redémarrage de l'agent avec réinitialisation de la configuration
- ▶ `warmStart` (1.3.6.1.6.3.1.1.5.2)  
redémarrage de l'agent sans changement de configuration
- ▶ `linkDown` (1.3.6.1.6.3.1.1.5.3)  
interface désactivée
- ▶ `linkUp` (1.3.6.1.6.3.1.1.5.4)  
interface activée
- ▶ `authenticationFailure` (1.3.6.1.6.3.1.1.5.5)  
pb. d'authentification (p.ex., requête reçue avec une mauvaise communauté)

Définition de l'objet system.

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

Définition de l'objet sysUpTime, fils n°3 de l'objet system.

```
sysUpTime OBJECT-TYPE
    SYNTAX      TimeTicks
    ACCESS      read-only
    STATUS      mandatory
    DESCRIPTION
        "The time (in hundredths of a second) since the
         network management portion of the system was last
         re-initialized."
    ::= { system 3 }
```

Définition de l'objet sysContact, fils n°4 de l'objet system.

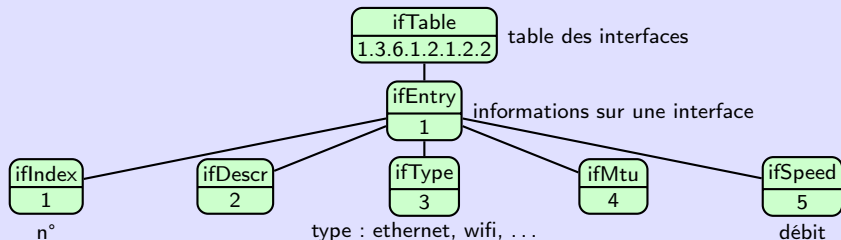
```
sysContact OBJECT-TYPE  
  SYNTAX DisplayString (SIZE (0..255))  
  ACCESS read-write  
  STATUS mandatory  
  DESCRIPTION  
    "The textual identification of the contact person  
    for this managed node, together with information  
    on how to contact this person."  
 ::= { system 4 }
```

Définition de la notification `coldStart`, fils n°1 de l'objet `snmpTraps`.

```
coldStart NOTIFICATION-TYPE
  STATUS current
  DESCRIPTION
    "A coldStart trap signifies that the SNMP entity,
     supporting a notification originator application, is
     reinitializing itself and that its configuration may
     have been altered."
 ::= { snmpTraps 1 }
```

- ▶ objet scalaire  $\Rightarrow$  objet ayant une seule instance
  - ▶ par exemple : les objets de la branche `system` : `sysName`, `sysLocation`, `sysUpTime`, ...
  - ▶ L'OID de l'objet doit être suivi de ".0" dans les requêtes et réponses SNMP.
- ▶ objet tabulaire  $\Rightarrow$  objet ayant plusieurs instances
  - ▶ Les objets tabulaires sont définis avec le type `SEQUENCE`.
  - ▶ L'OID de l'objet doit être suivi de ".i" dans les requêtes et réponses SNMP avec `i` = numéro d'instance.

IfTable : branche contenant les informations sur les interfaces réseau



```

ifTable OBJECT-TYPE
    SYNTAX SEQUENCE OF IfEntry ... ::= { interfaces 2 }
ifEntry OBJECT-TYPE
    SYNTAX IfEntry ... ::= { ifTable 1 }
ifDescr OBJECT-TYPE
    SYNTAX DisplayString (SIZE (0..255)) ... ::= { ifEntry 2 }
  
```

On a donc :

- ▶ 1.3.6.1.2.1.2.2.1.5.1 = débit (ifSpeed) de la 1<sup>ère</sup> interface réseau
- ▶ 1.3.6.1.2.1.2.2.1.3.2 = type de la 2<sup>ème</sup> interface réseau

- ▶ boîte à outils linux contenant :
  - ▶ des commandes d'interrogation d'agents
  - ▶ un agent SNMP
- ▶ principales commandes :
  - ▶ `snmpset`, `snmpget`, `snmpgetnext` et `snmpbulkget` : envoi de requêtes
  - ▶ `snmptrap` : envoi de notifications
  - ▶ `snmpinform` : envoi de informs
  - ▶ `snmpwalk` et `snmpbulkwalk` : exploration d'une branche de la MIB
    - ▶ Envoi de plusieurs `getRequest` (pour `snmpwalk`) ou `getBulkRequest` (pour `snmpbulkwalk`).
- ▶ Options requises par toutes les commandes :
  - ▶ `-v V`  
version de SNMP utilisée ( $V = 1, 2c$  ou  $3$ )
  - ▶ `-c COMMUNAUTE`  
communauté utilisée

Pour récupérer la valeur de sysName :

```
$ snmpget -v2c -cpublic localhost .1.3.6.1.2.1.1.5.0
SNMPv2-MIB::sysName.0 = STRING: localhost.localdomain
```

Pour modifier sa valeur :

```
$ snmpset -v2c -cprivate localhost .1.3.6.1.2.1.1.5.0 s "pc-de-sami"
SNMPv2-MIB::sysName.0 = STRING: pc-de-sami
```

La valeur de certains objets n'est pas modifiable (p.ex., sysDescr) :

```
$ snmpset -v2c -cprivate localhost .1.3.6.1.2.1.1.1.0 s "pc-de-sami"
Error in packet.
Reason: notWritable (That object does not support modification)
Failed object: SNMPv2-MIB::sysDescr.0
```

On peut aussi récupérer (ou modifier) plusieurs objets avec une seule requête.

```
$ snmpget -v2c -cpublic localhost .1.3.6.1.2.1.1.4.0 .1.3.6.1.2.1.1.6.0
SNMPv2-MIB::sysContact.0 = STRING: sami.evangelista@iutv.fr
SNMPv2-MIB::sysLocation.0 = STRING: IUTV (0108)
```

Avec un getNextRequest, on récupère l'objet suivant dans la MIB (via un parcours de l'arbre) :

```
$ snmpgetnext -v2c -cpublic -On localhost .1.3.6.1
.1.3.6.1.2.1.1.1.0 = STRING: Linux localhost.localdomain 4.17.19-200.f
$ snmpgetnext -v2c -cpublic -On localhost .1.3.6.1.2.1.1.3.0
.1.3.6.1.2.1.1.4.0 = STRING: sami.evangelista@iutv.fr
```

(Option `-On` = sortie numérique (affiche les OID plutôt que les noms))

`snmpwalk` permet de retrouver tous les objets dont l'OID est prefixé par un OID donné en argument :

```
$ snmpwalk -v2c -cpublic -On localhost .1.3.6.1.2.1.1
.1.3.6.1.2.1.1.1.0 = STRING: Linux localhost.localdomain 4.17.19-200.f
.1.3.6.1.2.1.1.2.0 = OID: .1.3.6.1.4.1.8072.3.2.10
.1.3.6.1.2.1.1.3.0 = Timeticks: (661096) 1:50:10.96
.1.3.6.1.2.1.1.4.0 = STRING: sami.evangelista@iutv.fr
.1.3.6.1.2.1.1.5.0 = STRING: pc-de-sami
...
```

- ▶ snmpwalk et snmpbulkwalk sont des commandes permettant d'explorer une branche de la MIB.
- ▶ Mais ce ne sont pas des requêtes du protocole.
- ▶ Elles utilisent des requêtes **getNextRequest** (pour snmpwalk) et **getBulkRequest** (pour snmpbulkwalk) pour leur exploration.
- ▶ Les seules requêtes SNMP sont : set, get, getNext, et getBulk.

- ▶ schéma d'encodage utilisé pour tous les messages SNMP
- ▶ Tout champ est décomposé en trois parties :

Type	Length	Value
1 o.	1 o.	N o.

- ▶ Type (1 o.) = type du champ (code normalisé)
- ▶ Length (1 o.) = nb. d'octets dans la partie Value
- ▶ Value (N o.) = valeur réelle du champ

- ▶ type primitif  $\Leftrightarrow$  Le champ Value ne peut pas être redécomposé en sous-champs.

Type	Code hexadécimal
Integer	0x02
String	0x04
Null	0x05
OID	0x06
IpAddress	0x40
Counter	0x41
Gauge	0x42
TimeTicks	0x43

## Exemples

- ▶ 02 01 04 = un entier (02) codé sur un octet (01) et valant 4
- ▶ 04 02 79 6F = une chaîne de caractères (04) de deux (02) octets valant "yo" (79 6F en ASCII)

- ▶ type composite  $\Leftrightarrow$  Le champ Value est une suite de sous-champs (eux-mêmes encodés en TLV).

Type	Code hexadécimal
sequence	0x30
getRequest	0xA0
getNextRequest	0xA1
response	0xA2
setRequest	0xA3
getBulkRequest	0xA5
informRequest	0xA6
trap	0xA7

- ▶ Le contenu d'un message est donc encodé en TLV.

### Exemple

- ▶ 30 07 02 01 04 04 02 79 6F = une séquence (30) de 7 octets contenant l'entier 4 et la chaîne de caractères "yo"

## Règle de codage des 2 premiers nombres de l'OID

- ▶ Les deux premiers nombres  $x.y$  d'un OID sont codés sur 1 octet valant  $40x + y$ .
- ▶ Exemples : la valeur de l'OID 1.3.4.5 sera : 2B 04 05
  - ▶  $2B = 2 \times 16 + 11 = 43 = 40 \times 1 + 3$

## Règle de codage des grands nombres ( $> 127$ )

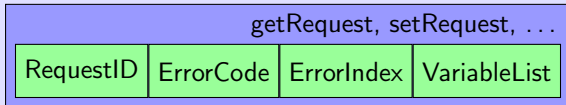
- ▶ Problème : comment coder 400 dans un OID ?
  - ▶ 400 est codé sur deux octets : 01 90 en hexadécimal.
  - ▶ Or 01 90 pourrait être interprété comme deux nombres : 1.144.
- ▶ Solution utilisée :
  - ▶ Seuls les 7 bits de poids faible sont utilisés pour coder les nombres.
  - ▶ Le 8<sup>ème</sup> bit (de poids fort) restant vaut
    - ▶ 0 si l'octet est le dernier apparaissant dans le nombre
    - ▶ 1 si l'octet suivant fait aussi partie du nombre.
- ▶ Exemple :  $400 = 11\ 0010000_2$   
 $\Rightarrow 400$  sera codé  $10000011\ 00010000$  en binaire, soit 83 10 en hexa.

- ▶ Les messages SNMP ont tous la même structure.
- ▶ Tous les champs sont encodés selon le schéma TLV.
- ▶ Un message est encodé comme une séquence (au sens TLV) de 3 éléments.



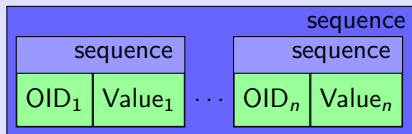
- ▶ Version = 0 pour SNMPv1, 1 pour SNMPv2c, 3 pour SNMPv3
- ▶ Community = chaîne de caractères contenant la communauté.
- ▶ PDU = contenu du message

- ▶ Structure du PDU identique quel que soit le type du message.
- ▶ On utilise les types composite `getRequest`, `setRequest`, ... pour déterminer le type du message.



- ▶ RequestID = identifiant de requête
  - ▶ choisi aléatoirement par l'émetteur de la requête
  - ▶ et recopié dans la réponse
- ▶ ErrorCode = code d'erreur
  - ▶ toujours à 0 (pas d'erreur) dans une requête
- ▶ ErrorIndex = numéro de l'OID dans VariableList qui a causé l'erreur (si ErrorCode  $\neq$  0)
- ▶ VariableList = séquence de couples (OID, valeur)

- ▶ Un message SNMP contient toujours une liste (séquence) d'OID avec leurs valeurs.
- ▶ Chaque couple (OID, valeur) est lui-même encodé comme une séquence.



## Valeurs associées aux objets envoyés

- ▶ Dans les requêtes set, les valeurs sont celles que la requête tente d'affecter aux objets.
- ▶ Dans les requêtes get, les valeurs sont toujours vides (type Null).
- ▶ Dans les réponses (qu'elle que soit la requête envoyée) les valeurs sont celles des objets de la requête (après modification dans le cas d'un set).

Quelques codes courant :

0x00 noError — pas d'erreur

0x01 tooBig — réponse trop longue pour être transportée

0x06 noAccess — pas d'accès à l'objet demandé (p.ex., la communauté n'a pas le droit de modifier l'objet demandé)

0x07 wrongType — la valeur fournie dans une requête set n'a pas le bon type

0x11 notWritable — tentative de modification d'une variable accessible en lecture seule

```
▶ Frame 7: 99 bytes on wire (792 bits), 99 bytes captured (792 bits) on interface 0
▶ Ethernet II, Src: Dell_3e:67:dd (b8:ac:6f:3e:67:dd), Dst: Cisco_67:90:40 (1c:1d:86:67:90:40)
▶ Internet Protocol Version 4, Src: 192.168.15.24, Dst: 192.168.15.100
▶ User Datagram Protocol, Src Port: 46864, Dst Port: 161
▼ Simple Network Management Protocol
  version: v2c (1)
  community: commR0
  ▼ data: get-request (0)
    ▼ get-request
      request-id: 1713239524
      error-status: noError (0)
      error-index: 0
      ▼ variable-bindings: 2 items
        ▶ 1.3.6.1.2.1.1.4.0: Value (Null)
        ▶ 1.3.6.1.2.1.1.6.0: Value (Null)
```

2 objets demandés :  
sysContact et sysLocation

▶ message getRequest envoyé par la commande :

```
$ snmpget -v2c -ccommR0 192.168.15.100 \  
1.3.6.1.2.1.1.4.0 \  
1.3.6.1.2.1.1.6.0
```

▶ getRequest ⇒ Les objets apparaissant dans le message ont la valeur Null.

```
▶ Frame 5: 115 bytes on wire (920 bits), 115 bytes captured (920 bits) on interface 0
▶ Ethernet II, Src: Dell_3e:67:dd (b8:ac:6f:3e:67:dd), Dst: Cisco_67:90:40 (1c:1d:86:67:90:40)
▶ Internet Protocol Version 4, Src: 192.168.15.24, Dst: 192.168.15.100
▶ User Datagram Protocol, Src Port: 33710, Dst Port: 161
▶ Simple Network Management Protocol
  version: v2c (1)
  community: commRW
  ▶ data: set-request (3)
    ▶ set-request
      request-id: 657359147
      error-status: noError (0)
      error-index: 0
      ▶ variable-bindings: 2 items
        ▶ 1.3.6.1.2.1.1.4.0: 61646d696e406963692e6672 — admin@ici.fr en hexadécimal
        ▶ 1.3.6.1.2.1.1.6.0: 49555456 — IUTV en hexadécimal
```

2 objets à modifier :  
sysContact et sysLocation

▶ message setRequest envoyé par la commande :

```
$ snmpset -v2c -c commRW 192.168.15.100 \  
1.3.6.1.2.1.1.4.0 s admin@ici.fr \  
1.3.6.1.2.1.1.6.0 s IUTV
```

- ▶ Quand un agent envoie une notification, la liste d'objets contient toujours :
  - ▶ objet n°1 : OID = 1.3.6.1.2.1.1.3 (sysUpTime)  
temps depuis lequel l'équipement est en marche
  - ▶ objet n°2 : OID = 1.3.6.1.6.3.1.1.4.1 (snmpTrapOID)  
OID de la notification envoyée
- ▶ L'agent peut aussi placer dans la notification d'autres objets pouvant donner des informations utiles au NMS.

Par exemple :

- ▶ sysName pour avoir le nom de l'équipement
- ▶ sysContact pour avoir l'adresse électronique de l'administrateur
- ▶ nom de l'interface désactivée dans le cas d'une notification linkDown

```
Frame 322: 207 bytes on wire (1656 bits), 207 bytes captured (1656 bits) on interface 0
Ethernet II, Src: Cisco_67:90:40 (1c:1d:86:67:90:40), Dst: Dell_3e:67:dd (b8:ac:6f:3e:67:dd)
Internet Protocol Version 4, Src: 192.168.15.100, Dst: 192.168.15.24
User Datagram Protocol, Src Port: 61296, Dst Port: 162
Simple Network Management Protocol
  version: v2c (1)
  community: trapComm
  data: snmpV2-trap (7)
    snmpV2-trap
      request-id: 1
      error-status: noError (0)
      error-index: 0
      variable-bindings: 6 items
        1.3.6.1.2.1.1.3.0: 625508
        1.3.6.1.6.3.1.1.4.1.0: 1.3.6.1.6.3.1.1.5.3 (iso.3.6.1.6.3.1.1.5.3)
        1.3.6.1.2.1.2.2.1.1.10007: 10007
        1.3.6.1.2.1.2.2.1.2.10007: 4661737445746865726e6574302f37
        1.3.6.1.2.1.2.2.1.3.10007: 6
        1.3.6.1.4.1.9.2.2.1.1.20.10007: 646f776e
```

sysUpTime

snmpTrapOID

autres objets de la MIB insérés par l'agent dans la notification

- ▶ 1<sup>er</sup> objet : (sysUpTime, 625508)  
⇒ équipement démarré depuis 625508 centièmes de seconde
- ▶ 2<sup>ème</sup> objet : (snmpTrapOID, 1.3.6.1.6.3.1.1.5.3)  
⇒ notification de type linkDown (1.3.6.1.6.3.1.1.5.3, voir diapo 24).
- ⇒ notification envoyée suite à la désactivation d'une interface réseau

## Différences entre SNMPv1 et SNMPv2c

- ▶ nouveau type de message : inform
- ▶ nouveau type de requête : getBulkRequest
  - ▶ Avec la v1, pour récupérer plusieurs objets consécutifs de la MIB : plusieurs getNextRequest consécutifs (⇒ nombreux aller-retours).
  - ▶ Un getBulkRequest permet d'en demander plusieurs en même temps.
- ▶ structure des messages de notification simplifiée

Pas de modifications importantes entre la v1 et la v2c.

## Différences entre SNMPv2c et SNMPv3

- ▶ Mécanismes de sécurité ajoutés
  - ▶ authentification
  - ▶ chiffrement des données

1. Introduction à la supervision des réseaux
2. SNMP : protocole de supervision
3. Centralisation de journaux avec syslog
4. Éléments de qualité de service

- ▶ fichiers (généralement texte) dans /var/log
- ▶ Ils contiennent des descriptions d'événements générés par les processus (p.ex., démarrage d'un service) ou par le noyau (p.ex., gestion des périphériques).
- ▶ L'analyse des journaux est primordiale pour la maintenance et la sécurité.
  - ▶ fournit un déroulé des événements qui peuvent expliquer le problème
- ▶ Problème : nombreux fichiers de log qui peuvent être présents sur différentes machines.

- ▶ protocole de la couche applicative
- ▶ Syslog utilise (généralement) UDP pour le transport.
- ▶ défini dans les RFC 3164 et 5424
- ▶ port par défaut : 514
- ▶ objectif : centralisation de journaux sur des serveurs syslog par l'échange de messages de log
- ▶ Comme SNMP, syslog est implanté dans la plupart des OS et équipements réseau.

## Dans la RFC3164

### ▶ Partie priorité

- ▶ catégorie (ou *facility*) : code de 0 à 23 (voir diapos suivantes)
- ▶ gravité : code de 0 à 7 (voir diapos suivantes)

La priorité associée à un message est  $8 \times \text{catégorie} + \text{gravité}$ . On la note généralement sous la forme `catégorie.gravité`.

### ▶ Partie en-tête

- ▶ date et heure de la génération du message
- ▶ nom ou @IP de la machine ayant généré le message

### ▶ Partie message

- ▶ un texte brut donnant des informations supplémentaires

## Dans la RFC5424

- ▶ plus d'info. dans la partie en-tête (p.ex. nom et PID du processus ayant généré le message)
- ▶ corps du message plus structuré

24 catégories parmi lesquelles

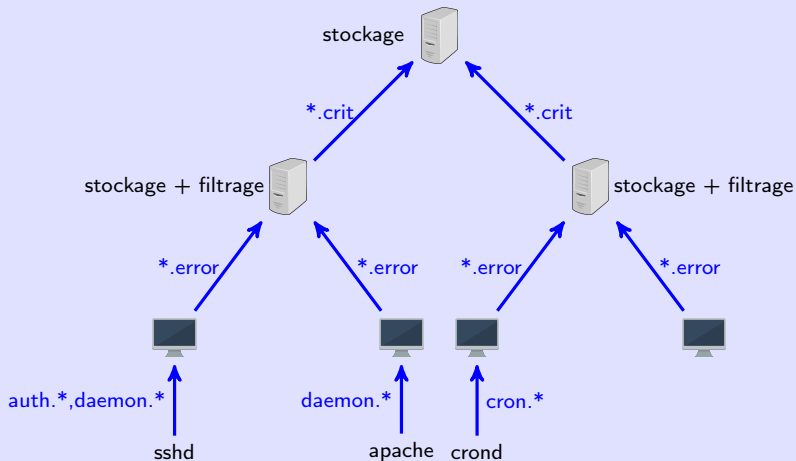
- ▶ 0 = kernel — messages du noyau
- ▶ 1 = user — messages de l'espace utilisateur
- ▶ 2 = mail — messages du système de messagerie
- ▶ 3 = daemon — messages des services
- ▶ 4 = auth — messages d'authentification
- ▶ 5 = syslog — messages générés par le service syslog

8 niveaux

- ▶ 0 = emergency — système inutilisable
- ▶ 1 = alert — intervention requise
- ▶ 2 = critical — conditions critiques
- ▶ 3 = error — erreur de fonctionnement
- ▶ 4 = warning — erreurs potentielles
- ▶ 5 = notice — information importante (mais pas une erreur)
- ▶ 6 = informational — information de moindre importance
- ▶ 7 = debug — message de débogage

- ▶ Que fait-il ?  
réception de messages syslog émis par les processus locaux ou par d'autres serveurs syslog, puis éventuellement :
  - ▶ écriture du message dans un fichier
  - ▶ redirection du message vers un autre serveur
  - ▶ écriture dans un tube (⇒ permet l'exécution d'une commande)
  - ...
- ▶ fichier de configuration principal : `/etc/rsyslog.conf`
- ▶ Ce fichier contient des règles de la forme :  
  
sélecteur action
  - ▶ sélecteur : détermine les messages (selon leurs catégories et gravités) sur lesquels s'applique cette règle
  - ▶ action : ce que rsyslog fait de ces messages(Plusieurs règles peuvent s'appliquer pour un même message.)

```
# tous les messages de la catégorie auth avec un niveau de gravité <=  
# error sont écrits dans le fichier /var/log/authentication.log  
auth.error          /var/log/authentication.log  
# tous les messages de la catégorie auth  
auth.*             /var/log/authentication.log  
# tous les messages des catégories auth et kernel  
auth.*;kernel.*   /var/log/important.log  
# tous les messages ('-' <=> écritures non synchrones)  
*.*               -/var/log/messages  
# tous les messages sauf ceux de la catégorie auth  
*.*;auth.none     -/var/log/messages  
# tous les messages avec une gravité strictement égale à critique  
*=crit           /var/log/critical.log  
# les messages de la cat. mail avec un niveau de gravité > notice  
mail.*;mail.!notice -/var/log/messages  
# redirection vers 10.0.0.1 avec UDP (un seul @) et TCP (deux @)  
syslog.*          @10.0.0.1  
syslog.*          @@10.0.0.1  
# messages de cat. cron redirigés vers le tube /var/run/rsylog/pipe  
cron.*            |/var/run/rsylog/pipe
```



- ▶ Quand on fait `cmd1 | cmd2` l'OS crée un tube anonyme.
- ▶ C'est un fichier de type particulier (type pipe) dans lequel `cmd1` va écrire et `cmd2` va lire.
- ▶ On peut aussi créer des tubes nommés avec `mkfifo`.

```
$ mkfifo /tmp/test-fifo
$ ls -l /tmp/test-fifo
prw-rw-r-- 1 sami sami 0 10 déc. 10:53 /tmp/test-fifo
```

- ▶ C'est un moyen de communication entre les processus.
- ▶ Si un processus écrit (resp. lit) dans un tube il faut qu'un autre processus soit en train de lire (resp. d'écrire) dans le tube. Sinon il est en attente. Il faut qu'il y ait un "rendez-vous" entre les deux processus.
- ▶ Par exemple :

```
$ echo salut >> /tmp/test-fifo
```

*j'attends qu'un  
processus lise dans le fifo*

```
$
```

```
$ cat /tmp/test-fifo | \
while read data
do echo $data ; done
salut
$
```

Elle permet d'envoyer un message au serveur syslog local.

- ▶ option `-p` pour préciser la priorité (par défaut `user.notice`)  
(Remarque : impossible d'envoyer des messages de la catégorie `kernel`)
- ▶ option `-n` pour envoyer le message à un serveur syslog distant

Exemples :

```
logger -p cron.error "une erreur cron s'est produite"  
logger -n 10.0.0.1 -p auth.notice "nouvel connexion"
```

## Quels messages stocker/centraliser ?

- ▶ compromis entre les infos que l'on veut garder et les performances
- ▶ par exemple, éviter de mettre sur les clients syslog :

```
*.*      @@10.0.1.100
```

⇒ réseau saturé avec tous les messages de log (même les plus insignifiants comme ceux de débogage)

- ▶ Mais envoyer uniquement les messages les plus graves n'est pas forcément très utile pour résoudre les problèmes.

## Gestion des fichiers de logs

- ▶ utilisation du service logrotate pour organiser une rotation des logs et/ou éviter qu'ils n'occupent trop de place
- ▶ Exemple de règle logrotate :
  - ▶ stocker uniquement les logs des 4 derniers mois dans des fichiers séparés (/var/log/syslog.1, ... /var/log/syslog.4)

1. Introduction à la supervision des réseaux
2. SNMP : protocole de supervision
3. Centralisation de journaux avec syslog
4. **Éléments de qualité de service**

*Capacité d'un réseau à véhiculer des flux dans de bonnes conditions.*

- ▶ flux = séquence de paquets d'une source vers une destination
- ▶ Critères de QoS : fiabilité, délai, gigue, bande passante.

Une série statistique  $X$  est une suite de valeurs  $X_1, \dots, X_n$ .

## Moyenne

$$\bar{X} = \frac{1}{n} \sum_{i=1}^n X_i$$

**Variance** — moyenne des carrés des écarts à la moyenne

$$V(X) = \frac{1}{n} \sum_{i=1}^n (X_i - \bar{X})^2$$

## Écart-type

$$\sigma(X) = \sqrt{V(X)}$$

**Médiane** — valeur  $Me$  qui sépare la série en 2

Soit  $X' = X'_1, \dots, X'_n$  la série  $X$  ordonnée.

- ▶ Si  $n$  est impair,  $Me = X'_k$  avec  $k = \frac{n+1}{2}$  (valeur centrale)
- ▶ Si  $n$  est pair,  $Me = \frac{X'_k + X'_{k+1}}{2}$  avec  $k = \frac{n}{2}$  (moyenne des deux valeurs centrales)

Soit un flux entre deux points A et B du réseau.

### Fiabilité

- ▶ Taux d'erreur tolérable du flux.

Soit un flux entre deux points A et B du réseau.

## Fiabilité

- ▶ Taux d'erreur tolérable du flux.

## Délai (ou latence)

- ▶ Temps mis par un paquet pour traverser le réseau de A à B.
- ▶ Principales causes :
  - ▶ temps de traversée des équipements (routeurs)
  - ▶ temps de propagation (élevé pour les réseaux longues distances)

Soit un flux entre deux points A et B du réseau.

## Fiabilité

- ▶ Taux d'erreur tolérable du flux.

## Délai (ou latence)

- ▶ Temps mis par un paquet pour traverser le réseau de A à B.
- ▶ Principales causes :
  - ▶ temps de traversée des équipements (routeurs)
  - ▶ temps de propagation (élevé pour les réseaux longues distances)

## Gigue

- ▶ Variation dans les délais d'acheminement.
- ▶ utilisation de la variance statistique
- ▶ gigue = 0  $\Leftrightarrow$  délai d'acheminement constant

Soit un flux entre deux points A et B du réseau.

## Fiabilité

- ▶ Taux d'erreur tolérable du flux.

## Délai (ou latence)

- ▶ Temps mis par un paquet pour traverser le réseau de A à B.
- ▶ Principales causes :
  - ▶ temps de traversée des équipements (routeurs)
  - ▶ temps de propagation (élevé pour les réseaux longues distances)

## Gigue

- ▶ Variation dans les délais d'acheminement.
- ▶ utilisation de la variance statistique
- ▶ gigue = 0  $\Leftrightarrow$  délai d'acheminement constant

## Bande passante

- ▶ Débit max. de bout en bout entre A et B.

Critères de QoS pour quelques applications :

Application	Fiabilité	Délai	Gigue	BP
Mail	Haute	Faible	Faible	Faible
FTP	Haute	Faible	Faible	Haute
Web	Haute	Moyenne	Faible	Moyenne
VoIP	Faible	Haute	Haute	Faible
Conf. vidéo	Faible	Haute	Haute	Haute

- ▶ Mail, FTP et web : applications de transfert de fichiers. faibles exigences sur les critères temporels (délai et gigue). forte exigence sur la fiabilité
- ▶ VoIP et Conf. vidéo : applications **temps-réel**. fiabilité secondaire. fortes exigences sur les critères temporels

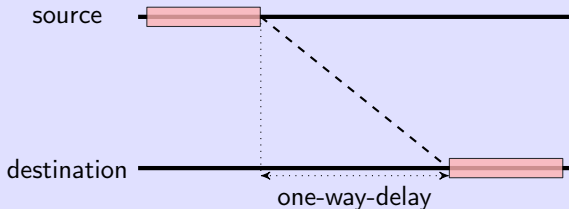
- ▶ Contrat passé entre
  - ▶ un fournisseur de service (p.ex., un fournisseur d'accès à Internet)
  - ▶ et un client (p.ex., une entreprise)et définissant
  - ▶ le type de service fourni
  - ▶ et les exigences de QoS attendues.
- ▶ Les exigences de QoS peuvent être fournies sous forme de **métriques**.  
Par exemple :
  - ▶ Bande passante garantie de 100 Mb/s.
  - ▶ Taux de perte de paquets  $< 0.001$ .
  - ▶ Délai max de bout en bout  $< 100ms$ .
  - ...

- ▶ IPPM = IP Performance Measurement
- ▶ Groupe de l'IETF chargé de la définition
  - ▶ de métriques réseau pour les réseaux IP ;
  - ▶ et de méthodes de calcul de ces métriques.
- ▶ Quelques métriques :
  - ▶ RFC 2679 : A One-way Delay Metric for IPPM  
mesure du délai d'acheminement
  - ▶ RFC 2680 : A One-way Packet Loss Metric for IPPM  
mesure du taux de perte (dans un seul sens)
  - ▶ RFC 2681 : A Round-trip Delay Metric for IPPM  
mesure du délai aller-retour

- ▶ La plupart des métriques IPPM sont dépendantes du type de paquet.
  - ▶ protocole de transport
  - ▶ ports source et destination
  - ▶ taille du paquet
- ▶ Quand une métrique est spécifique à un type de paquet, son nom est préfixé par `typePaquet`.  
⇒ Le nom de la métrique est alors `typePaquet-metrique`.
- ▶ Lors de la réalisation d'une mesure on doit préciser le type de paquet utilisé.  
Par exemple : `typePaquet` → `TCP-src1550-dst80-size800`  
(paquet TCP de 800 octets envoyé du port 1550 au port 80)

One-Way-Delay = délai dans un seul sens

- ▶ Le One-Way-Delay donne une indication plus fine que le Round-Trip-Delay (délai aller-retour).
  - ▶ Possibilité de routage asymétrique.
  - ▶ Pour de nombreuses applications (p.ex., streaming multimedia), on s'intéresse au délai dans un seul sens.
- ▶ Mesure le temps passé par le paquet sur le réseau :



- ▶ Nécessite une synchronisation des horloges (via GPS, par exemple).