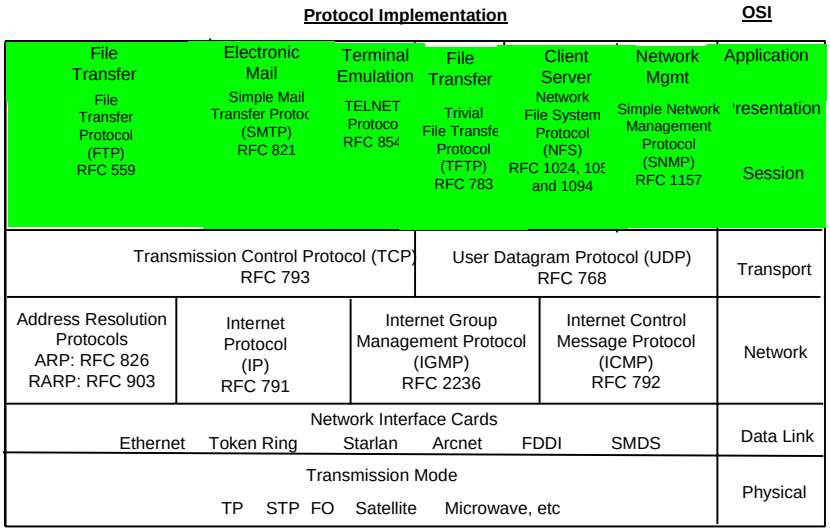
Cours 5: TCP/IP Applications et Services

Cours 5 : Plan

5.1 Principes des protocoles de la couche Applications
5.2 DNS
5.3 Electronic Mail
- SMTP, POP3, IMAP

5.4 DHCP/BOOTP
5.5 NFS
5.6 Web et HTTP
5.7 FTP
5.8 Telnet/Rlogin
5.9 SNMP

Les couches de protocoles : TCP/IP et le modèle OSI



3

Applications réseau

Processus: programme s'exécute sur une host.

- sur la même host, deux processus interagissent en utilisant la communication interprocessus (règle définie par l'OS)
- Processus sur deux machines différentes communiquent par message à travers un protocole de la couche application

Agent utilisateur: interfaces avec l'utilisateur au dessus et le réseau en dessous

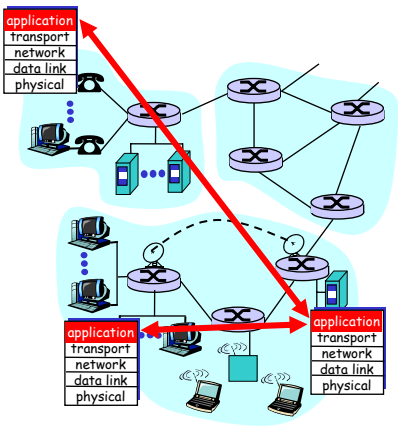
- Implémentations interface utilisateur et le protocole du niveau application
  - Web: browser
  - E-mail: mail reader
  - streaming audio/vidéo: media player

4

Applications et protocoles de la couche application

Application: processus distribués
- e.g., e-mail, Web, telnet
- s'exécutant sur les terminaux (hosts)
- échangent des messages pour implémenter l'application

Protocoles de la couche application
- définissent des messages échangés par les applications et les actions prises
- utilisent les services de communication fournis par les protocoles de la couche inférieure (TCP, UDP)



5

Protocole de la couche application

- Types de messages échangés, ex, messages de demande et de réponse
- la syntaxe adoptée par les différents types de message: soit les différents champs qu'il contient et leur délimitation
- la sémantique des différents champs, c'est à dire le sens des informations qu'ils renferment

les règles utilisées pour déterminer quand et comment un processus doit envoyer ou répondre à un message

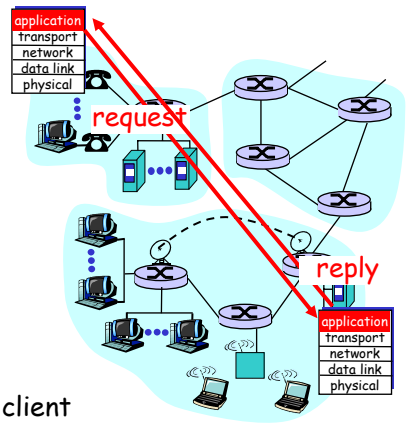
Protocoles domaines publics:
- définis dans les RFCs
- Permettent l'interopérabilité
- ex, HTTP, SMTP

Protocoles propriétaires:
- ex, KaZaA

6

Paradigme Client-server

Application réseau contient deux parties: *client* et *serveur*



Client:

- initialise le contact avec le serveur
- Demande de service de serveur
- Web: client implementé dans le browser; e-mail: dans mail reader

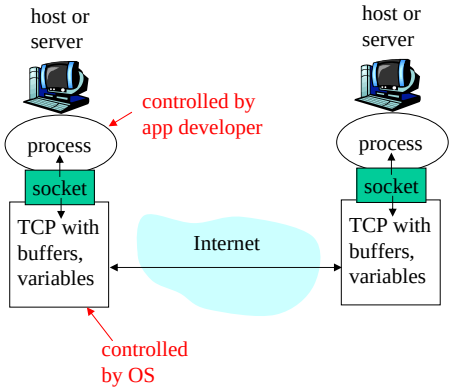
Serveur:

- fournit le service demandé par le client
- e.g., Web server envoie la page Web demandée, mail server délivre e-mail

7

Processus de communication à travers le réseau

- processus reçoit /envoie les messages à travers son socket
- socket analogue à une porte
  - l'envoie de message à travers cette porte (interface)
  - le processus d'envoi suppose qu'il y à une infrastructure de transport de l'autre coté de cette porte prête à prendre en charge le message et à l'emmener jusqu'à la porte de destinataire via destinataire



- API (1) le choix du protocole de transport; (2) la possibilité de définir quelques paramètres

8

Processus d'adressage:

- Un processus local a besoin d'identifier le processus distant
- Chaque host a une unique adresse IP
- Q: l'adresse IP de la machine sur laquelle le processus tourne suffit-elle pour identifier le processus?
- Réponse: Non, plusieurs processus peuvent être exécutés sur la même machine

- Identificateur inclut l'adresse IP et le numéro de port associé sur le host.
- Exemple port numbers:
  - HTTP server: 80
  - Mail server: 25

9

Services nécessaires à une application?

Data loss (transfer fiable)

- certaines apps (e.g., audio) tolèrent la perte
- autres apps (e.g., file transfer, telnet) exigent 100% transfert fiable

Contraintes de temps

- certaines apps (e.g., Internet telephony, interactive games) demandent un bas délai

Bandwidth (débit)

- certaines apps (e.g., multimédia, téléphonie par internet) exigent un débit minimal disponible
- autres apps ("elastic apps, comme ftp et web") peuvent s'adapter aux débits disponibles

10

Fonctionnement de quelques applications de réseau

Application	Data loss	Bandwidth	Time Sensitive
file transfer	no loss	elastic	no
e-mail	no loss	elastic	no
Web documents	no loss	elastic	no
real-time audio/video	loss-tolerant	audio: 5kbps-1Mbps video:10kbps-	yes, 100's msec
stored audio/video	loss-tolerant	5Mbps	yes, few secs
interactive games	loss-tolerant	same as above	yes, 100's msec
instant messaging	no loss	few kbps up	yes and no
		elastic	

11

Internet apps: application, transport protocols

Application	Application layer protocol	Underlying transport protocol
e-mail	SMTP [RFC 2821]	TCP
remote terminal access	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
file transfer	FTP [RFC 959]	TCP
streaming multimedia	proprietary (e.g. RealNetworks)	TCP or UDP
Internet telephony	proprietary	typically UDP

12

Cours 5 : Plan

5.1 Principes des protocoles de la couche Applications
5.2 DNS
5.3 Electronic Mail
- SMTP, POP3, IMAP

5.4 DHCP/BOOTP
5.5 NFS
5.6 Web et HTTP
5.7 FTP
5.8 Telnet/Rlogin
5.9 SNMP

DNS: Domain Name System

Personne: plusieurs manières de l' identifier:
- NSS, nom, N.passeport

Internet hosts, routeurs:
- IP address (32 bit)
- "nom", gaia.cs.umass.edu utilisé par le humains

Q: correspondance entre adresses IP et nom?

Domain Name System:
- *datatbase distribuée* implémentée en hiérarchie dans plusieurs *name servers*
- *protocole couche application* host, routeurs, name servers communiquent pour *résoudre* noms (translation adresse/name)

DNS name servers

Pourquoi le DNS n'est pas centralisé?

- Un seul point de rupture
- volume de trafic
- database centralisée distante
- maintenance

n'est pas *scalable!*

- Pas de serveur a toutes les correspondances nom-adresse IP

name servers locaux:

- chaque ISP, a son *local (default) name server*
- host DNS consulte en premier le name server local

name server de source autorisée:

- pour une host: stocke son adresse IP, nom
- peut accomplir la translation name/address IP

15

DNS: Racine des name servers

- contacté par un name server local qui ne peut pas résoudre le nom
- name server racine:
  - contacte name server de source autorisée si la correspondance n'est pas connue
  - obtenir la correspondance
  - retourne la correspondance au name server local

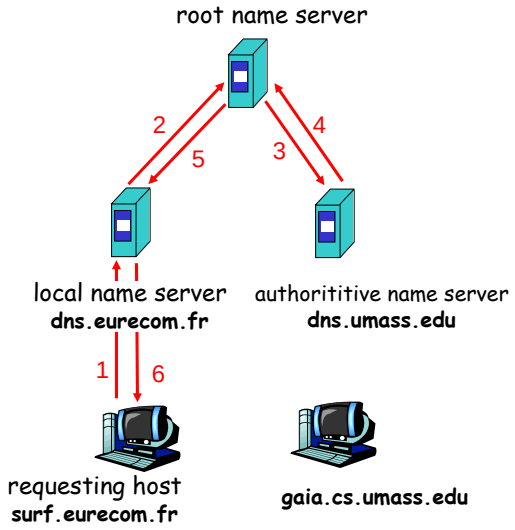
13 root name servers worldwide

16

Exemple d'un Simple DNS

La machine **surf.eurecom.fr** veut l'adresse IP de **gaia.cs.umass.edu**

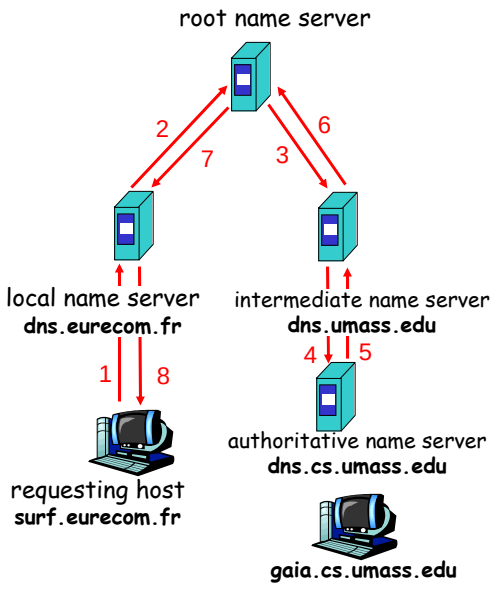
1. contacte son local DNS server, **dns.eurecom.fr**
2. **dns.eurecom.fr** contacte name server, si nécessaire
3. name server racine contacte name server de source autorisée, **dns.umass.edu**, si nécessaire



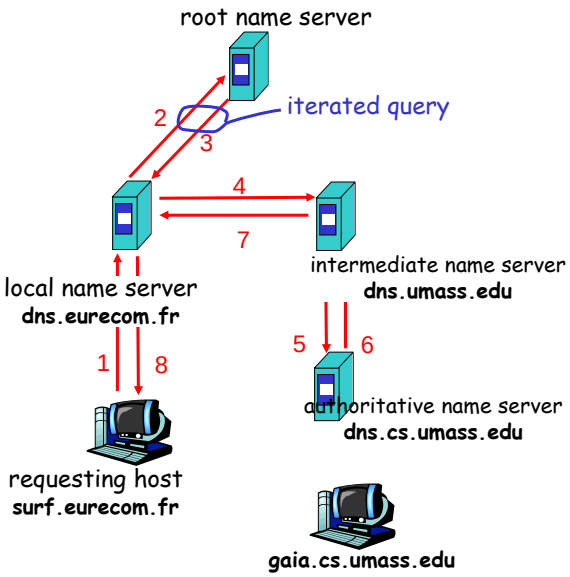
Exemple DNS

name server racine:

- Ne connaît pas le name server de source autorisée
- connaît un *intermédiaire name server:* qui lui contacte pour trouver name server de source autorisée



DNS: demandes itératives



19

DNS: mise en mémoire cache

- Le DNS utilise la mémoire cache afin de diminuer le temps de réponse et réduire le nombre de messages de transit
- IETF
  - RFC 2136
  - http://www.ietf.org/html.charters/dnsind-charter.html

20

DNS: Enregistrements

DNS: base de données répartie conserve des enregistrements de ressources (RR)

RR format: `(name, value, type, ttl)`

- Type=A
  - **name** :hostname
  - **value** :IP address
- Type=NS
  - **name** is domain (e.g. foo.com)
  - **value** IP address de name serveur de source autorisée pour ce domaine
- Type=CNAME
  - **name** le nom de serveur canonique de l'alias name

    www.ibm.com est réellement servereast.backup2.ibm.com
  - **value** est nom canonique
- Type=MX
  - **value** est un nom canonique d'un mailserver associé avec le **nom**

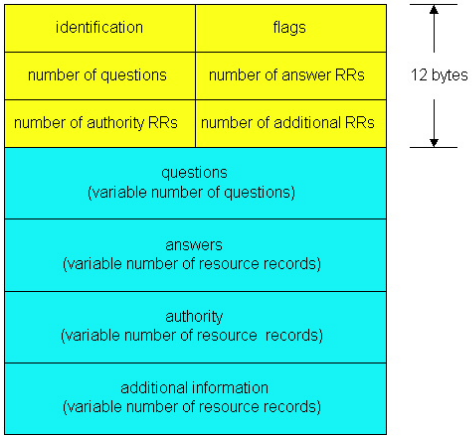
21

DNS : protocole, messages

DNS protocol : messages de *demande* et *réponse*, avec le même *format message*

msg header

- identification: 16 bits
- flags:
  - demande/réponse
  - recursion désirée
  - recursion disponible
  - Source autorisée



22

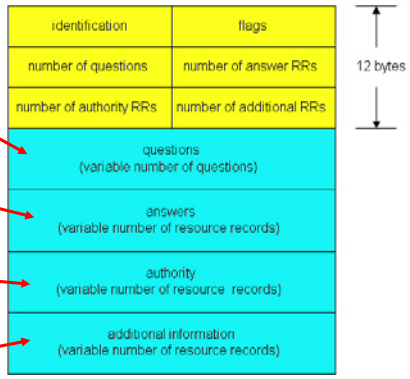
DNS: protocole, messages

Name, type fields for a query

RRs in reponse to query

records for authoritative servers

additional "helpful" info that may be used



Cours 5 : Plan

5.1 Principes des protocoles de la couche Applications

5.2 DNS

5.3 Electronic Mail

- SMTP, POP3, IMAP

5.4 DHCP/BOOTP

5.5 NFS

5.6 Web et HTTP

5.7 FTP

5.8 Telnet/Rlogin

5.9 SNMP

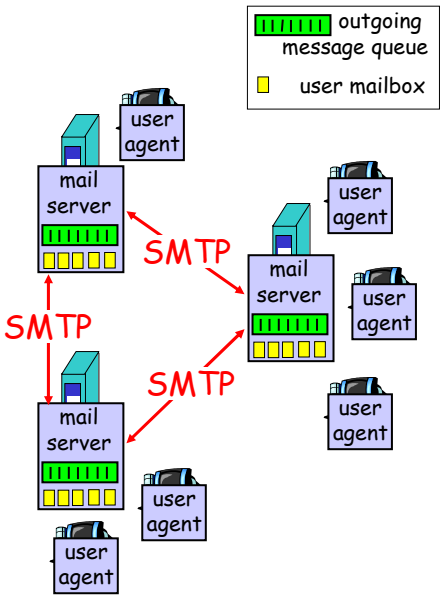
Electronic Mail

Trois éléments fondamentaux:

- Agents utilisateurs
- serveurs de messagerie
- Le protocole SMTP:simple mail transfer protocol

Agent utilisateur

- Appelé aussi lecteur de messagerie, éditeur….
- exemples, Eudora, Outlook, elm, Netscape Messenger
- Messages sortant, entrant sont stockés sur le serveur

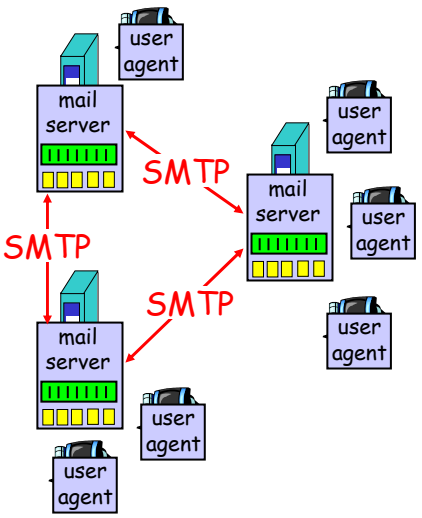


25

Electronic Mail: mail servers

Serveurs de messagerie

- mailbox contient des messages entrants pour l'utilisateur
- File de messages de messages sortants (d'être envoyés)
- SMTP protocol pour envoyer les messages entre les serveurs
  - client: sending mail server
  - "server": receiving mail server



26

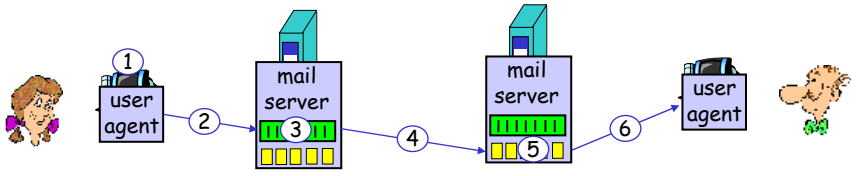
Electronic Mail: SMTP [RFC 2821]

- utilise TCP pour transférer d'email de client au serveur, port 25
- Transfert direct: serveur d'envoi au serveur de réception
- trois phases de transfert
  - La connexion
  - Transfert de messages
  - fermeture
- Interaction commande/réponse
  - commandes: ASCII
  - réponses: code d'état et phrase
- messages en ASCII de 7 bits

27

Scénario: Alice envoie un message à Bob

1) Alice utilise un UA pour composer le message "to" bob@someschool.edu
2) UA d'Alice envoie le message à son serveur de messagerie, où il et placé dans une file de messages
3) Le pôle client de SMTP opérant au niveau de serveur de messagerie d'Alice, aperçoit le message dans la file, ouvre une connection TCP avec le serveur mail de Bob
4) SMTP client envoie le message d'Alice sur la connexion TCP
5) Serveur mail de Bob place le message dans le mailbox de Bob
6) Bob ouvre son user agent pour lire le message



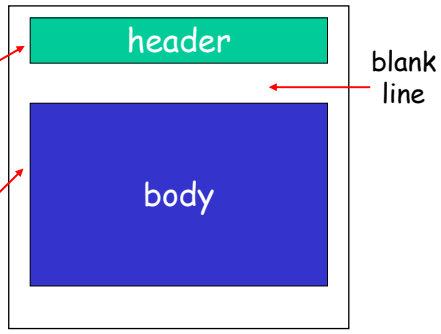
28

Format de message Mail: texte

SMTP: RFC 822:

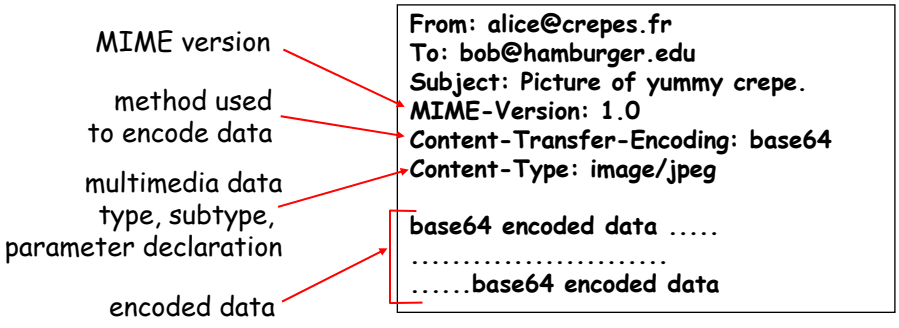
- Ligne d'en-tête,
  - To:
  - From:
  - Subject:

  different from SMTP commands!
- body
  - le "message", en caractères ASCII seulement



Format de message: multimedia extensions

- MIME: multipurpose internet mail extension, RFC 2045, 2056
- additional lines in msg header declare MIME content type



MIME types

`Content-Type: type/subtype; paramètres`

Text

- exemple subtypes: **`plain`**, **`html`**

Image

- exemple subtypes: **`jpeg`**, **`gif`**

Audio

- exemple subtypes: **`basic`** (8-bit mu-law encoded), **`32kadpcm`** (32 kbps coding)

Vidéo

- exemple subtypes: **`mpeg`**, **`quicktime`**

Application

- Le type application est utilisé pour les données ne correspondant pas à aucune autre catégorie, notamment celle qui devant être soumises à une application particulière avant d' être accessibles à l'utilisateur
- exemple subtypes: **`msword`**, **`octet-stream`**

31

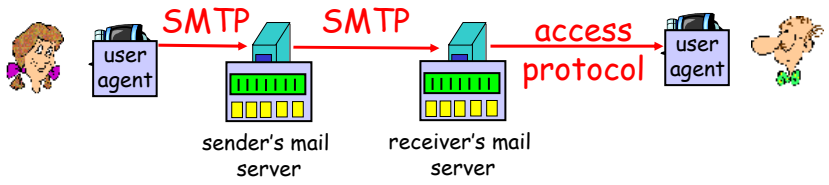
Type Multipart

```
From: alice@crepes.fr
To: bob@hamburger.edu
Subject: Picture of yummy crepe.
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary=StartOfNextPart

--StartOfNextPart
Dear Bob, Please find a picture of a crepe.
--StartOfNextPart
Content-Transfer-Encoding: base64
Content-Type: image/jpeg
base64 encoded data .....
.....
.....base64 encoded data
--StartOfNextPart
Do you want the reciple?
```

32

Protocoles d'accès à la messagerie



- SMTP: délivre/stocke au serveur de réception
- Protocole d'accès:
  - POP: Post Office Protocol [RFC 1939]
    - authorisation (agent <-->server) et download
  - IMAP: Internet Mail Access Protocol [RFC 1730]
    - Plus de caractéristiques (plus complexe)
    - manipulation des messages stockés sur le serveur
  - HTTP: Hotmail , Yahoo! Mail, etc.

33

POP3 protocol

Phase d'autorisation

- client commands:
  - **user:** declare username
  - **pass:** password
- server responses
  - **+OK**
  - **-ERR**

Phase de transaction, client:

- **list:** list message numbers
- **retr:** retrieve message by number
- **dele:** delete
- **quit**

```
S: +OK POP3 server ready
C: user bob
S: +OK
C: pass hungry
S: +OK user successfully logged on
```

```
C: list
S: 1 498
S: 2 912
S: .
C: retr 1
S: <message 1 contents>
S: .
C: dele 1
C: retr 2
S: <message 1 contents>
S: .
C: dele 2
C: quit
S: +OK POP3 server signing off
```

34

POP3 et IMAP

POP3

- Exemple précédent utilise le mode "download et delete"
- Bob ne peut pas relire e-mail si le client change
- "Download-et-keep": copie des messages sur des différents clients
- POP3 sans mémoire, ne conserve aucune information d'état d'une session à l'autre

IMAP

- Garde de tous les messages dans une place: le serveur
- permet à l'utilisateur d'organiser les messages en répertoires
- IMAP conserve des informations d'état sur ses utilisateurs d'une session à l'autre
  - Noms des répertoires et la correspondance entre les IDs de message et le nom de répertoire

35

Cours 5 : Plan

5.1 Principes des protocoles de la couche Applications
5.2 DNS
5.3 Electronic Mail
  - SMTP, POP3, IMAP
5.4 DHCP/BOOTP
5.5 NFS
5.6 Web et HTTP
5.7 FTP
5.8 Telnet/Rlogin
5.9 SNMP

36

BOOTstrap Protocol (BOOTP)

37

BOOTstrap Protocol (BOOTP)

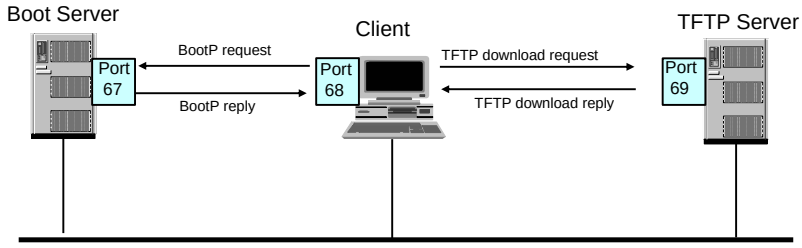
- **RARP** est un protocole au niveau physique utilisé avec les stations sans disque pour obtenir leurs adresses IP. Cependant, la workstation a besoin de :
  - Connaître l'adresse de serveur
  - Charger le système d'exploitation
  - Connaître l'adresse IP du plus proche routeur.
  - Connaître le mask du sous réseau
  - Connaître le Domain Name Server
- à cause de ces exigences, le **RARP** est remplacé par ***BOOTstrap Protocol (BOOTP)*** et amélioré par ***Dynamic Host Configuration Protocol (DHCP)***
- BOOTP était le premier standard automatique de boot dans TCP/IP

BOOTP fournit les services de base suivant:

- Le client **broadcasts** une demande dans un paquet **UDP**
- Le serveur retourne l'adresse IP et optionnellement l'endroit des fichiers à charger
- Le client utilise ***Trivial File transfer Protocol (TFTP)*** pour charger et exécuter le software

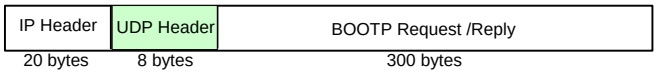
38

BOOTstrap Protocol (BOOTP)



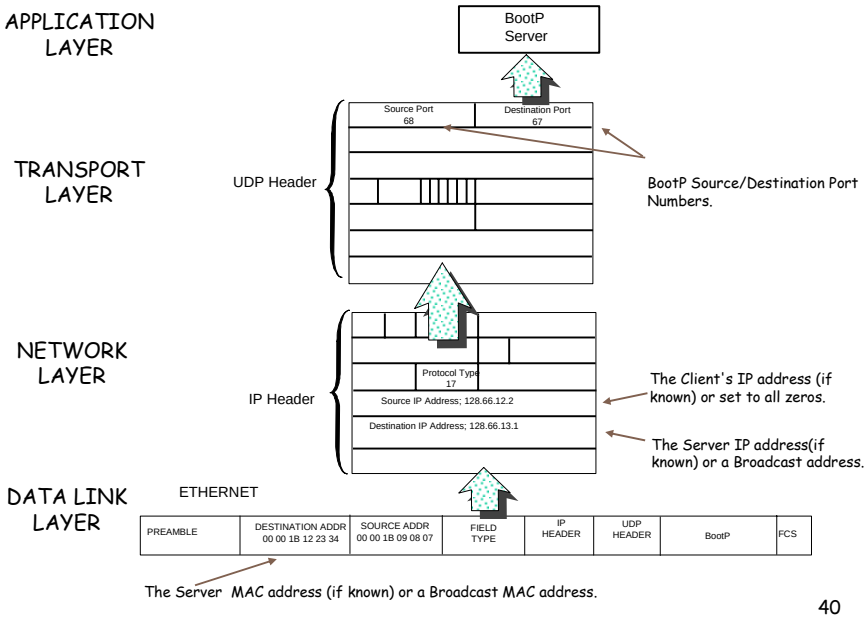
BOOTstrap Concept

1. Le client envoie un ***bootrequest*** message du **port 68** au Boot Server **port 67**, encapsulé dans **UDP**. Le client retransmettra le message en cas de non réception d'une réponse pendant le temps timeout
2. Le serveur répond sur le port 67 avec un ***bootreply*** au client sur le port 68. Le reply peut optionnellement contenir l'endroit des fichiers à charger
3. Le client demande de charger le fichier avec **TFTP request** au TFTP server sur le **port 69**.
4. Le TFTP server répond avec le chargement de fichier



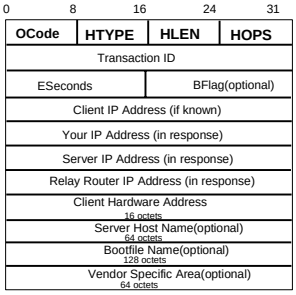
39

PROTOCOL and PORT NUMBERS



40

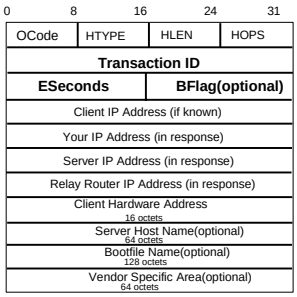
BOOTstrap Protocol (BOOTP)



- **OCode.** Le message est une demande/réponse (**1/ 2)**
- **HTYPE**. Type d'interface d'émetteur(**1 pour Ethernet)**
- **HLEN**. La longueur de l'adresse physique, contient la valeur 6 (MAC address field is 48 bits)
- **HOPS**. Le **client** met ce champ à **zéro**.
  - Si un serveur BOOTP se trouve sur un autre réseau (permettre le démarrage à travers plusieurs routeurs) , incrémente la valeur du compteur

41

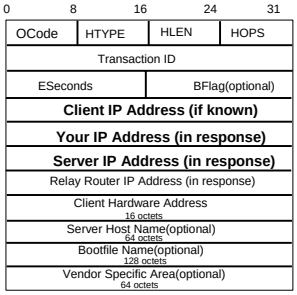
BOOTstrap Protocol (BOOTP)



- **Transaction ID**. Un nombre générer par le client qui permet l'association entre le client et le serveur (réponses /demandes)
- **ESeconds**. Indique le nombre de secondes écoulées depuis le redémarrage du client et même peut être utilisé comme un temporisateur pour la retransmission des demandes à des intervalles (4, 8, 16, 32 and 64 secondes), en utilisant une moyenne aléatoire (entre 0-4 secondes)
- **Bflag**. Généralement inutilisé. Mais quelques clients ne peuvent pas recevoir datagrammes sans être configurés précédemment avec une adresse
  - Le **Broadcast flag = 1** indique que le client à besoin de recevoir la réponses via une adresse broadcast **255.255.255.255** sur le **port 68**.

42

BOOTstrap Protocol (BOOTP)

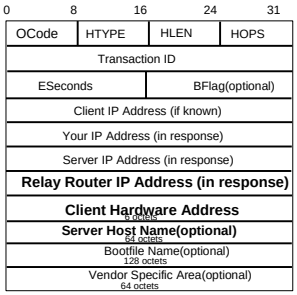


- **Client IP Address**. L'adresse IP client s'elle est connue, autrement il est à 0
- **Your IP Address**. C'est une IP address fournie par le serveur au client si le Client IP Address est initialisée à zéro

- **Server IP Address**. Le BOOTP server place le **IP address of the TFTP server**, c'est elle est connue, dans ce champ. Le client utilise cette adresse pour charger le système
  - BOOTP normalement sépare la **configuration service** de **software download service**.

43

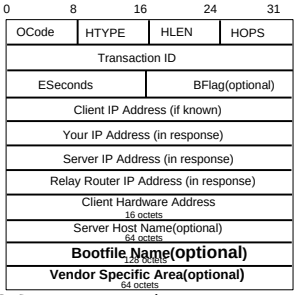
BOOTstrap Protocol (BOOTP)



- **Relay Router Address. Utilisé en cas d'utilisation d'un serveur central qui se trouve sur un réseau différent**
- **Client Hardware Address**. adresse MAC du client NIC est placée dans ce champ
  - Le serveur utilise le type hardware et l'adresse MAC client comme clé pour chercher l'adresse IP dans sa table
- **Server Host Name**. le client place le **BootP host name**, s'elle est connue.Sinon à zéro et ellle envoie un paquet avec une source IP address à 0.0.0.0 et une destination IP address à 255.255.255.255.

44

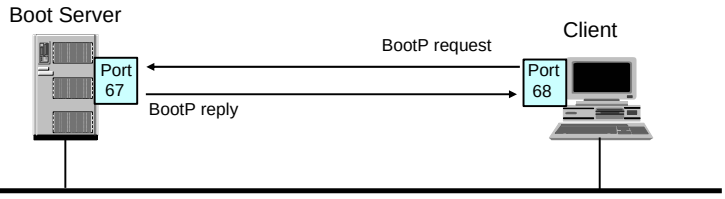
BOOTstrap Protocol (BOOTP)



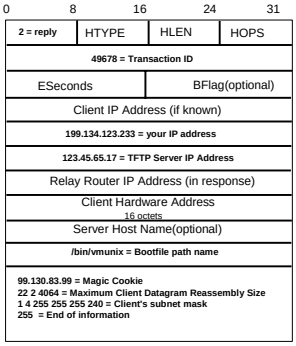
- **Bootfile Name**. Dans un **BootP Request**, ce champ contient un **nickname** (Unix, SunOS ,etc).Le BOOTP serveur était configuré avec le nickname dans un IP adresse de serveur TFTP et complète le nom du chemin
  - Dans un **BootP Reply**, le BOOTP server recherche le nickname dans sa table et place l' **IP address de TFTP server** dans le champ TFTP Server IP address et remplace le champ **Bootfile Name** avec le **complete path address** ou il réside le système sur TFTP server.
- **Vendor Specific Area**. Champ optionnel et contient l'information qui peut être passée du serveur au client. Il peut inclure **subnet mask**, **time of day**, **router addresses**, **DNS**, etc.
  - Les 4 premiers octets ont la valeur **99.130.83.99**
  - Les champs d' **options** sont constitués d' 1 octet pour le **code** et d'1 octet **length** suivi par un nombre d'octets de données

45

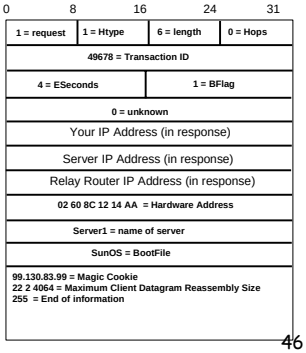
BOOTstrap Protocol (BOOTP)



BootP Reply



BootP Request



46

Dynamic Host Configuration Protocol (DHCP)

47

Dynamic Host Configuration Protocol (DHCP)

- **BOOTP** problèmes:
  - Désigné pour un **environnement** static où la configuration reste inchangée
  - N'est pas efficace pour les portables et les réseaux mobiles
  - Tout ça demande une intervention manuelle par l'administrateur système

- **Dynamic Host Configuration Protocol (DHCP)** était désigné par l'IETF pour remplacer le BOOTP
  - Permet au client d'obtenir l'adresse IP **dynamiquement**
    - Le DHCP server: Un **block** (plage) d'adresses IP
    - Le DHCP server choisit une des adresses en répondant au client
  - Permet au client d'obtenir tous les paramètres de configuration dans un seul message
  - **DHCP** est basé sur le **BootP message format** et utilise le **BootP relay agent**.
  - BootP clients peuvent interopérer avec les DHCP servers.

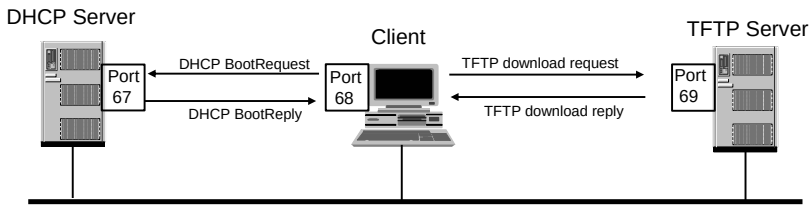
48

Dynamic Host Configuration Protocol (DHCP)

- DHCP attribution d'adresses
  - **Allocation Manuelle**: l'Administrateur entre une IP adresse dans le serveur qui est attribuée de façon permanente au client
  - **Allocation Automatique**: une IP adresse est sélectionnée et attribuée de façon permanente dans l' **address pool** au client quand la machine est attachée la première fois au réseau
  - **Allocation dynamique**: une IP address est attribuée ou loué au client pour une période limitée

49

Dynamic Host Configuration Protocol (DHCP)

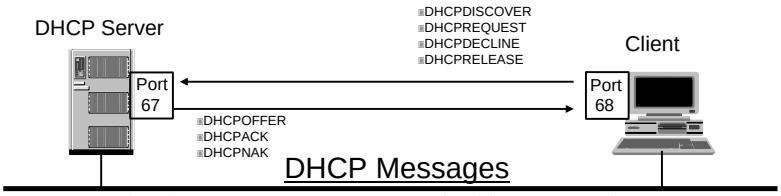


DHCP Concept

Même méthode que le BOOTP avec une spécification de la durée de bail

50

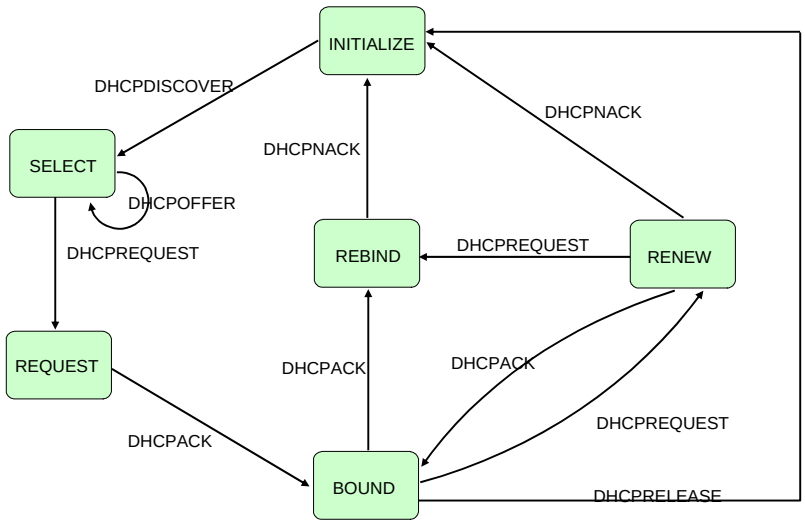
Dynamic Host Configuration Protocol (DHCP)



- ***DHCPDISCOVER***: le client envoie un message de découverte des serveurs qui peuvent le fournir une adresse et les paramètres désirés. Ce message peut inclure les valeurs suggérées pour l'adresse du réseau et la durée de bail. **Relay agents** peuvent passer ce message aux DHCP servers.
- ***DHCPOFFER*** : Serveurs répondent au message client par des offres d'adresses
- ***DHCPREQUEST***: le client sélectionne un serveur et envoie une demande une sends **IP address** et, optionnellement, une ***DHCP Parameter Request List***
- ***DHCPACK***: Le serveur créé un attachement et répond avec les paramètres de configuration demandés. L'adresse de matériel et de réseau de clients identifient uniquement le bail
- ***DHCPNAK***:Le serveur refuse la demande et le client recommence une nouvelle demande
- ***DHCPDECLINE***: Le client refuse les paramètres de configuration parce qu'au moins un est invalide
- ***DHCPRELEASE***: Le client libère l'adresse IP

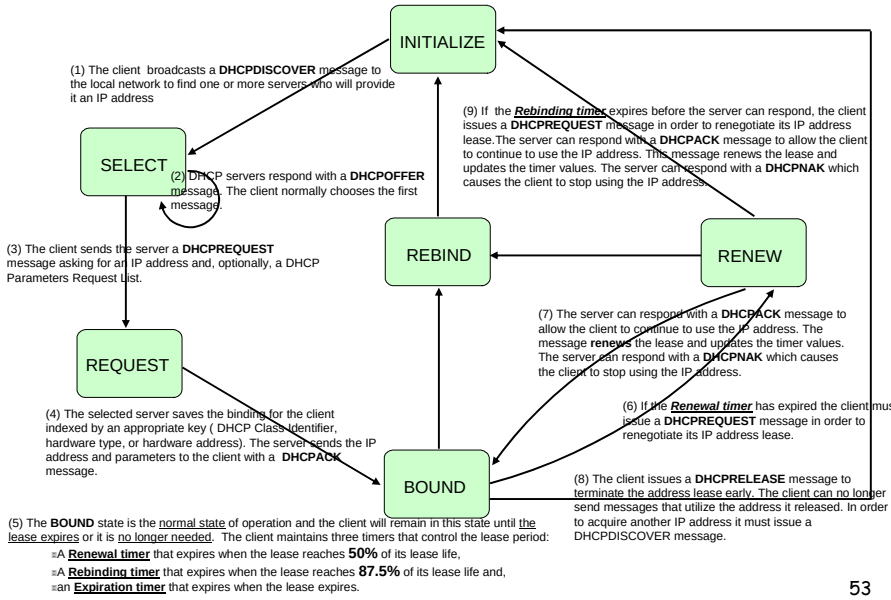
51

Dynamic Host Configuration Protocol (DHCP)



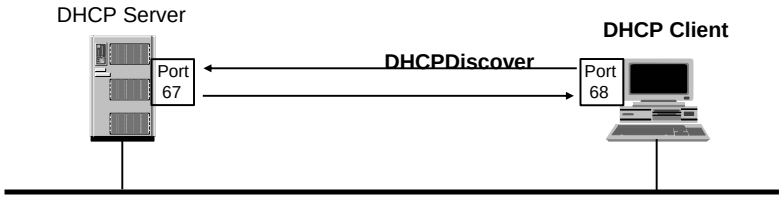
52

Dynamic Host Configuration Protocol (DHCP)



53

DHCP Client

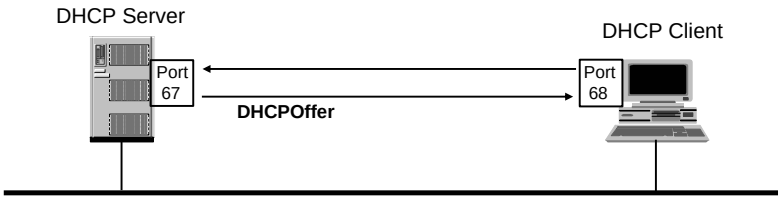


- DHCP client peut être configuré pour l'**automatic DHCP configuration**.
- DHCP client accomplit cette tache en 5 phases (p1à p5):

P1. Le **DHCP client broadcasts** un **DHCPDISCOVER** message pour demander l'adresse IP au DHCP serveur.
  - Le message est broadcast avec des paramètres suivants: **DIP=255.255.255.255**, **SIP = 0.0.0.0**, **clients hardware address.**
  - La demande peut être reçue par le DHCP server sur le même **subnet** ou
  - Peut être **forwardeé** par le **BootP relay** aux autres sous réseaux

54

DHCP Client

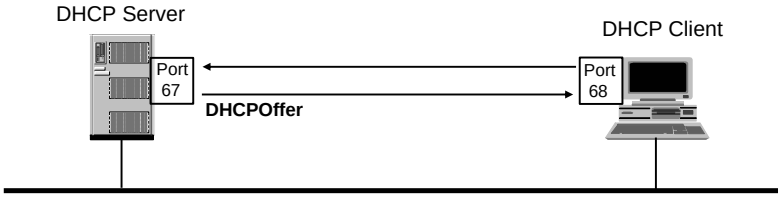


P2. Tous les **DHCP serveurs** reçoivent la demande et répondent au client avec un **broadcast DHCPOFFER**.

- **local et remote DHCP serveurs** répondent avec un message **DHCPOFFER**
  - Le DHCP server sur le même sous réseau doivent répondre en premier
  - L'offre de DHCP serveur distant est accepté s'il n'y a pas d'offre du DHCP server local
- L'offre contient une **client IP address**, **subnet mask**, **default gateway**, **lease duration** et **IP address options**.
- L' offre est envoyé en broadcast au **client's hardware address** si le client fait partie du même sous réseau
- **unicast** au **BootP Relay Agent** qui sera envoyé en Broadcast/Unicast au client

55

DHCP Client

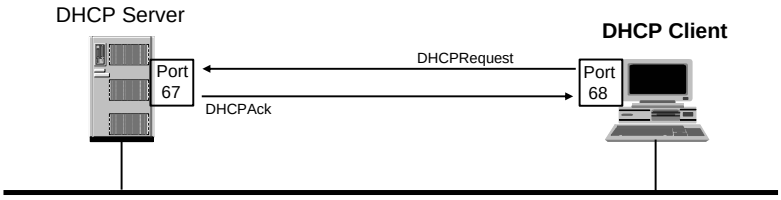


P2.

- Les DHCP servers marquent leurs IP adresses déjà allouées dans leurs **databases**
- Si le client ne reçoit pas un DHCPOFFER dans une seconde il rebroadcasts le message DHCPDISCOVER 4 fois pendant 60 secondes (Microsoft spécifie des intervalles ,10 secondes, 23 secondes, 39 secondes et 5 minutes) à chaque fois il recommence le processus de configuration
- S'il n'y a aucune réponse, le client informe l'utilisateur de l'échec

56

DHCP Client

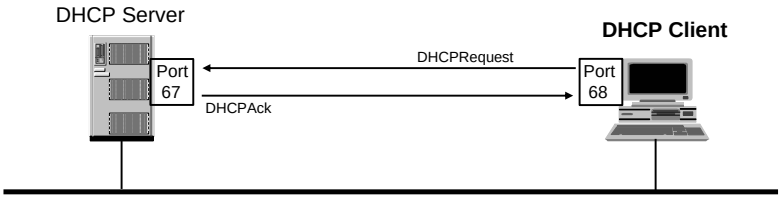


P3. **DHCP client** sélectionne le premier offre reçu et **broadcasts** un message **DHCPREQUEST** au serveur DHCP sélectionné

- DHCP serveur est identifié dans le champ option "server identifier"
- Si DHCP serveur est sur un réseau différent, le **BootP Relay Agent** a la responsabilité pour relayer le message au DHCP serveur via **IP unicast**.

57

DHCP Client

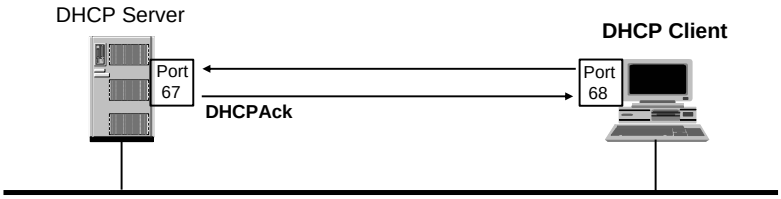


P4. DHCP serveur qui a offert le bail répond avec un message **de DHCPACK** au client qui reconnaît le bail d'IP

- DHCP serveur **updates** son database de bail pour indiquer que l'adresse IP ne sera pas disponible

58

DHCP Client



P5. Le client reçoit le DHCPACK et **valide** l'adresse IP à travers le protocole **ARP**

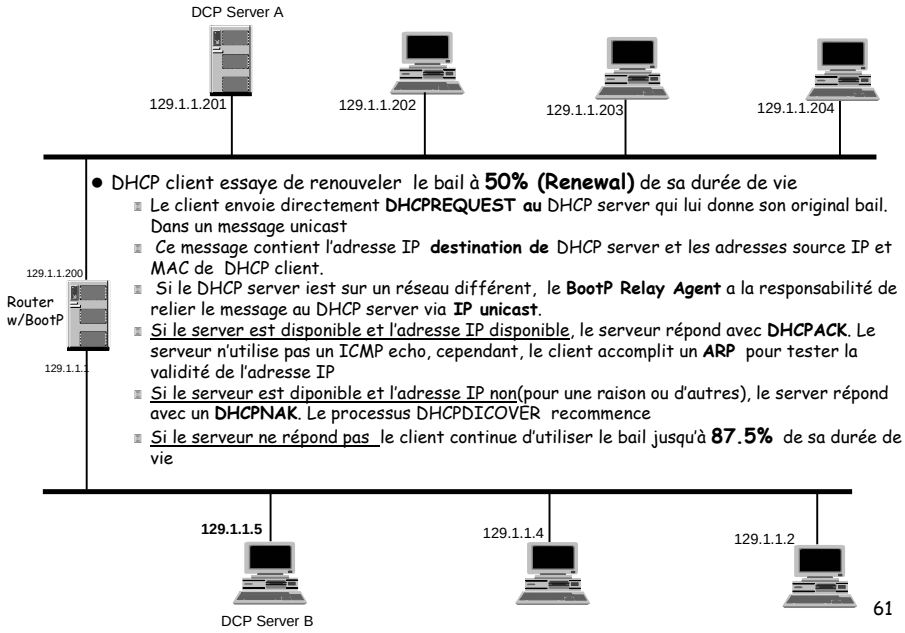
- Le client envoie un **ARP Request** avec le **SMAC address** = son adresse physique, **SIP** = 0.0.0.0, **DMAC** address = 1s et **DIP address** = à l'adresse IP reçue de DHCP server.
- Si IP address est en utilisation, le client envoie un message **DHCPDECLINE** au serveur OU
- Si IP address n'est pas utilisée le client broadcasts un **ARP Reply** pour annoncer sa nouvelle adresse et effacer les entrées précédentes dans le cache ARP
- Il met alors à jour son registre et continue le processus d'initialisation

59

Démonstration

60

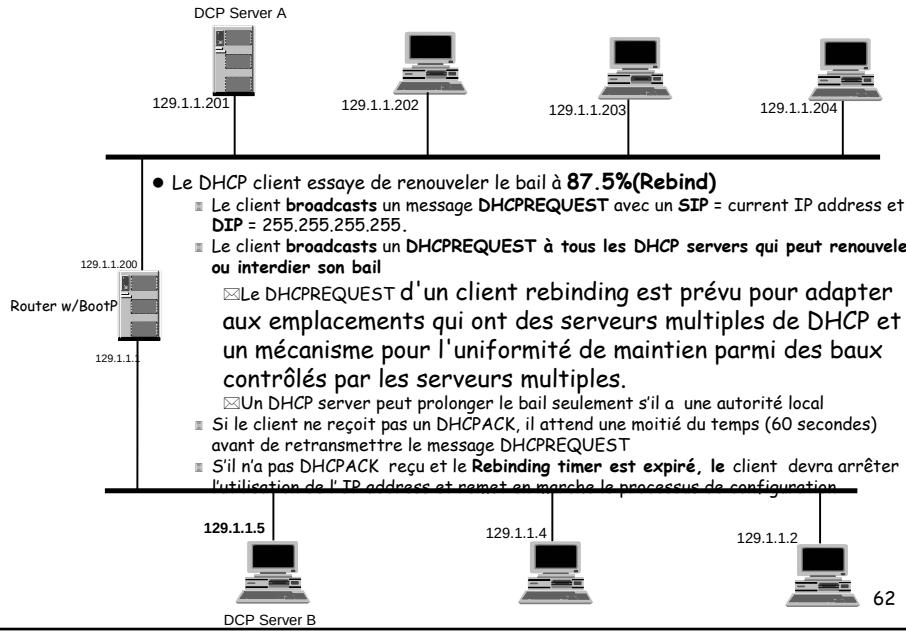
DHCP Client Address Renewal



- DHCP client essaye de renouveler le bail à **50% (Renewal)** de sa durée de vie
  - Le client envoie directement **DHCPREQUEST au** DHCP server qui lui donne son original bail. Dans un message unicast
  - Ce message contient l'adresse IP **destination de** DHCP server et les adresses source IP et MAC de DHCP client.
  - Si le DHCP server iest sur un réseau différent, le **BootP Relay Agent** a la responsabilité de relier le message au DHCP server via **IP unicast**.
  - Si le server est disponible et l'adresse IP disponible, le serveur répond avec **DHCPACK**. Le serveur n'utilise pas un ICMP echo, cependant, le client accomplit un **ARP** pour tester la validité de l'adresse IP
  - Si le serveur est diponible et l'adresse IP non(pour une raison ou d'autres), le server répond avec un **DHCPNAK**. Le processus DHCPDICOVER recommence
  - Si le serveur ne répond pas le client continue d'utiliser le bail jusqu'à **87.5%** de sa durée de vie

61

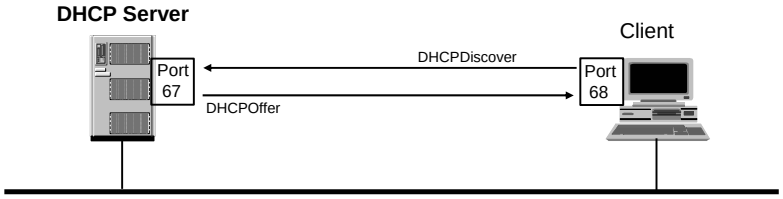
DHCP Client Address Rebinding



- Le DHCP client essaye de renouveler le bail à **87.5%(Rebind)**
  - Le client **broadcasts** un message **DHCPREQUEST** avec un **SIP** = current IP address et **DIP** = 255.255.255.255.
  - Le client **broadcasts** un **DHCPREQUEST à tous les DHCP servers qui peut renouveler ou interdier son bail**
    - Le DHCPREQUEST d'un client rebinding est prévu pour adapter aux emplacements qui ont des serveurs multiples de DHCP et un mécanisme pour l'uniformité de maintien parmi des baux contrôlés par les serveurs multiples.
    - Un DHCP server peut prolonger le bail seulement s'il a une autorité local
  - Si le client ne reçoit pas un DHCPACK, il attend une moitié du temps (60 secondes) avant de retransmettre le message DHCPREQUEST
  - S'il n'a pas DHCPACK reçu et le **Rebinding timer est expiré, le** client devra arrêter l'utilisation de l' IP address et remet en marche le processus de configuration

62

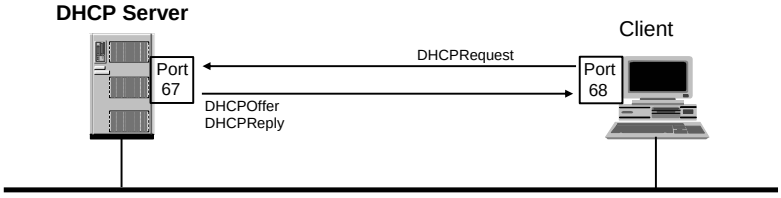
DHCP Server



- Il y a normallement un DHCP par sous réseau. Un seul **DHCP server**, cependant peut couvrir plusieurs sous réseaux
- Le DHCP Server peut être configuré manuellement avec son **IP address**, **Subnet Mask** et **default gateway**.
- **Un serveur peut gérer deux plages différentes d'adresses pour deux sous réseaux**
  - ✉ Les deux plages doivent être mutuellement exclusif pour empêcher l'affectation d'adresses double (à moins qu'il y a une méthode pour maintenir l'uniformité de location).
  - ✉ L'adresse sous réseau **client déterminera quelle plage est employée en faisant une lovation d'une adresse**
  - ✉Chaque plage est configurée avec des options comme le mask sous réseau, **Default Route**r, **DNS Server**, **DNS Domain Name** et l'adresse IP de bail
    - Le bail est par défaut de trois heures
    - Le bail n'a pas une durée **minimale,** cependant peut varier entre une heure à l'infini
- DHCP Server peut être configurer pour :
  - **Exclure les adresses IP d'une plage**
  - **Réserver des adresses** IP pour des clients DHCP spécifiés

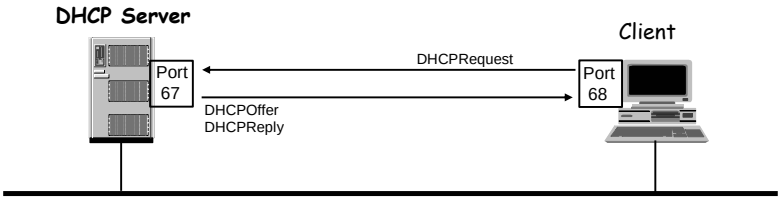
63

DHCP Server



- DHCP server reçoit les messages sur le **UDP port 67** et les traitent comme suite:
  - Un **DHCPOFFER** et un **DHCREPLY sont créés de la manière suivante**:
    - ✉L' **IP address** est normalement attribué comme suivant:
      - ➔ Le client a un **CURRENT** binding, ELSE
      - ➔ Le client a un **PREVIOUS** binding, ELSE
      - ➔ L address dans le champ "**Requested IP Address**" (si valide et non allouée), ELSE
      - ➔ Une nouvelle **adresse** sélectionnée dans la plage d'adresses basé sur le sous réseau (si le relay agent =0s) ou par l'adresse d'interface du **Relay Agent** (si le champ relay agent différent de 0s).
  - Le DHCP server enregistre l'adresse comme étant offerte au client et n'utilise pas l'IP adresse jusqu'à la réception de la réponse du client
    - ✉Le bail de **IP address est attribué comme suite**:
      - ➔ Si le client n'a pas demandé un bail et le client a une adresse attribuée,THEN
      - ➔ Si le client n'a pas demandé un bail et le client a une adresse attribuée,THEN
      - ➔ Si le client a demandé un spécifique bail, THEN
      - ➔ Le server retourne un bail sélectionné ou un autre bail

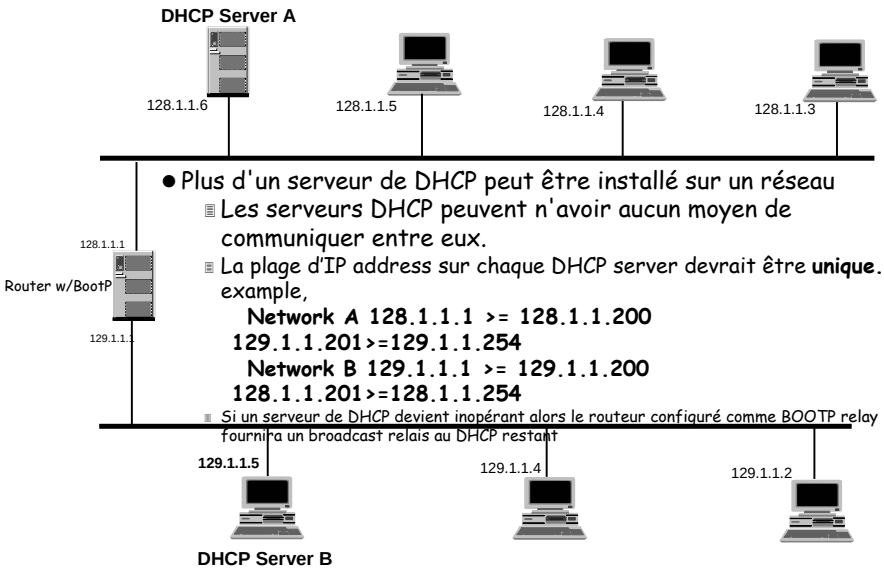
DHCP Server



- Le DHCP server retourne le même champ "**Transaction ID**"
- Le message est retourné au client comme suite:
- Si le champ "**client IP address**" est **non-zero**, THEN
  - Le message est envoyé dans un **IP unicast** au l'IP adresse identifiée dans "**client IP address**", ELSE
- Si le champ "**relay agent**" est **non-zero**, THEN
  - Le message est envoyé dans **IP unicast** à l'adrese IP identifiée dans le champ "**relay agent**" avec destination **port 67**(cette action délivre le message directement au BootP relay agent le plus proche au client), ELSE
- Si le champ "**relay agent**" est **zero** (le client réside sur le même sous réseau que le BootP relayagent) et si le **Broadcast Flag** = **1**, THEN
  - le message est **broadcast** au **DIP** = **255.255.255.255** et **DMAC** = **Broadcast** avec un **UDP port** = **68**, ELSE
- If le **Broadcast Flag** = **0**, THEN
  - le message est envoyé en IP **unicast** au **DIP** = **"your IP address"** et **DMAC** = **"client hardware address"** avec un **UDP port** = **68**

65

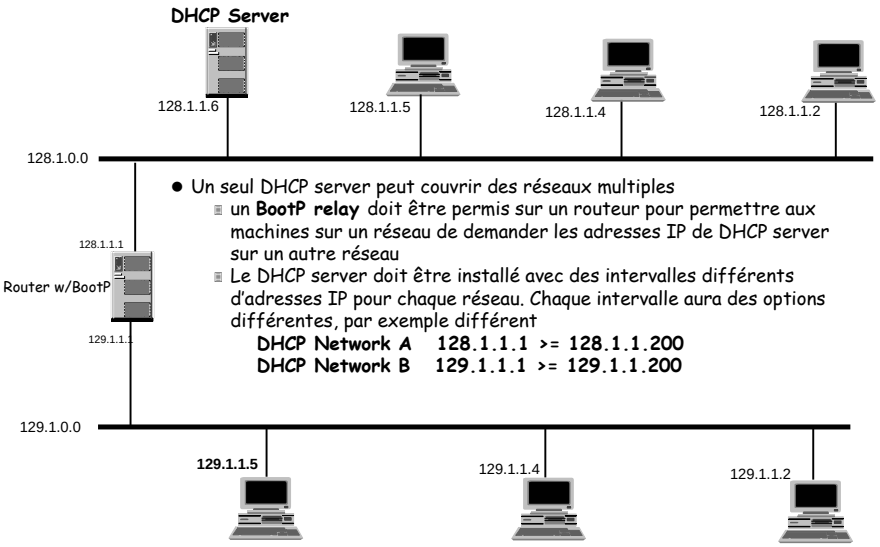
Multiple DHCP Servers



- Plus d'un serveur de DHCP peut être installé sur un réseau
  - Les serveurs DHCP peuvent n'avoir aucun moyen de communiquer entre eux.
  - La plage d'IP address sur chaque DHCP server devrait être **unique**. example,
    Network A 128.1.1.1 >= 128.1.1.200
    129.1.1.201>=129.1.1.254
    Network B 129.1.1.1 >= 129.1.1.200
    128.1.1.201>=128.1.1.254
  - Si un serveur de DHCP devient inopérant alors le routeur configuré comme BOOTP relay fournira un broadcast relais au DHCP restant

66

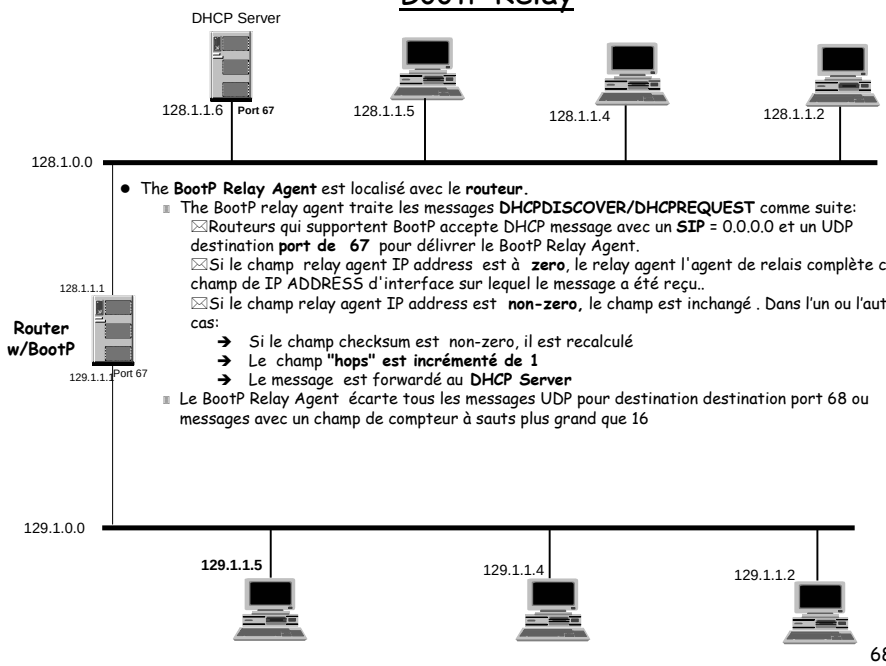
Single DHCP Server w/Multiple Networks



- Un seul DHCP server peut couvrir des réseaux multiples
  - un **BootP relay** doit être permis sur un routeur pour permettre aux machines sur un réseau de demander les adresses IP de DHCP server sur un autre réseau
  - Le DHCP server doit être installé avec des intervalles différents d'adresses IP pour chaque réseau. Chaque intervalle aura des options différentes, par exemple différent
    DHCP Network A 128.1.1.1 >= 128.1.1.200
    DHCP Network B 129.1.1.1 >= 129.1.1.200

67

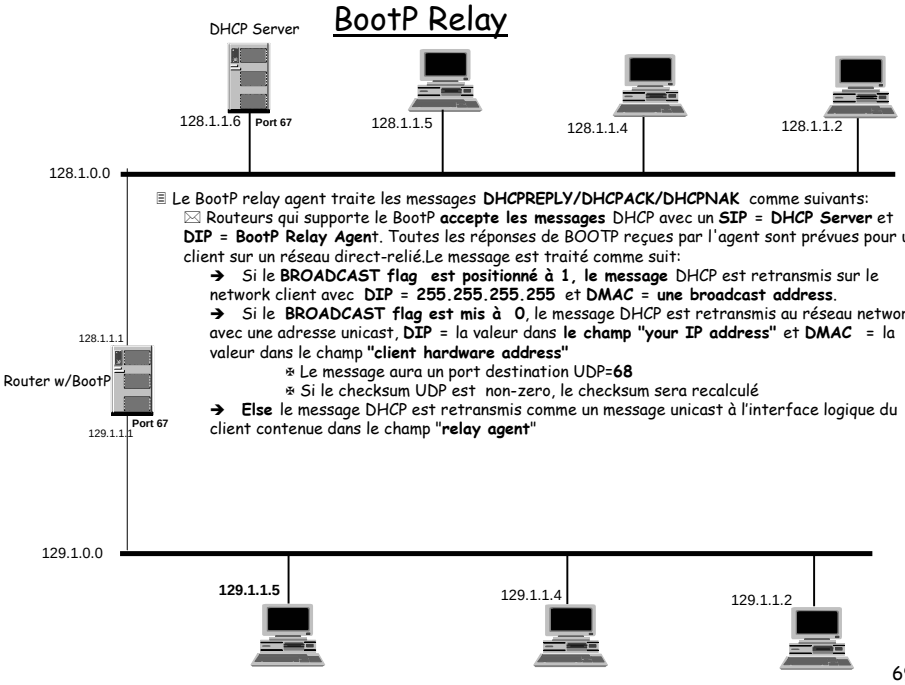
BootP Relay



- The **BootP Relay Agent** est localisé avec le **routeur**.
  - The BootP relay agent traite les messages **DHCPDISCOVER/DHCPREQUEST** comme suite:
    - Routeurs qui supportent BootP accepte DHCP message avec un **SIP** = 0.0.0.0 et un UDP destination **port de 67** pour délivrer le BootP Relay Agent.
    - Si le champ relay agent IP address est à **zero**, le relay agent l'agent de relais complète ce champ de IP ADDRESS d'interface sur lequel le message a été reçu..
    - Si le champ relay agent IP address est **non-zero**, le champ est inchangé . Dans l'un ou l'autre cas:
      - ➔ Si le champ checksum est non-zero, il est recalculé
      - ➔ Le champ **"hops" est incrémenté de 1**
      - ➔ Le message est forwardé au **DHCP Server**
  - Le BootP Relay Agent écarte tous les messages UDP pour destination destination port 68 ou messages avec un champ de compteur à sauts plus grand que 16

68

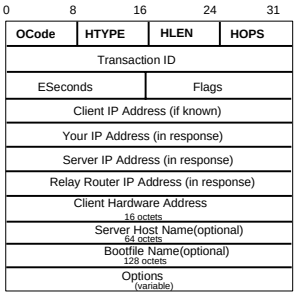
BootP Relay



- Le BootP relay agent traite les messages **DHCPREPLY/DHCPACK/DHCPNAK** comme suivants:
  - Routeurs qui supporte le BootP **accepte les messages** DHCP avec un **SIP** = **DHCP Server** et **DIP** = **BootP Relay Agen**t. Toutes les réponses de BOOTP reçues par l'agent sont prévues pour un client sur un réseau direct-relié.Le message est traité comme suit:
    - ➔ Si le **BROADCAST flag est positionné à 1, le message** DHCP est retransmis sur le network client avec **DIP** = **255.255.255.255** et **DMAC** = **une broadcast address**.
    - ➔ Si le **BROADCAST flag est mis à 0**, le message DHCP est retransmis au réseau network avec une adresse unicast, **DIP** = la valeur dans **le champ "your IP address"** et **DMAC** = la valeur dans le champ **"client hardware address"**
      - Le message aura un port destination UDP=**68**
      - Si le checksum UDP est non-zero, le checksum sera recalculé
    - ➔ **Else** le message DHCP est retransmis comme un message unicast à l'interface logique du client contenue dans le champ "**relay agent**"

69

Dynamic Host Configuration Protocol (DHCP)



- Identique au BOOTP

70

Cours 5 : Plan

5.1 Principes des protocoles de la couche Applications
5.2 DNS
5.3 Electronic Mail
- SMTP, POP3, IMAP

5.4 DHCP/BOOTP
5.5 NFS
5.6 Web et HTTP
5.7 FTP
5.8 Telnet/Rlogin
5.9 SNMP

71

NFS : Network File System

- Partage de fichiers à travers un réseau de machines et de systèmes hétérogènes (VAX-VMS, PC-MSDOS et de nombreux UNI-NFS)
- Accès aux fichiers distants masqué aux utilisateurs et aux programmes
- Performances des accès voisines de celles d'un disque local
- Résistance aux pannes: serveur, client et réseau
- Extensibilité du système de fichiers simple
- Facilité d'administration (NIS. Network Information, ex. YP)

72

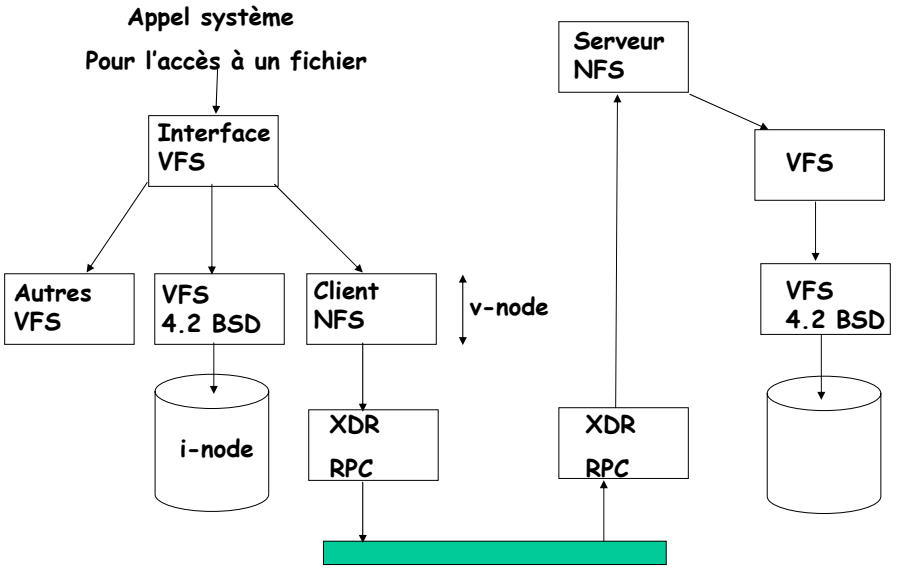
NFS : Network File System

- Point de vue réseau:
  - Utilisation d'UDP
  - Conçu sur le modèle client/serveur, à travers un RPC
  - Gestion de l'hétérogénéité par XDR

- Point de vue Système:
  - Réécriture de l'interface d'E/S avec le noyau UNIX: concept de Virtual File System et virtual-node
  - Protocole NFS et serveur sans état
  - Protocoles périphériques et serveur avec état: **Mount** (montage d'arborescences), **NIS** (gestion des paramètres), **Lock Manager** (Gestion des verrous) et **Network Status** (Surveillance du réseau)

73

NFS : Network File System



74

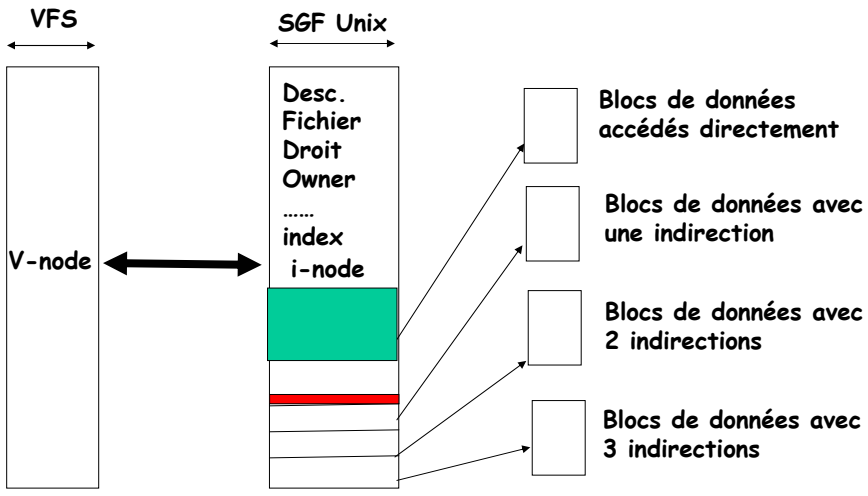
NFS : Network File System

- VFS: couche logique, interface qui masque l'accès à un sous-arbre local ou distant de l'arborescence des fichiers ou partition ("File System" au sens logique)
  - il est construit sur le concept de V-node
  - il gère le montage et le démontage de partitions
  - il permet l'accès aux fichiers lors de traversées de points d'attachement ou points de montage ("mount point")
- V-node: Couche logique, interface qui généralise la notion de descripteur de fichier (i-node Unix), et qui sépare les opérations d'accès à un fichier de leur implantation sur disque

75

NFS : Network File System

- Notion de descripteur virtuel de fichier



76

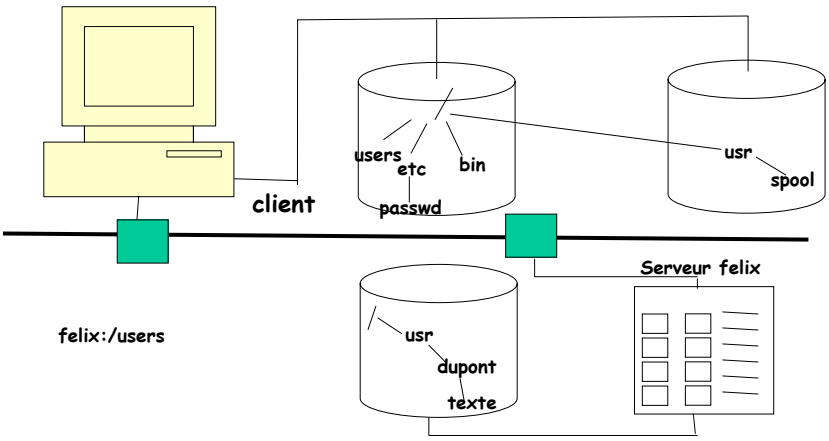
NFS : Network File System

- Notion de file system:
  - Vue administration système
    + partition logique du disque physique qui contient une sous arborescence des fichiers du système (ex. Sd0a, première partition d'un disque)
    + une autre partition à un rôle important(ex. Deuxième partition sd0b et sd1b: pagination et swapping)
  - Vue implantation
    + un file system est désigné par un no de périphérique logique et contient un Boot bloc, des infos. De gestion, super-bloc, une suite de descripteurs de fichiers ou de répertoire (i-node: un numéro unique dans un système de fichiers)

77

NFS : Network File System

- Montage de fichiers distants:
  - la liste des répertoires attachables exportés par le serveur est décrit dans le fichier /etc/exports < nom de répertoire, liste d'accès des utilisateurs et machines autorisées>
  - le nom d'un répertoire à attacher ne correspond pas nécessairement à une sous arborescence complète et donc à une partition locale du serveur



78

NFS : Network File System

- Le serveur NFS est sans état. Il ne maintient aucune information sur les fichiers qu'il gère pour le compte d'un client.
- Le client conserve toutes les informations qui permettent au serveur de retrouver le fichier
- Résistance aux pannes: quand un serveur tombe en panne, les clients restent bloqués jusqu'à la remise en route du serveur (mécanisme RPC avec temporisation 1100s entre deux tentatives avant un message d'erreur)
- Le protocole NFS est lui aussi réalisé à l'aide de procédure de type RPC Sun:
  GETATTR(rend les attributs d'un fichier), READFILE(crée un fichier), WRITE(écrit dans le fichier), ….
- Les identificateurs d'utilisateurs et de groupes d'utilisateurs entre machines clientes et serveurs doivent être identiques sinon il risque d'y avoir des problèmes de propriétés et de droits d'accès
- Les accès concurrents ne sont pas supportés par NFS, c'est le dernier qui écrit qui a gagné
- Synchronisation des horloges sur les différentes machines (ex: pour que la commande make fonctionne correctement)

79

NFS : Network File System

- Les clients NFS peuvent utiliser une technique de cache pour améliorer les performances (cache biod)
- Le cache réside en mémoire centrale du client, le disque local s'il existe n'est pas impliqué dans la gestion
- Les informations mises dans le cache sont:
  - page contenu de fichier
  - page contenu de répertoire
  - descripteur de fichier
- Problème de validation du cache en mémoire centrale:
  - les pages sont estampillées avec leur date de dernière modification. Il y a validation par comparaison de cette date avec celle qui réside avec le fichier sur le serveur, s'il y a une différence, la page doit être rechargée.
  - Quand cette comparaison peut-elle s'effectuer? **À l'initiative du client**
  - Pour un fichier:
    à chaque ouverture de fichier
    à chaque défaut de page dans le cache
  - Pour un répertoire:
    à chaque ouverture de fichier, il y a contrôle de validité pour le répertoire qui le contient
    les répertoires sont conservés dans le cache en lecture seulement, les modifications sont opérées sur le serveur directement, (validation périodique toutes les 3s pour un fichier et toutes les 30s pour un répertoire)

80

NFS : Network File System

- Mise à jour du serveur:
  - une page modifiée est marquée "sale" et devra être transférée vers le serveur
  - cette activité est réalisée de façon asynchrone par le noyau… quand il a le temps
  - garantie:
  + toutes les pages modifiées seront recopiées au plus tard avant la fermeture du fichier (technique"write-on-close")
  + la gestion de la concurrence est sous la responsabilité de l'utilisateur…. À défaut, c'est le dernier qui écrit qui a gagné
  - transfert serveur-client:
  + par blocs de 8 Ko
  + tout le fichier pour les petits fichiers pas plus grands que 8 Ko

81

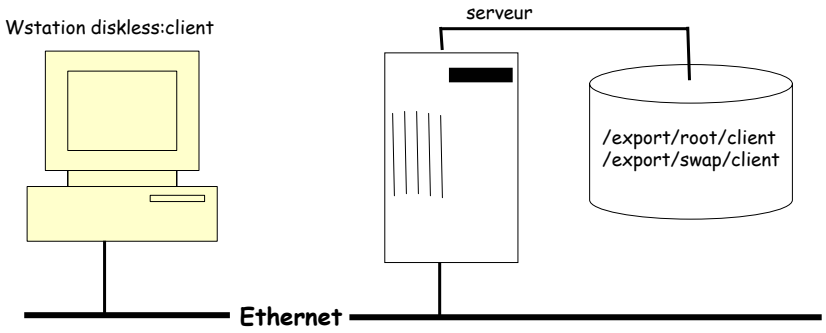
NFS : Network File System

- Réplication
  - Permet d'avoir plusieurs serveurs pour réaliser un attachement de sous-arbre.
  - Le premier serveur qui répond au client demandeur qui est utilisé
  - Pratique pour les fichiers exécutables et les fichiers de données accédés en lecture (tolérance aux pannes pour ces fichiers)
  - Pour les fichiers modifiés, la propagation des écritures d'un serveur à un autre doit être faite "à la main"

82

NFS : Network File System

- Les stations sans disque: utilisent NFS pour leur partition système (fichiers généraux, bibliothèques, compilateurs, commandes, espace utilisateur et aussi la partition swap.



83

Cours 5 : Plan

5.1 Principes des protocoles de la couche Applications
5.2 DNS
5.3 Electronic Mail
- SMTP, POP3, IMAP

5.4 DHCP/BOOTP
5.5 NFS
5.6 Web et HTTP
5.7 FTP
5.8 Telnet/Rlogin
5.9 SNMP

84

Web et HTTP

- Web page composée des objets
- Objet peut être fichier HTML, image JPEG, applet Java, fichier audio,…
- Web page composée de base HTML-file qui inclut plusieurs objets référencés
- Chaque objet est adressé par un URL
- Exemple URL:

`www.someschool.edu/someDept/pic.gif`

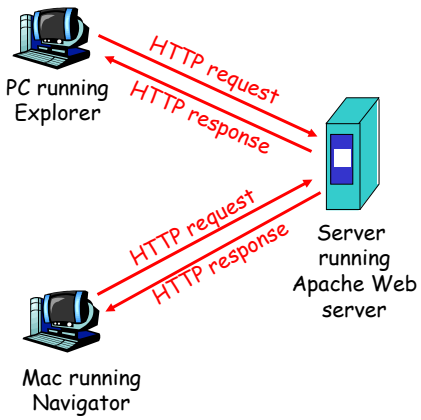
host name | path name

85

HTTP

HTTP: hypertext transfer protocol

- Protocole de la couche application Web
- Modèle client/server
  - *client:* browser qui demande et reçoit , "displays" les objets Web
  - *server:* Web serveur envoie les objets demandés
- HTTP 1.0: RFC 1945
- HTTP 1.1: RFC 2068



86

HTTP

Utilise TCP:

- client initialiseTCP connection (crée une socket) au serveur, port 80
- serveur accepte TCP connection de client
- messages HTTP (application-layer protocol messages) échangés entre browser (HTTP client) et Web server (HTTP server)
- TCP ferme la connection

HTTP protocole sans mémoire

- serveur ne maintient pas l'information concernant les demandes passées du client

Protocoles avec mémoire sans complexes!

- Historique (state) doit être maintenu
- si server/client crashent, on garde une image de l'historique pour re-établir l'état d'avant

87

HTTP connections

Non-persistant HTTP

- Un seul objet Web peut être transféré à la fois sur une connexion TCP
- HTTP/1.0 utilise des connexions non persistantes

Persistant HTTP

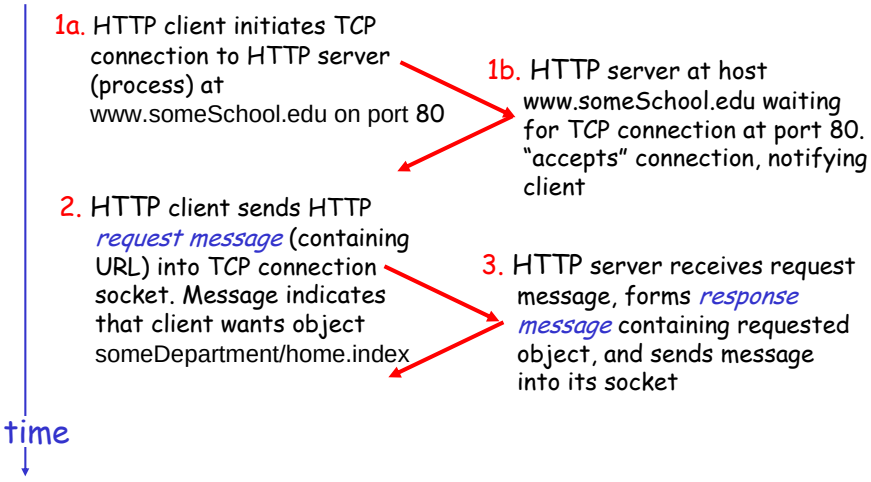
- Multiple objets peuvent emprunter la même connexion TCP
- HTTP/1.1 utilise les connexions persistants par défaut avec pipelinage

88

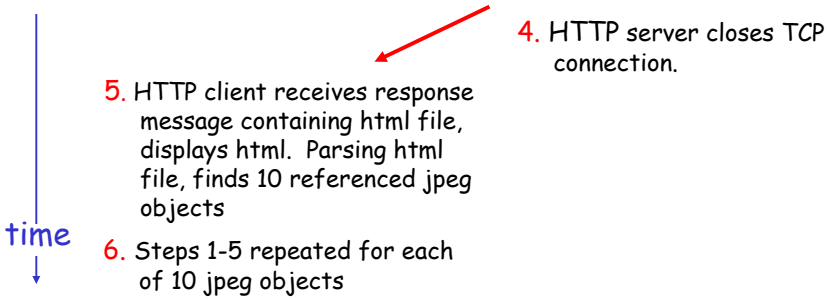
Non persistant HTTP

Supposons l'uitlisateur entre l' URL suivant:
www.someSchool.edu/someDepartment/home.index

(contient text, référence 10 Image jpeg)



Non persistant HTTP



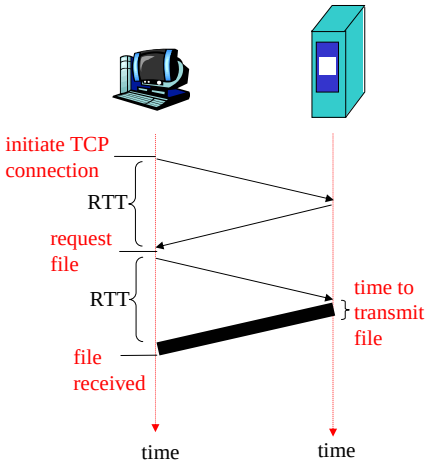
Modélisation du temps de réponse

Définition de RRT: temps d'envoyer un petit paquet d'aller au serveur et retourne au client

Temps de réponse:

- un RTT pour initialiser la connection TCP
- un RTT pour la demande HTTP et les premiers octets de HTTP de réponse
- Temps de transmission de fichier

total = 2RTT+transmit time



91

Persistant HTTP

HTTP non persistant demande:

- 2 RTTs par objet
- Le sysstème devrait travailler et alloue des ressources pour chaqueTCP connection
- mais browser souvent ouvre des connections parallèles pour traiter des objets référencés

Persistant HTTP

- serveur maintient la connexion ouvert après l'envoi de la réponse
- les messages successifs, entre le même couple client/server sont envoyés sur la connection

Persistant sans pipelinage:

- client envoie une nouvelle demande seulement quand il reçoit une réponse de la requête précédente
- un RTT pour chaque objet référencé

Persistant avec pipelinage:

- Par défaut dans HTTP/1.1
- client envoie des requêtes dés qu'il rencontre une référence objet
- un seul RTT pour tous les objets référencés

92

HTTP: message de demande

- deux types de messages HTTP: *demande, réponse*
- HTTP message de demande:
  - ASCII (format humain lisible)

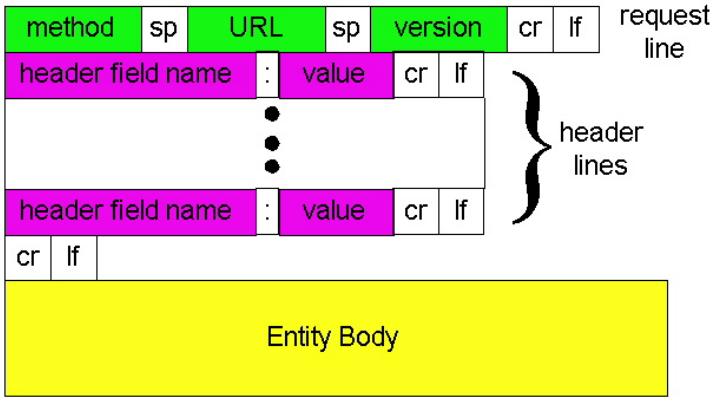
Ligne de demande (GET, POST, HEAD commands)

lignes d'en-tête

```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
User-agent: Mozilla/4.0
Connection: close
Accept-language:fr
```

93

HTTP demande de message: format général



94

Chargement de contenu

Méthode POST:

- page Web souvent inclut la forme de contenu
- Le contenu est chargé de serveur dans la squelette "body"

Méthode URL:

- Utilise la méthode GET
- Le contenu est chargé dans le champ URL de la ligne de demande:

www.somesite.com/animalsearch?monkeys&banana

95

Types de méthodes

HTTP/1.0

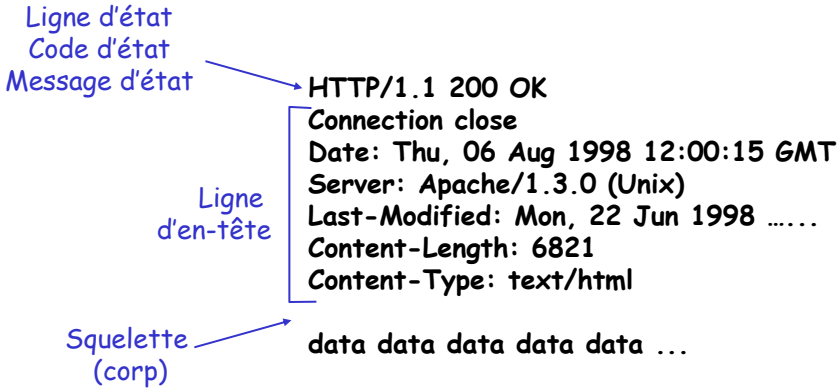
- GET
- POST
- HEAD
  - Demande au serveur de quitter l'objet demandé après la réponse

HTTP/1.1

- GET, POST, HEAD
- PUT
  - Permet le chargement d'un objet vers un chemin spécifié dans le champ URL
- DELETE
  - Permet d'effacer un objet (fichier) hébergé sur un serveur web

96

HTTP: message de réponse



97

HTTP: codes d'état

Dans la première ligne dans le message de réponse server->client

quelques codes d'état:

200 OK
- la requête et réussie et l'information est contenue dans la réponse

301 Moved Permanently
- l'objet sollicité a été définitivement déplacé, une nouvelle URL est spécifiée dans la ligne d'en-tête (Location:)

400 Bad Request
- demande de message n'a pas été comprise par le serveur

404 Not Found
- document recherché est introuvable sur le serveur

505 HTTP Version Not Supported

98

Exemple de message de réponse

1. Telnet sur votre serveur Web favoris:

telnet www.eurecom.fr 80

Opens TCP connection to port 80 (default HTTP server port) at www.eurecom.fr. Anything typed in sent to port 80 at www.eurecom.fr

2. Taper la ligne suivante:

GET /~login/index.html HTTP/1.0

By typing this in (hit carriage return twice), you send this minimal (but complete) GET request to HTTP server

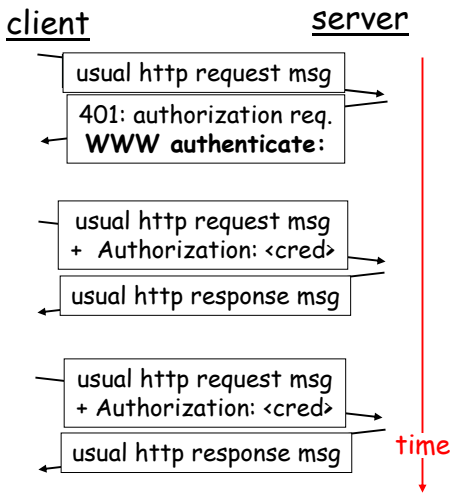
3. Observer la réponse du serveur!

Interaction Utilisateur-serveur: autorisation

Autorisation : contrôle d' accès au contenu de serveur

- authorization credentials: typically name, password
- stateless: client doit s'authentifier à chaque demande
  - autorisation: ligne d'en-tête dans chaque demande
  - S'il n'a pas d' autorisation: le serveur refuse l' accès,

    Le serveur ajoute l'en-tête WWW-authentificate:



Cookies

Plusieurs sites Web utilisent les cookies

4 étapes suivantes pour réaliser les cookies:

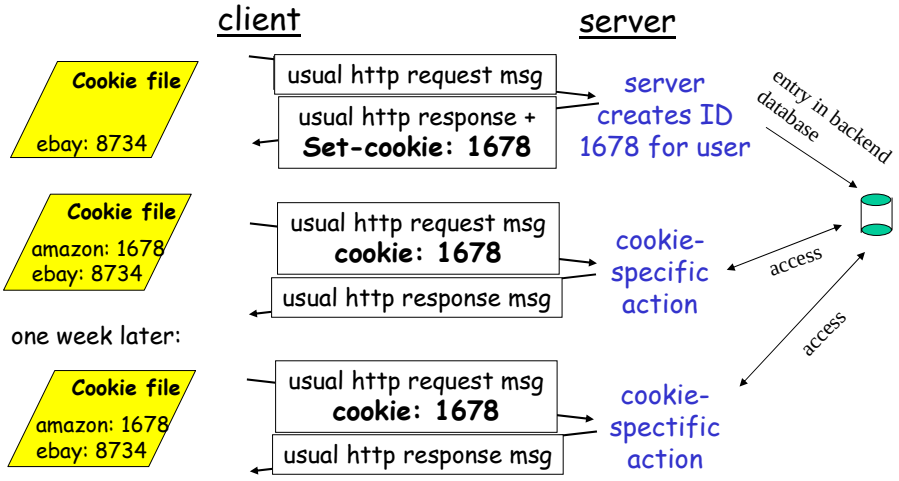
1) l'insertion d'une ligne d'en-tête particulière dans le message de réponse HTTP
2) la réitération du message de demande avec la ligne d'en-tête correspondante
3) l'envoi d'un fichier témoin qui est conservé dans le poste de l'utilisateur et est activé par le navigateur
4) l'intégration des informations dans une base de données située sur le serveur du site Web

Exemple:

- Susan accède Internet toujours à partir de même PC
- Elle visite un site spécifié au commerce électronique pour la première fois
- Quand les demandes initiales HTTP arrivent au site, le site crée un unique ID et crée une entrée dans la database pour ID

101

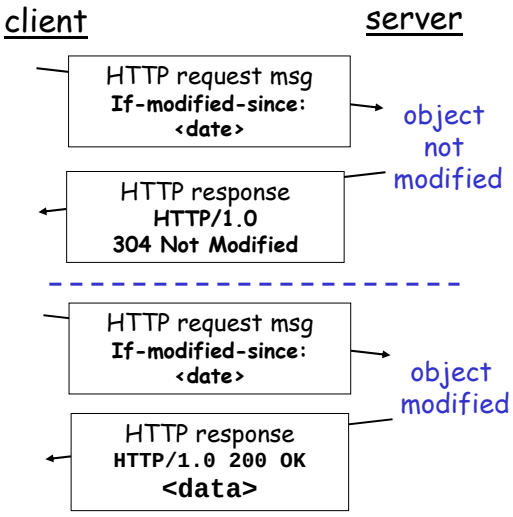
Cookies exemple



102

Conditional GET: client-side caching

- Goal: enregistrer les objets déjà demandés une fois
- client: spécifie la date de mettre une copie en cache dans la demande HTTP
  `If-modified-since: <date>`
- server: la réponse de serveur ne contient pas l'objet si sa mise à jour n'est pas modifiée:
  `HTTP/1.0 304 Not Modified`



103

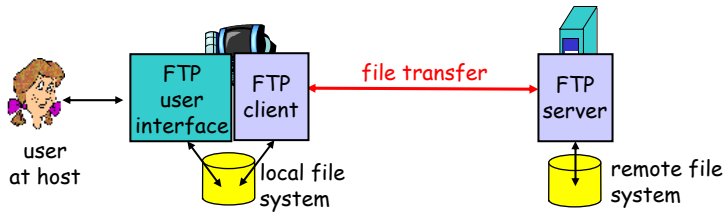
Cours 5 : Plan

5.1 Principes des protocoles de la couche Applications
5.2 DNS
5.3 Electronic Mail
- SMTP, POP3, IMAP

5.4 DHCP/BOOTP
5.5 NFS
5.6 Web et HTTP
5.7 FTP
5.8 Telnet/Rlogin
5.9 SNMP

104

FTP: the file transfer protocol

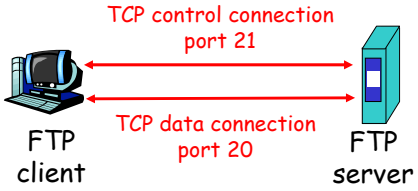


- Fichier de transfert de/à host distante
- modèle client/server
  - *client:* partie qui initialise le transfert
  - *server:* host distante
- ftp: RFC 959
- ftp serveur: port 21

105

FTP: sépare les connexions de contrôle et de données

- FTP client contacte FTP server au port 21, spécifiant TCP comme protocole de transport
- Client obtient l' autorisation sur la connexion de contrôle
- Client navigue dans le répertoire distant en envoyant les commandes sur la connexion de contrôle
- Quand le serveur reçoit une commande pour un transfert de fichier, le serveur ouvre une connexion de données TCP au client
- Après le transfert d'un fichier, le serveur ferme la connection



- Serveur ouvre une seconde connection de données TCP pour transférer un autre fichier
- Connection de contrôle: "hors bande"
- FTP serveur garde trace des changements dans le répertoire tout le long de son exploration du répertoire distant

106

FTP: commandes, réponses

Simples commandes:

- Format as ASCII à 7 bits sur la connexion de contrôle
- **USER** ***username***
- **PASS** ***password***
- **LIST** demande au serveur de transmettre une liste de tous les fichiers contenus dans le répertoire distant actuel
- **RETR filename** = gets (obtenir un fichier à partir du répertoire distant)
- **STOR filename** stocke (puts) le fichier dans le répertoire distant

Simples codes de retour

- Code d'état et phrase (comme dans HTTP)
- **331 Username OK, password required**
- **125 data connection already open; transfer starting**
- **425 Can't open data connection**
- **452 Error writing file**

107

Cours 5 : Plan

5.1 Principes des protocoles de la couche Applications

5.2 DNS

5.3 Electronic Mail

- SMTP, POP3, IMAP

5.4 DHCP/BOOTP

5.5 NFS

5.6 Web et HTTP

5.7 FTP

5.8 Telnet/Rlogin

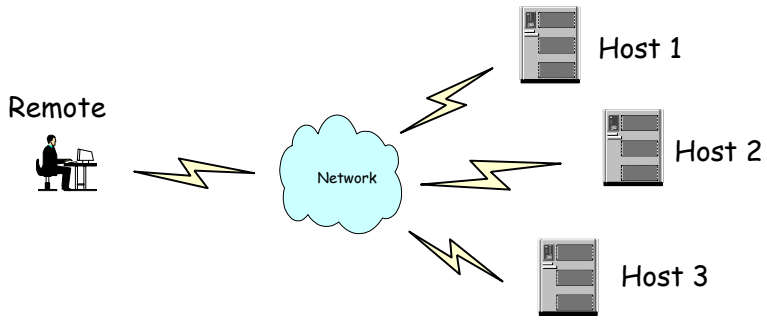
5.9 SNMP

108

TELECOMMUNICATION NETWORK PROTOCOL - TELNET-

109

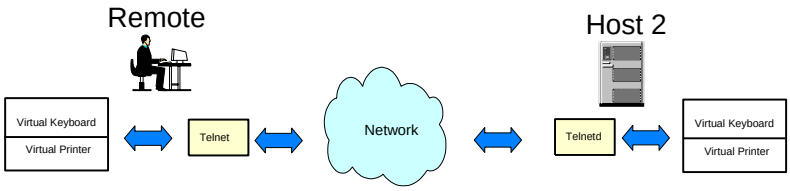
TELNET Background



- Au début de l'Internet il y avait une diversité, les terminaux et les serveurs qui étaient incompatibles.
- Les machines distantes gèrent la translation du code terminal
- Plusieurs connections peuvent consommer les ressources de la machine distante à fin de réaliser cette translation
- **Telecommunication Network Protocol (TELNET)** était développé pour résoudre ce problème
  - The **Network Virtual Terminal (NVT)** était défini comme une interface standard aux systèmes distants

110

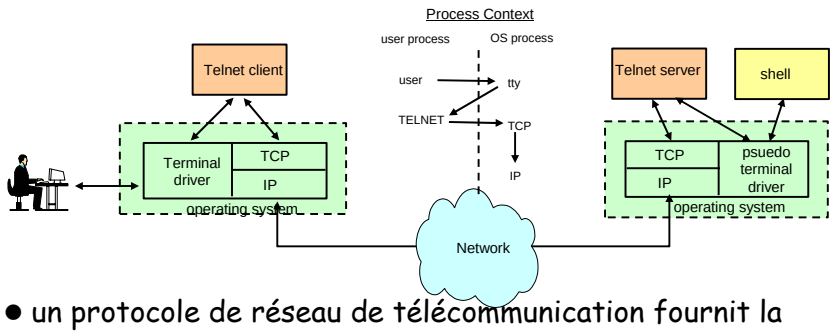
TELNET Background



- **Network Virtual Terminal (NVT)** définit tous les terminaux de la connexion Telnet
  - C'est un dispositif imaginaire ayant une structure de bas commune pour émuler des terminaux réels
- NVT est composé de:
  - **Virtual Keyboard** (un clavier réel) pour produire des caractères
  - **Virtual Printer** (un écran réel) qui affiche les caractères
- Les programmes **(telnet** et **telnetd)** contrôlent et gèrent la translation des instructions des terminaux virtuels sur les dispositifs matériels ou physiques
- Des fonctions supportées sont négociées à l'initialisation d'une connexion Telnet
  - Ceci permet à Telnet de **translater (traduire)** les commandes entre NVT et les machines d'extrémité.

111

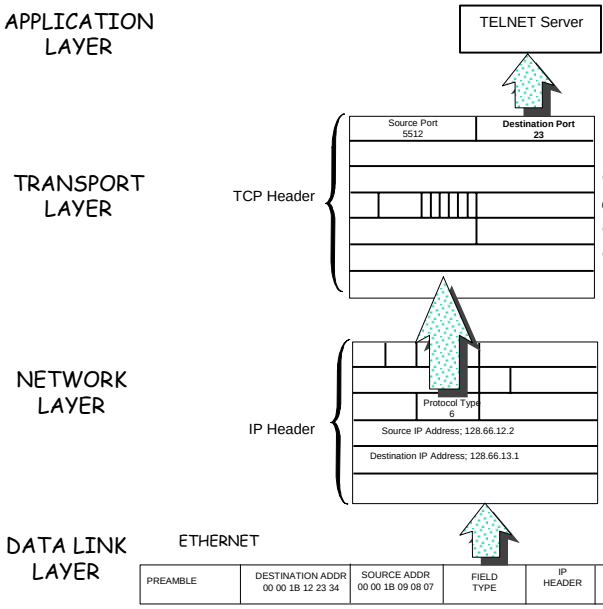
TELNET CONCEPT



- un protocole de réseau de télécommunication fournit la possibilité d'accéder à distance (**remote login**)
- TELNET est conçu pour fonctionner sur des machines qui utilisent des systèmes différents
- TELNET est un service **encapsulé** dans TCP
- TELNET définit un a langage commun:
  - **N**etwork **V**irtual **T**erminal(**NVT**) - pour gérer le transfert des commandes et des données à travers le réseau

112

PROTOCOL ENCAPSULATION and TELNET PORT NUMBER

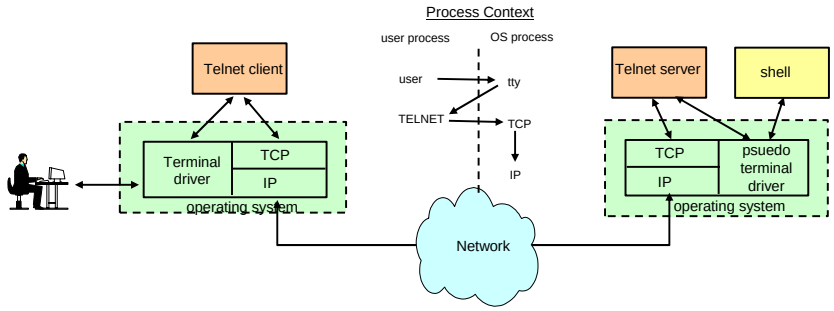


NOTE

- TELNET uses **TCP** for encapsulation.
- **Port 23** is used for TELNET.
- TELNET to **any port**.

113

TELNET SERVICES de base



Service 1 : Network Virtual Terminal (NVT): fournit un dispositif terminal imaginaire pour émuler TELNET **(client/serveur)**

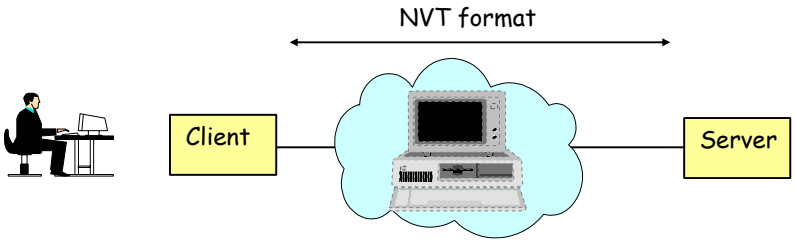
Service 2 : Option Negotiation: un mécanisme par lequel le client et le server **négocient** des options utilisées pendant une session Telnet

- Le client/server utilisent un ensemble de conventions pour établir les caractéristiques opérationnelles de leur session via **"DO, DON'T, WILL, WON'T"** mécanismes.

Service 3 : Opération Symétrique: les deux extrémités d'une session telnet sont traitées de façon symétrique (l'une ou l'autre extrémité peut commencer la négociation)

114

Network Virtual Terminal(NVT)



- **Network Virtual Terminal (NVT)** fournit une interface standard aux systèmes distants
- NVT est un imaginaire dispositif avec un **keyboard** et **printer**. Caractères saisis par l'utilisateur sont convertis au NVT caractères.
  - Le client **translate** le format client vers le format NVT.
  - Le NVT est transmis à travers le réseau
  - Le server **translate** le format NTV au format de la machine serveur.
- le données sont envoyées caractère par caractère mais elles peuvent être envoyées par ligne
- Chaque ligne est terminée par un CR/LF
- Telnet est en **Half-duplex**.
  - Le client envoie une ligne et attend la réponse.
  - Le server envoie une ligne, alors un envoi d'une commande ***Go Ahead*** indiquant au client qui peut envoyer des données

115

TELNET COMMANDS

- NVT utilise des mots de 8 bits, telnet utilise des mots de 7 bits pour les caractères et des mots de 8 bits pour les commandes
  - **FTP, SMTP, Finger** et **Whois** utilisent NVT
- Les commandes TELNET sont composées d'une séquence de trois octets dont un optionnel
  - Le **premier octet** est toujours interprété comme un caractère de commande **Interpret as Command(IAC)**
  - Le **second octet** est le **code commande.**
  - Le **troisième octet** est utilisé pour **négocier les options.**

TELNET Command Structure

Interpret as Command	Command Code	Negotiated Options
byte 1	byte 2	byte 3

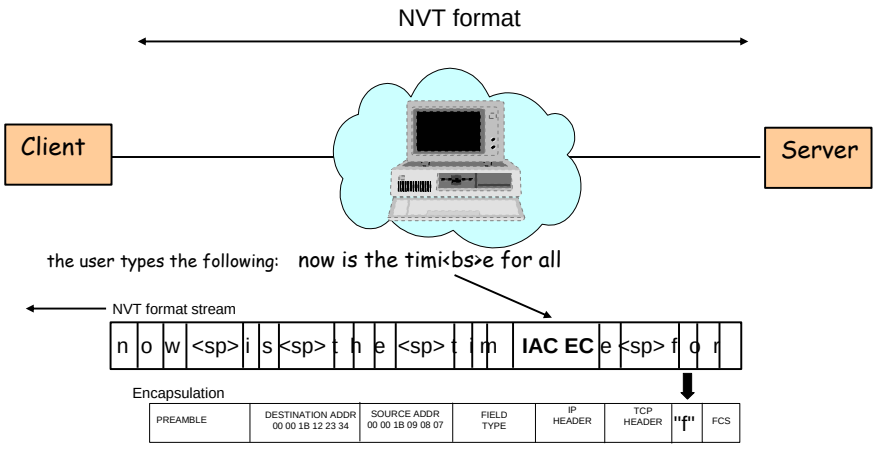
116

TELNET COMMANDS

Code	Name	Description
255	**IAC**	interpret next octet as command
255 236	EOF	end-of-file
255 237	SUSP	suspend current operation
255 238	ABORT	abort process
255 239	EOR	end of record
255 240	**SE**	end of subnegotiation parameters
255 241	NOP	no operation
255 242	DM	data stream portion of the TCP Synch signal that is always accompanied by a TCP Urgent Notification flag.
255 243	BRK	"break control" signal
255 244	IP	"Interrupt process" control signal
255 245	AO	"abort output" control signal
255 246	AYT	"are you there?" control signal
255 247	EC	"erase character" control signal
255 248	EL	"erase line" control signal
255 249	GA	"go ahead" control signal
255 250	**SB**	start of subnegotiation of indicated option
255 251	WILL	**sender** wants to **enable** option
255 252	WON'T	**sender** wants to **disable** option
255 253	DO	**sender** wants **receiver** to **enable** option
255 254	DON'T	**sender** wants **receiver** to **disable** option

17

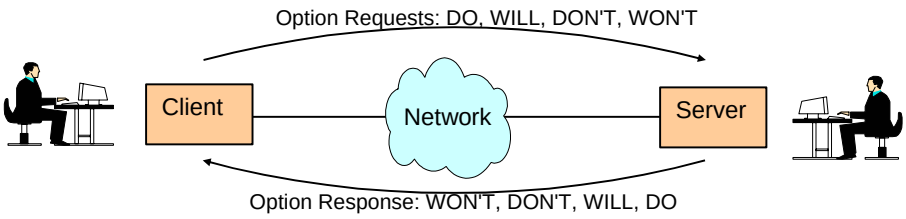
NVT Example



- Les fonctions de contrôle et de service passées en tant qu'élément de données sont précédés par le caractère échappe suivi du code "IAC"
- plusieurs implémentations TELNET placent un caractère dans chaque paquet TCP à moins que le **LINEMODE** est permis.

118

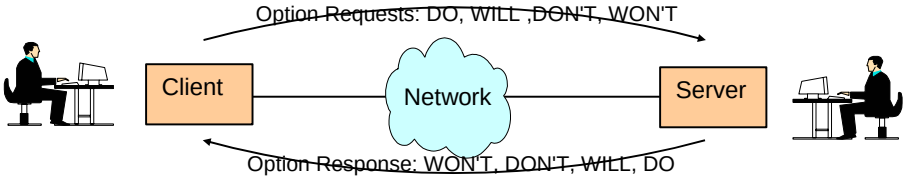
TELNET Negotiations



- Le client et le server exigent de supporter un **basic NVT implémentation.**
- Le client et le server négocient **lowest NVT implementation level.**
- Lors du premier échange, les options sont négociées
  - Il y a plus de **40 différent codes d'option** disponibles.
  - Multiple options peuvent apparaître dans un seul paquet TCP

119

TELNET Negotiations



	Sender	Receiver	Description
1.	WILL	→	**sender** wants to **enable** option
		← DO	receiver says **OK**
2.	WILL	→	**sender** wants to **enable** option
		← DON"T	receiver says **NO**
3.	DO	→	sender wants **receiver** to **enable** option
		← WILL	receiver says **OK**
4.	DO	→	sender wants **receiver** to **enable** option
		← WON'T	receiver says **NO**
5.	WON't	→	**sender** wants to **disable** option
		← DON't	receiver says **OK**
6.	DON't	→	sender wants **receiver** to **disable** option
		← WON't	receiver says **OK**

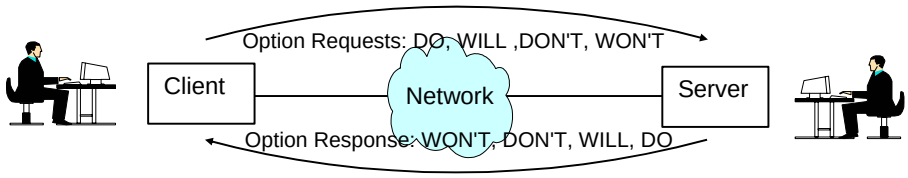
120

Common TELNET Options

Code	Name	Description
0	Transmit Binary	Transmit in 8-bit binary code
1	Echo	Allow the receiver to echo the data
3	SGA	Suppress sending the "Go Ahead(GA)" signal at data end
4	Message Size	Negotiate approximate message size
5	Status	Request for status of TELNET option at data end
6	Timing Mark	Request that a timing mark be inserted in the return stream for synch
8	Line Width	Negotiate output line width
9	Page Length	Negotiate page length
11	Horizontal Tabs	Negotiate output horizontal tab stop settings
14	Vertical Tabs	Negotiate output vertical tab stop settings
17	Extended ASCII	Negotiate extended ASCII characters
24	**Terminal Type**	Exchange information about the terminal type make/model
31	NAWS	Negotiate about window size
32	TSpeed	Send terminal speed information
33	TFC	Terminal(remote) flow control
34	**Linemode**	Send complete lines instead of individual characters
37	Authentication	Negotiate type of Authentication
38	Encryption	Negotiate type of Encryption

121

TELNET Suboption Negotiations Example



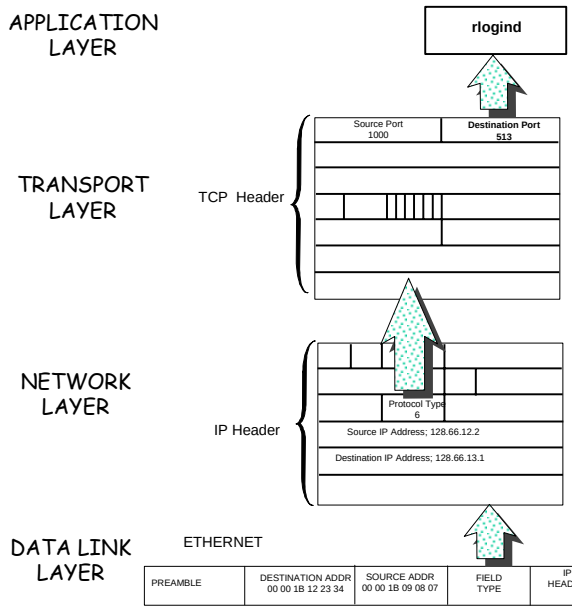
Client	Server
<IAC,DO,SGA>	
	<IAC,WILL,SGA>
<255,251,24>	
	<255,253,24>
<255,250,24,1,255,240>	
	<255,250,24,0,'I','B','M','P','C',255,240>

122

REMOTE LOGIN
- RLOGIN-

123

PROTOCOL ENCAPSULATION et Rlogin PORT NUMBER



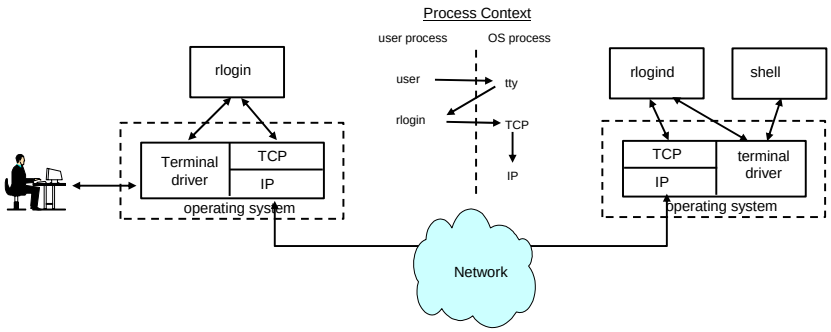
NOTE

- Rlogin utilise TCP pour l'encapsulation.
- **Port 513** est utilisé pour *rlogin*.
  - **Port 514** est utilisé pour *rsh*, *rcmd*,etc.
- Le client devrait utiliser un port au dessous de 1023.

.

124

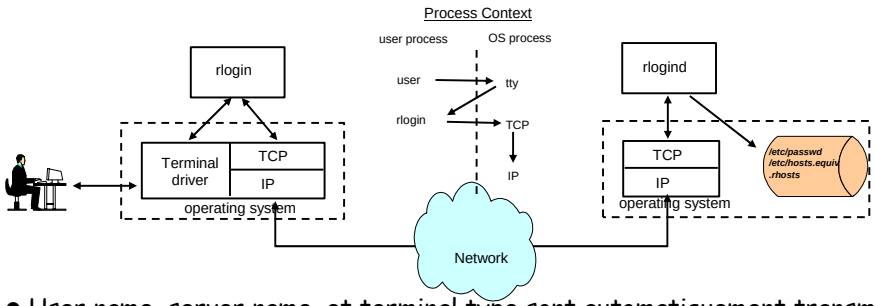
Rlogin CONCEPT



- Rlogin était désigné de **fonctionner sur les systèmes Unix** seulement. Il est plus simple que TELNET, la négociation des options n'est pas exigé
- ***rlogin*** agit en tant que client tandis que ***rlogind*** agit en tant que server

125

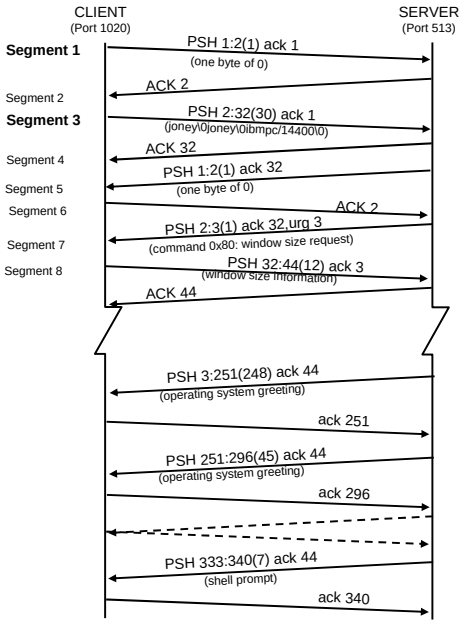
Rlogin CONCEPT



- User name, server name, et terminal type sont automatiquement transmis au début de la connection.
- The server laisse l'accès libre à l'utilisateur si la connexion vient d'un serveur de confiance
  - le fichier ***/etc/passwd*** est cherché sur le serveur. S'il n'est pas trouvé alors
  - le fichier ***/etc/hosts.equiv*** sur le serveur est scanné (il contient une liste des machines sures sur le serveur), s'il n'est pas trouvé alors
  - le fichier ***.rhosts*** sur le serveur est scanné, s'il n'est pas trouvé alors
  - un mot de passe en clair est exigé au client

126

Rlogin CONNECTION



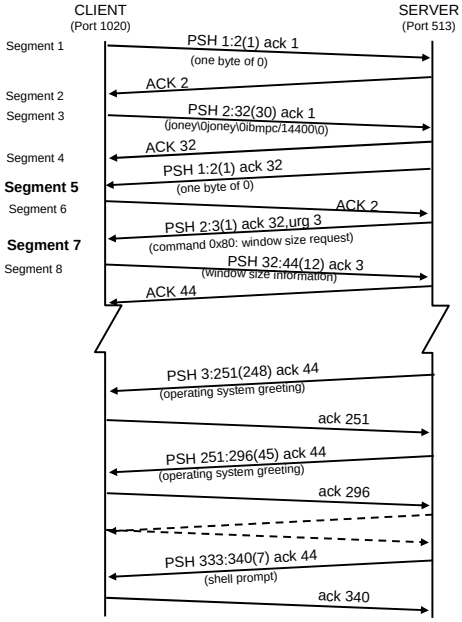
NOTE: This sequence does not show the opening, closing, window size or mss, In addition, only selected segments are described.

Segment 1 shows the client on **port 1020** (below 1023) sending a PSH segment with an **Initial Sequence(ISN)** Number of 1, a colon, and an **implied ending sequence** number of 2 followed by the number of data bytes sent and an ack to the client. The client sends a byte of 0.

Segment 3 shows the client sending a **PSH** segment with an Initial Sequence Number of 2, a colon, and an implied ending sequence number of 32 followed by the number of data bytes sent and an ack. The client sends a:
(a) **login name of the user on the client host,** a termination byte of 0,
(b) **the login name of the user on the server host**, a termination byte of 0,
(c) t**he user terminal type,** a slash,
(d) **the terminal speed**, and
(e) a final termination of 0.

127

Rlogin CONNECTION



Segment 5 shows the server on sending a PSH segment with an ISN of 1, a colon, and an IESN of 2 followed by the number of data bytes sent and an ack to the client.

NOTE: Since he may be logging in from a trusted host, the user does not need a password.

- The server checks for a user password, in order, the ***/etc/passwd*** file, the ***/etc/hosts*** file then the ***.rhosts*** file.
- The server then has the option of asking the client for a password.
- The password is sent in the clear.
- If no password is sent within a specified time, normally 60 seconds, the connection is closed.

Segment 7 shows the server sending a command of 0x80 asking for the **clients window size**. The server to client commands are as follows:

0x02: Flush the output. The server receives an interrupt and sends the command to discard all received data.
0x10: Client stops performing flow control
0x20: Client resumes flow control
0x80: Request for window size.

128

TELNET and Rlogin Features

Feature	Rlogin	Telnet
Transport Protocol	One TCP connection. Uses Urg mode.	One TCP connection. Uses Urg mode.
Packet Mode	Character at a time w/remote echo	Common default is character at a time w/remote echo
Flow Control	Normally by client, disabled by server.	Normally by server, option for client.
Terminal Type	Always Provided.	Option, commonly supported.
Terminal Speed	Always Provided.	Option.
Window Size	Option supported by most servers.	Option.
Environ Var	Not supported	Option.
Automatic Login	Default. A prompted password is sent cleartext. Newer versions support Kerberos.	Default is login name and password. Password is sent cleartext. Newer versions provide authentication option.

Cours 5 : Plan

5.1 Principes des protocoles de la couche Applications
5.2 DNS
5.3 Electronic Mail
- SMTP, POP3, IMAP

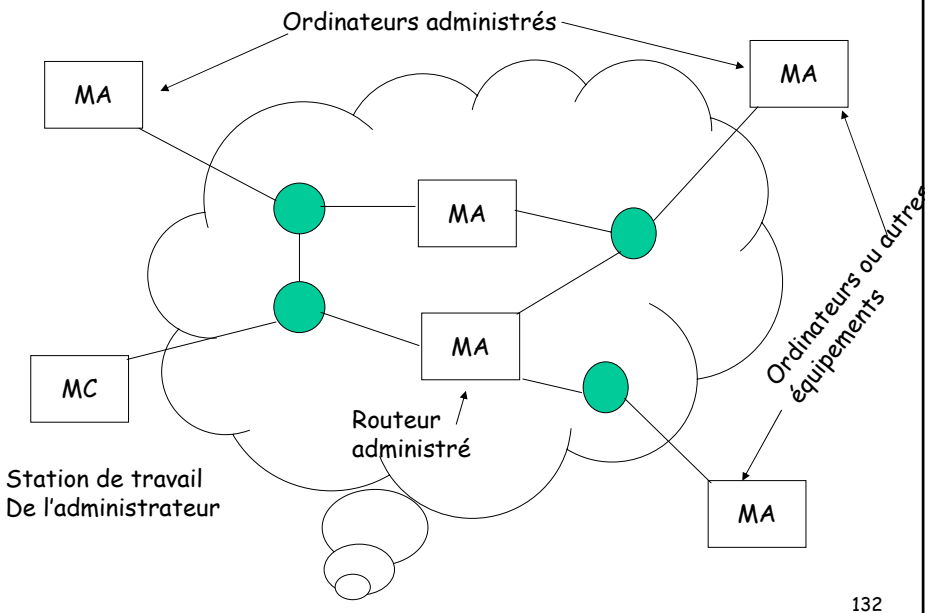
5.4 DHCP/BOOTP
5.5 NFS
5.6 Web et HTTP
5.7 FTP
5.8 Telnet/Rlogin
5.9 SNMP

SNMP(Simple Network Management Protocol) :introduction

- Permet de traiter les problèmes:
  - fonctionnement
  - Contrôle le routage
  - Signalisation des machines qui ont des comportements anormaux
- L'ensemble de ces activités correspond à l'administration de réseau
- Niveau d'action des protocoles d'administration de réseaux
  - Au niveau WAN
  - Les routeurs sont des commutateurs actifs que les administrateurs doivent surveiller et contrôler

131

SNMP :modèle Architectural



132

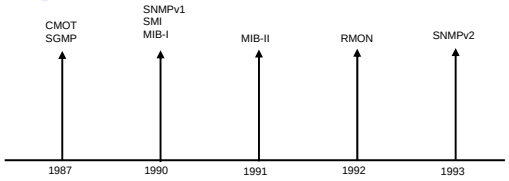
SNMP: Cadre architectural

- deux parties
  - Les données prises en compte
- Les échanges d'information
  - Les échanges d'information
  - Définir la façon d'exécution le programme client sur la machine de l'administrateur
- Les données prises en compte
  - Les éléments de données qu'un routeur doit conserver , leur nom et leur syntaxe de représentation

- Versions SNMP: 3 versions SNMP
- Définition standard des informations d'administration
  - Doit fournir des statistiques des machines
  - Pas de détails sur les données associées le rôle de la base de données MIB (Management Information Base)

133

SNMP: généralités



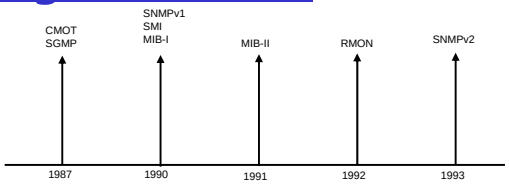
- **CMOT(Common Management Information Service and Protocol over TCP/IP)** désigné d'administrer pour une période longue TCP/IP et il était développé en parallèle avec SNMP.

- **SNMPv1** est le standard d'administration . Il était créée par Internet pour administrer pour une période courte et temporaireTCP/IP. L'architecture SNMP était basée plus tard sur un protocole d'administration appelé **SGMP(Simple Gateway Monitoring Protocol)**.

- **SMI(Structure of Management Information)** est utilisé pour définir les structures de donnée de SNMP. C'est un mécanisme de nommage et d'organisation des objets pour être administrés par SNMP.

134

SNMP: généralités



- **MIB-I (Management Information Base)** stocke l'information sur chaque objet SNMP administré. Il définit un nombre minimal d'objets pour être administré par chaque noeud. Il y a huit catégories avec un total de 114 objets.

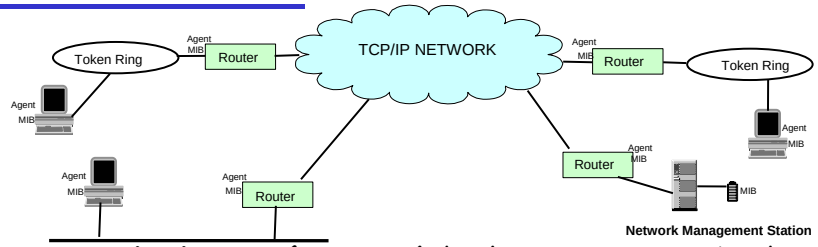
- **MIB-II** est une version améliorée de MIB-1 en ajoutant deux nouvelles catégories, nouvelles valeurs, variables, tables, colonnes,etc pour améliorer le support pour le matériel multiprotocol.

- **RMON(Remote Network Monitoring)** est un utilitaire d'administration pour observer le trafic sur les liens dans l'ordre d'établir le trafic, statistiques sur les performances, etc.

- **SNMPv2** essaye d'améliorer et résoudre les problèmes de SNMPv1.

135

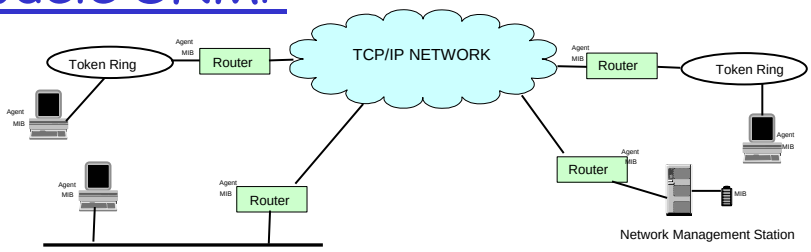
Modèle SNMP



- Un **noeud administré** composé des hosts, routeurs, bridges, imprimantes, etc capable de communiquer l'**information d'état** au **Network Management Station(NMS).**
- A **Network Management Station** contient **manager software d'administration** qui envoie et reçoit les messages aux **Agents** residants dans les noeudes.
- l' **Agent** est un software réside dans le noeud et répond au questions NMS, accomlplit les maj.
- le **Management Information Base** réside dans les Noeuds et le NMS et est une logique description de tous les **network management data.** Il contient le système et l'information d'état, données de performance, et les paramètres de configuration.

136

Modèle SNMP

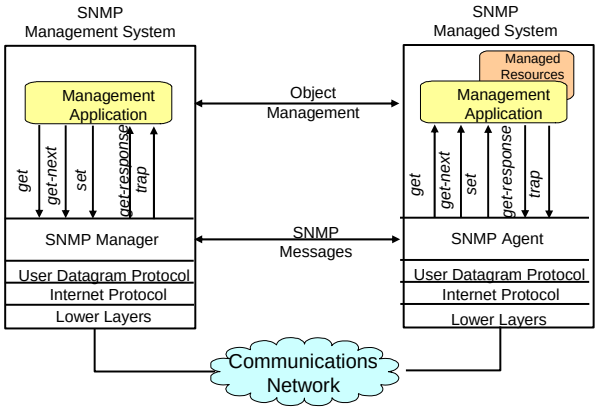


- le NMS ***supervises*** le noeud en envoyant les **requests** à ses agents demandant une réponse avec les valeurs de données dans leurs **MIB database**. Les valeurs MIB incluses sur l'interface, compteur de trafic, adresses ,etc.
- le NMS ***contrôle*** un node en appelant son agent pour mettre à jour son état de MIB ou les paramètres de configuration. Par exemple une interface réseau pouvait être désactivée en positionnant un état de variable en bas.
- le **SNMP** est utilisé comme un protocole de communication entre l'administrateur et l'agent. SNMP normalement utilise **UDP** pour le transport.
  - SMNP définit **cinq types de messages** échangés entre l'agent et l'administrateur.

137

SNMP ARCHITECTURE

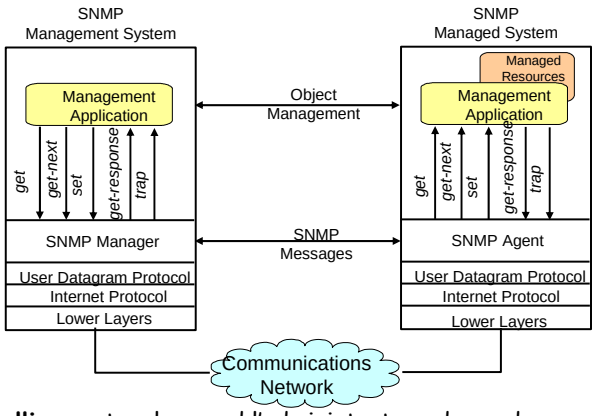
Modules spécifiques de gestion de réseau de fournisseur tels que la gestion d'erreurs,la sécurité



138

SNMP ARCHITECTURE

Modules spécifiques de gestion de réseau de fournisseur tels que la gestion d'erreurs,la sécurité

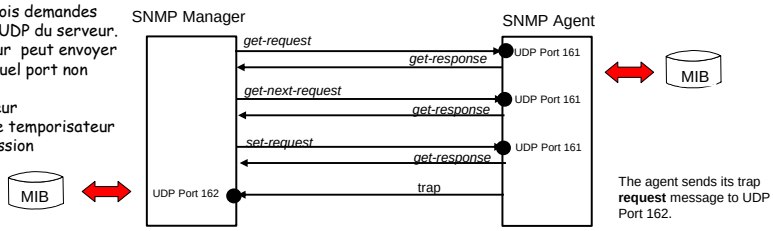


- SNMP est un **polling** protocole quand l'administrateur demande une question et l'agent lui répond.
  - la commande ***get*** obtient la valeur de MIB.
  - la commande ***set*** stocke une valeur dans la MIB.
  - l' agent renvoie une ***réponse***

139

SNMP MESSAGE TYPES

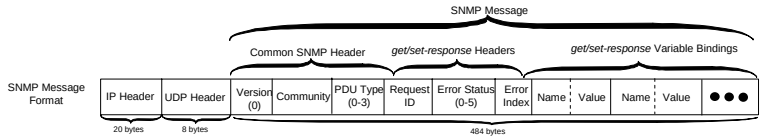
- L'administrateur envoie ses messages de trois demandes sur le port 161 UDP du serveur. L'administrateur peut envoyer sur n'importe quel port non affecté.
- L'administrateur implémentera le temporisateur et la retransmission



- ***get-request*** – obtient la valeur d'un ou plusieurs variables de l'agent. **PDU type 0.**
- ***get-next-request*** – obtient la variable prochaine (de variables multiples) aprés une variable a été obtenue. agents MIB variable est une table. **PDU Type 1.**
- ***set-request*** – positionne la valeur d'une ou plusieurs variables dans le MIB. **PDU Type 2.**
- ***get-response*** - retourne la valeur d'une ou plusieurs variables. C'est un message retourné par l'agent à l'administrateur en réponse au ***get-request***, ***get-next-request*** et ***set-request***. **PDU type 3.**
- **trap operator** – envoyée par l'agent à l'administrateur quand une occurrence d'un événement non défini sur le noeud (message d'alarme). **PDU type 4.**

140

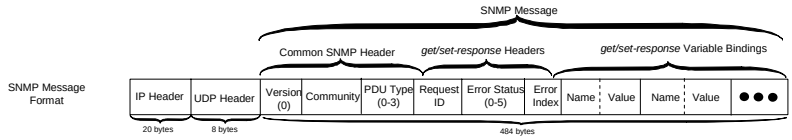
SNMP MESSAGE ENCAPSULATION



- **Version**: la valeur transportée sur le champ – 1, 0: SNMPv1 et 1: SNMPv2.
- **Community**: chaîne de 6 caractères, représente le mot de passe en claire, échangé entre l'administrateur et l'agent. Il définit les droits de l'utilisateur sur la MIB
- **PDU Type**: Le type de **P**rotocol **D**ata **U**nit(message type):
  PDU type 0 = ***get-request*** *: lecture des valeurs des objets de la MIB*
  PDU type 1 = ***get-next-request*** *: permet de lire les objets qui suivent dans l'ordre (appels récursifs)*
  PDU type 2 = ***set-request*** *: permet de fixer une valeur à un objet*
  PDU type 3 = ***get-response*** *: permet l'acquittement des autres primitives*
- **Request ID:** positionné par l'administrateur et retourné par l'agent dans un message ***get-response*** afin de coupler demande/réponse et cela permet de gérer plusieurs demande à la fois par l'administrateur et différencie ses réponses

141

SNMP MESSAGE ENCAPSULATION



- **Error Status**: un entier retourné par l'agent pour spécifier une condition d'erreur
- **Error index**: un entier spécifiant quelle variable dans le champ in **Variable Bindings (VarBind)** était erronée. Il est envoyé par l'agent dans les cas: ***noSuchname***, ***badValue***, and ***readOnly*** errors.
- **VarBind Field**: paires des noms d'objets et leurs valeurs. C'est une information d'administration associée avec ***get***, ***get-next*** and ***set*** requests.

142

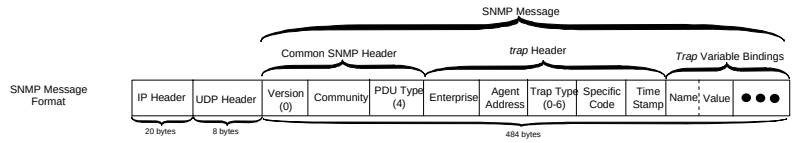
SNMP ERROR STATUS VALUES

Error Status
An integer returned by the agent to specify an error condition.

Error Status	Name	Description
0	noError	pas d'erreurs
1	tooBig	réponse de taille trop grande
2	noSuchName	variable inexistante
3	badValue	écriture d'une valeur invalide
4	readOnly	essai de modification d'une variable en lecture seule
5	genErr	autre erreur

143

SNMP MESSAGE ENCAPSULATION



- **PDU Type**: Le type du **P**rotocol **D**ata **U**nit(message type). PDU **type 4** trap (alarme ou évènement), envoyé par l'agent vers l'administrateur.
- **Community**: représente le mot de passe en claire, échangé entre l'administrateur et l'agent.
- **Enterprise:** indique le type d'objet généré et identifie le type d'opération système
- **Agent Address**: indique l'adresse IP de l'agent qui à généré le trap
- **Trap Type**: un entier de 0-6 valeurs, qui identifie le type de trap envoyé de l'agent vers l'administrateur.

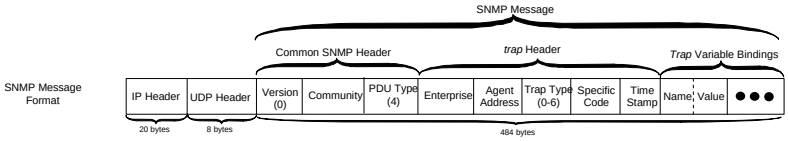
144

SNMP TRAP TYPES

Trap Type	Name	Description
0	coldStart	Initialisation de l'agent
1	warmStart	Réinitialisation de l'agent
2	linkDown	Passage de l'interface à l'état bas (première variable)
3	linkUp	Passage de l'interface à l'état haut (première variable)
4	authenticationFailure	Emission par le manager d'une communauté invalide
5	egpNeighborLoss	Passage d'un homologue EGP à l'état bas (première variable indiquant l'@ IP de l'homologue)
6	enterpriseSpecific	cf. champ spécifique pour avoir de l'information

145

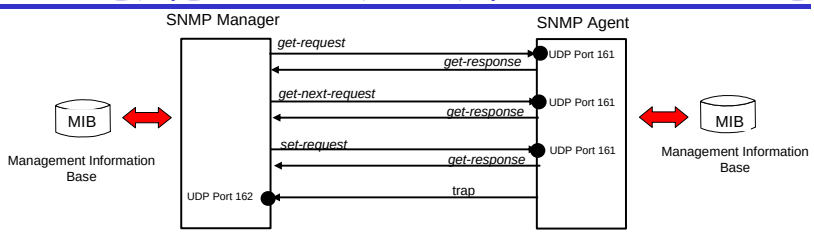
SNMP MESSAGE ENCAPSULATION



- **Specific Code**: spécifie le type d'événement produit taux d'erreurs/traffic, saturation max. de la gateway,….etc.
- **Time Stamp**: indique le temps passé depuis l'initialisation de l'agent (quelques centaines de seconde)
- **VarBind Field**: Variables que l'agent peut envoyer à l'administrateur. Couples de nom et de valeurs pairs a variable.

146

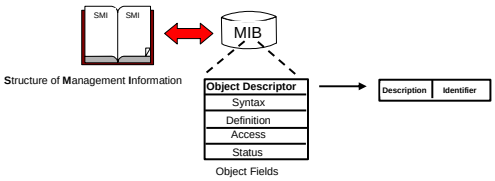
MANAGEMENT INFORMATION BASE



- le **management station** interagit avec les **agents** utilisant **SNMP protocol**.
- L'agent se refere à sa **Management Information Base (MIB)** pour avoir des détails sur les données associées
- La MIB définit tous les **network objects** (eg., router, interface, counts, etc.) pour être administrés ou contrôlés. C'est la base de données maintenue par l'agent que l'administrateur peut interroger ou positionner
  - La MIB est composée des **object groups** comme IP, UDP, TCP, SNMP, etc qui sont décrit dans une **Structure and Identification of Management Information(SMI)**.
  - Un object group est composé des **séries of objects** (bridges, routers, packet switch, modems, etc) qui sont important pour être administrés
  - Un **object to be managed** est décrit dans un **standard** en utilisant **Abstract Syntax Notation One(ASN.1)**.

147

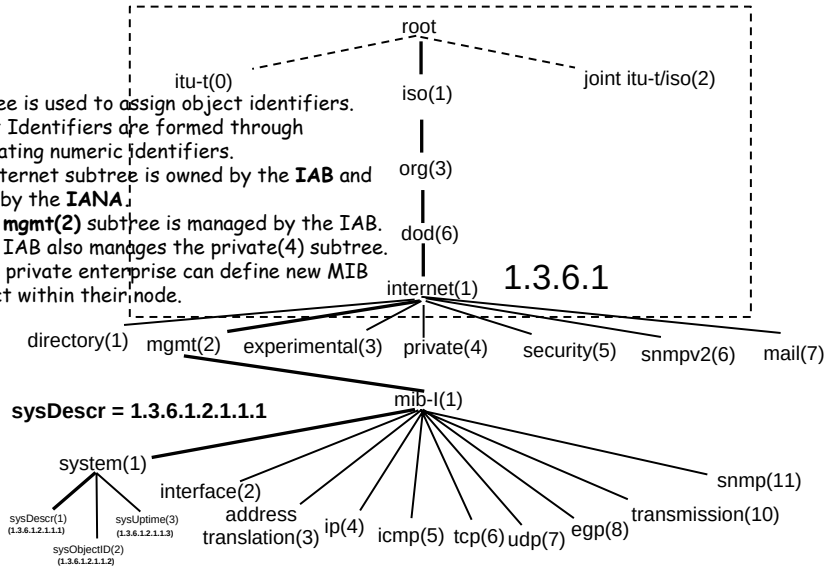
LA MIB



- Le **SMI** définit un ensemble des règles utilisées pour définir et identifier les variables MIB
- La **MIB** definite tous les **network objects**
  - La description d'un object suit le format suivant: **Object Descriptor**, **Syntax**, **Definition**, **Access** and **Status**.
- L' Object Descriptor ID est composé de deux champs:
  - **Un nom (OID)** and
  - **Une syntaxe** qui définit (on type et son codage) au niveau de **Global Registration Tree**.

148

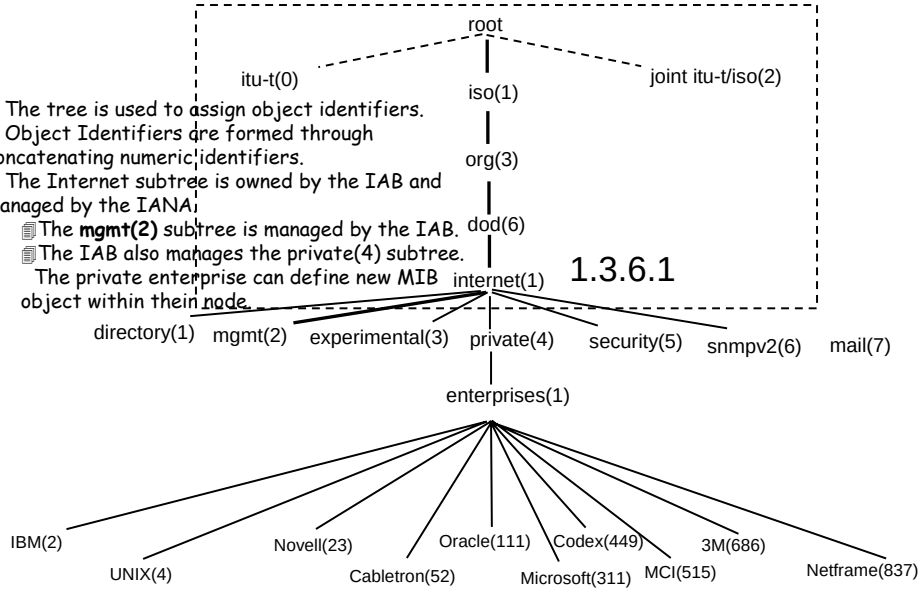
THE INTERNET REGISTRATION TREE



- The tree is used to assign object identifiers.
- Object Identifiers are formed through concatenating numeric identifiers.
- The Internet subtree is owned by the **IAB** and managed by the **IANA**.
  - The **mgmt(2)** subtree is managed by the IAB.
  - The IAB also manages the private(4) subtree. The private enterprise can define new MIB object within their node.

149

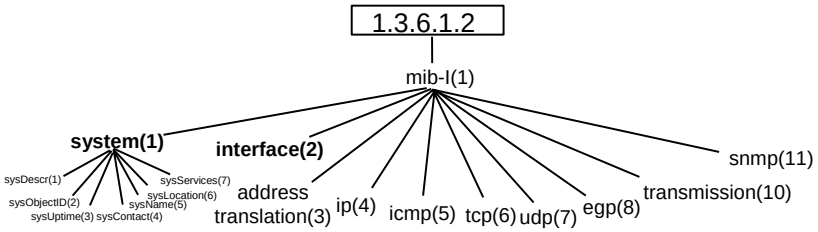
THE INTERNET VENDOR SUBTREE



- The tree is used to assign object identifiers.
- Object Identifiers are formed through concatenating numeric identifiers.
- The Internet subtree is owned by the IAB and managed by the IANA.
  - The **mgmt(2)** subtree is managed by the IAB.
  - The IAB also manages the private(4) subtree. The private enterprise can define new MIB object within their node.

150

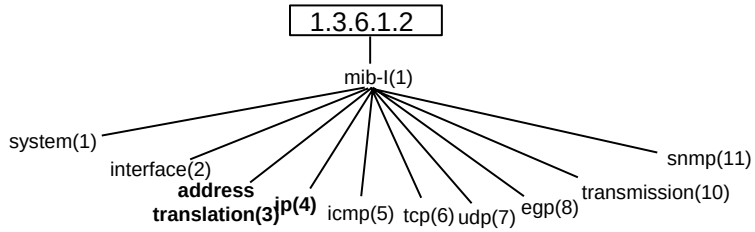
THE INTERNET REGISTRATION TREE



- **System** object describes 7 variables
  - name and version of the hardware, operating system and networking software.
  - hierarchical name of the group.
  - reinitialization time of management application.
  - contact person, object location and services offered.
- **Interface** object describes 23 variables
  - number of network interfaces supported.
  - interface type below IP (Ethernet, T-R, etc).
  - datagram size interface can handle.
  - interface speed(b/s).
  - interface address.
  - interface operational state(up/down).
  - interface traffic received, delivered , discarded and reason.

151

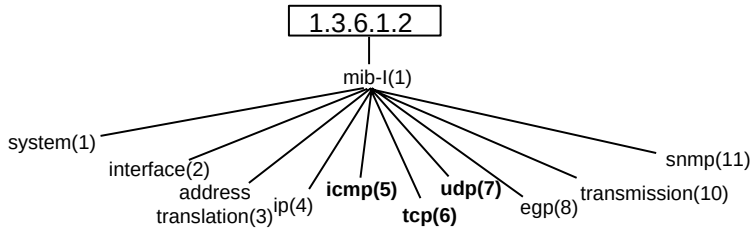
THE INTERNET REGISTRATION TREE



- **Address Translation** object describes 3 variables
  - address translation tables for the network-physical address translation.
  - This group was dropped in MIB-II since its functions now reside in the IP group.
- **Internet Protocol(IP)** object describes 42 variables
  - forwarding or discarding datagrams by device.
  - datagram ttl value originated at this site.
  - device traffic received, delivered, discarded and the reason.
  - fragmentation operation
  - address tables/subnet mask.
  - routing tables to include destination address, distance metrics, age of route, next hop, how learned(RIP, etc).

152

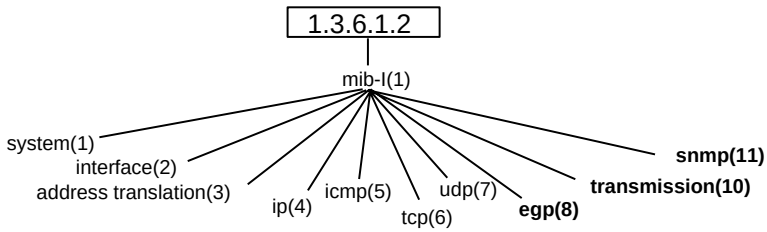
THE INTERNET REGISTRATION TREE



- **ICMP object** describes 26 variables
  - ICMP messages received and sent.
  - problem statistics(destination unreachable, time-exceeded, etc.)
- **TCP** object describes 19 variables
  - retransmission algorithm.
  - max/min retransmission values
  - TCP connections number supported
  - state transition (open to close) operations
  - traffic received/sent
  - each connections port/IP number
- **UDP** object describes 6 variables
  - traffic received/sent
  - problems encountered

153

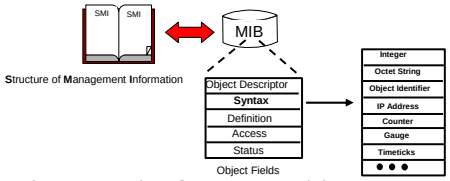
THE INTERNET REGISTRATION TREE



- **egp** object describes 20 variables
  - traffic sent/received.
  - problems encountered.
  - neighbor addresses.
  - egp neighbor state.
- **transmission** object describes 0 variables(future transmission MIBs)
  - contains MIBs for transmission systems(e.g. ATM, FDDI, X.25,etc.)
  - added in MIB-II
- **snmp** object describes 29 variables
  - objects dealing mostly with error conditions
  - added in MIB-II

154

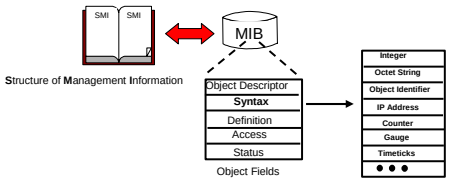
THE MANAGEMENT INFORMATION BASE



- La **Syntaxe** décrit le type d'information(datatypes qui sera passé dans l'objet entre l'agent et le NMS (serveur)). **datatypes** sont :
  - **Integer** est utilisé pour énumérer une liste de possibilités (e.g., MTU), indiquant des limites supérieure et inférieure d'un port (0-65535)
  - **Octet string** est une séquence des octets où chaque byte a une valeur entre 0-255. Il peut être employé pour représenter Par exemple:
    - ***DisplayString*** peut contenir des caractères "texte" que décrit un ***sysDescr,***
    - ***PhysAddress*** peut contenir les 6-octet Ethernet physical address, or
    - ***IpAddress*** peut contenir les 4-octet IP Internet address.

155

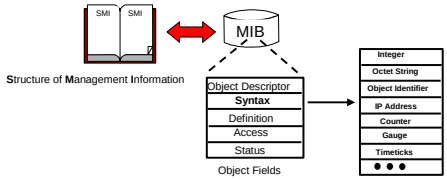
THE MANAGEMENT INFORMATION BASE



- **Object Identifier** appelé comme trouvé dans la structure arborescente. Par exemple, 1.3.6.1.4.1.9.1.1 est la marque de produit (sysObjectID) pour Cisco qui est dans le sous-arbre d'entreprise privée
- **IP Address** est une chaîne d'octets de longueur 4, spécifie l'adresse IP.

156

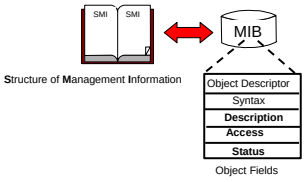
THE MANAGEMENT INFORMATION BASE



- **datatypes** sont:
  - **Counter** est un entier non négatif, utilisé pour le comptage des datagrammes (e.g. datagrams sent/received)
  - **Gauge**$_{32}$ est un compteur variable non négatif qui s'incrémente et se décrémente (4,294,967,295)(e.g., TCP connections, queue lengths, etc.).
  - **TimeTicks** est un entier$_{32}$ (compteur) non négatif, qui peut compter des centaines de secondes (ms) depuis l'événement. Par exemple, **sysUpTime** est le nombre des centaines de ms que le matériel est en état haut

157

THE MANAGEMENT INFORMATION BASE



- Le reste des champs sont :
  - **Description** est le texte ordinaire qui décrit ce que l'objet . Ce champ est prévu pour l'interprétation humaine par exemple, que le champ pourrait décrire les types de données dans un domaine identifié comme ***dataCnt***.
  - **Access** spécifie qui est admis et avec le type d'accès. L' accès spécifie est **R**ead only, **W**rite only, **R**ead-**W**rite, est non accessible.
  - **Status** indique l'état de description de l'objet **current**, **obsolete** or **deprecated (désapprouvée)** (sur le chemin à être obsolette)

158

SNMPv1 LIMITATIONS

1. L'authenticfiation est inadéquate car le nom de community est placé en clair dans un message SNMP

2. SNMP **trap directed polling** peut être générer d'échange multiple de trafics entre l'administrateur et les agents. si le réseau est congestionné, des messages SNMP peuvent être perdus

3. SNMP standards permet de définir la **proprietary MIBs** .
qui peut mener aux problèmes d'interopérabilité
4. Les variables MIB peuvent être sélectionnés "**polled separately"**, i.e. le MIB entier ne peut pas être cherché avec une commande simple

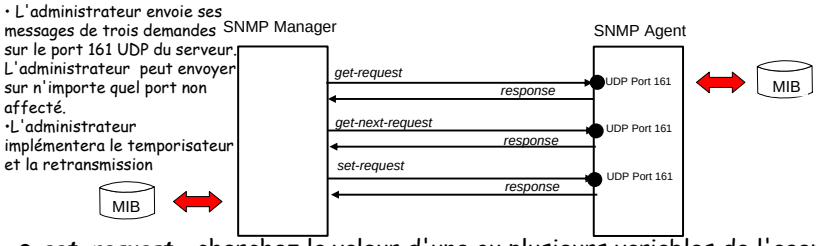
159

SNMPv2 ENANCEMENTS

1. A ***get-bulk-request*** permet au NMS, la recherche par bloc de données (un tableau) contrairement au ***get-requests***.
2. A ***inform-request*** permet à un administrateur d'envoyer une information à un autre administrateur (référencement d'une donnée à un tiers (un proxy))
3. Définit plusieurs **Management Information Bases**.
4. Fournit des options de sécurité pour l' options for **authentification et integrity**, **access controls**, et **security and privacy**.
5. Le message **trap** a le même format comme les messages ***get/set***
6. Un agent peut mettre un **error code** dans un champ de pour une qui ne peut pas être recherchée
7. A **locking function** pour empêcher l'administrateur d'écrire au même agent
8. Les agents peuvent recevoir une **confirmation** de leurs messages d'événement (trap)

160

SNMPv2 MESSAGE TYPES



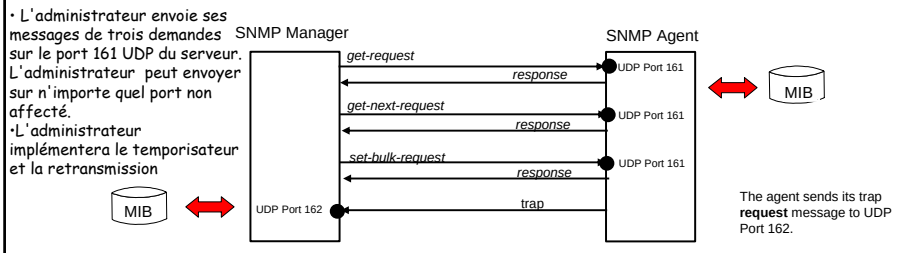
- ***get-request*** - cherchez la valeur d'une ou plusieurs variables de l'agent. SNMPv2 fournira une réponse partielle tandis que SNMPv1 est atomique (tous ou rien). **PDU type 0**.
- ***get-next-request*** - cherchez la prochaine variable (des variables multiples) après qu'une variable ait été cherchée. SNMPv2 traitera autant de variables car possible tandis que SNMPv1 est atomique. **PDU Type 1**.

response **-** renvoyez la valeur d'une ou plusieurs variables.C'est un message retourné par l'agent à l'administrateur en réponse au the get-request, get-next-request et the set-request. Il s'est raccourci dans SNMPv2 à la réponse juste. **PDU type 2**.

- ***set-request*** - placez la valeur d'une ou plusieurs variables dans le noeud MIB. Une opération d'ensemble est exécutée en deux phases. Chaque variable est validée et si on échoue l'opération entière échoue. Si la première phase réussit alors la deuxième phase effectue l'opération. SNMPv1 et SNMPv2 sont atomiques. **PDU Type 3**.

161

SNMPv2 MESSAGE TYPES



- **get-bulk-request** - c'est une nouvelle opération pour permettre la récupération d'une quantité massive de données avec une opération simple. Elle agit semblable comme ***get-next-request***. **PDU type 5**.
- **inform-request** - permet à un administrateur SNMP d'envoyer l'information choisie à un autre administrateur SNMP. Semblable au trap,sauf que ***inform-request*** reçoit une confirmation le récepteur et lui peut rendre compte d'un événement beaucoup plus complexe. **PDU type 6.**
- **trap** – l'agent notifie l'administrateur quand un évenement anormal produit au niveau du noeud. Comme SNMPv1 sauf il utilise le même format commes les autres opérations sauf ***get-bulk-request***. ***PDU type 7.***
- **report** - l'utilisation et la sémantique ne sont pas actuellement définies. ***PDU type 8***

162