Partie 2 : Applications de l'Internet de type Client/Serveur

Olivier GLÜCK
Université LYON 1/Département Informatique
Olivier.Gluck@univ-lyon1.fr
http://perso.univ-lyon1.fr/olivier.gluck

1

Copyright

2

Remerciements

- Certains transparents sont basés sur des supports de cours de :
  - Olivier Aubert (LYON 1)
  - Olivier Fourmaux (UPMC)
  - Bénédicte Le Grand (UPMC)
- Des figures sont issues des livres cités en bibliographie

3

Bibliographie

- « *Réseaux* », 4ième édition, Andrew Tanenbaum, Pearson Education, ISBN 2-7440-7001-7
- « La communication sous Unix », 2ième édition, Jean-Marie Rifflet, Ediscience international, ISBN 2-84074-106-7
- « *Analyse structurée des réseaux* », 2ième édition, J. Kurose et K. Ross, Pearson Education, ISBN 2-7440-7000-9
- « *TCP/IP Illustrated Volume 1. The Protocols* », W. R. Stevens, Addison Wesley, ISBN 0-201-63346-9
- « *TCP/IP. Architecture, protocoles, applications* », 4ième édition, D. Comer, Dunod, ISBN 2-10-008181-0
- Internet…
  - http://www.w3.org/
  - http://www.rfc-editor.org/ (documents normatifs dans TCP/IP)

4

Plan de la partie 2

- Introduction / Rappel
- Connexions à distance (telnet/rlogin/rsh/ssh/X11)
- Applications de transfert de fichiers (FTP/TFTP)
- Accès aux fichiers distants (NFS/SMB)
- Gestion d'utilisateurs distants (NIS)
- DNS : un annuaire distribué
- LDAP : un annuaire fédérateur sécurisé
- La messagerie électronique (SMTP/POP/IMAP)

5

Introduction / Rappels

6

La couche application

- La couche application
  - gère les logiciels utilisateurs (applications) en s'appuyant sur les services de bout en bout définis dans les couches de niveau inférieur
  - repose généralement sur le modèle Client/Serveur (modèle requête/réponse)
  - supporte les environnements hétérogènes
- On distingue l'application et le protocole applicatif
  - le protocole applicatif définit les échanges entre les parties cliente et serveur de l'application
  - une interface (API) permet au protocole applicatif d'utiliser les services de bout-en-bout fournis par un protocole de transport sous-jacent

7

Quel service de transport ?

- Socket = interface entre le processus applicatif et le protocole de transport
  - Côté émetteur : l'application envoie des messages par la porte
  - De l'autre côté de la porte, le protocole de transport doit déplacer les messages à travers le réseau, jusqu'à la porte du processus récepteur
- De nombreux réseaux (dont Internet) fournissent plusieurs protocoles de transport
  - Lequel choisir lorsqu'on développe une application ?
    - Étude des services fournis par chaque protocole
    - Sélection du protocole qui correspond le mieux aux besoins de l'application

8

Quel service de transport ?

- Faut-il choisir le train ou l'avion pour faire Paris/Nice ?
  - tout dépend des critères du voyageur (rapidité, confort, sécurité, prix, arrivée en centre ville…)
- 3 types de besoins au niveau des applications :
  - fiabilité du transfert (S'autorise t-on à perdre quelques données ? Dans quelle proportion ?)
  - bande passante (Quelle est la taille minimale du tuyau de communication ?)
  - délai : latence et gigue (variation du délai)

9

Quel service de transport ?

- Fiabilité du transfert
  - Certaines applications nécessitent une fiabilité à 100%
    - Courrier électronique (SMTP)
    - Transfert de fichiers (FTP)
    - Accès distant (Telnet)
    - Transfert de documents Web (HTTP)
    - Applications financières
  - D'autres peuvent tolérer des pertes (loss-tolerant applications)
    - Applications multimédia : audio/vidéo (une perte d'une faible quantité de données n'induit qu'une petite irrégularité dans l'écoute ou la vision du film)

10

Quel service de transport ?

- Bande passante
  - Certaines applications requièrent une bande passante minimale
    - Téléphonie sur Internet : si la voix est codée à 32 Kbps, les données doivent être transmises et reçues à ce débit
    - Applications multimédia
  - D'autres utilisent la bande passante disponible (applications élastiques)
    - Courrier électronique, transfert de fichiers, accès distant, Web
    - Plus il y a de bande passante, mieux c'est !

11

Quel service de transport ?

- Délai (contraintes temporelles)
  - Certaines applications nécessitent un délai de bout-en-bout faible (moins de quelques centaines de ms)
    - Applications temps réel interactives :
      - Téléphonie sur Internet
      - Environnements virtuels
      - Téléconférence
      - Jeux en réseau
  - Pour les applications non temps réel, un délai court est préférable, mais pas de contrainte forte

12

Quel service de transport ?

Application	Pertes	Bande passante	Sensibilité temp.
Transfert de fichiers	sans perte	élastique	Non
e-mail	sans perte	élastique	Non
Pages Web	sans perte	élastique	Non
Audio/vidéo temps réel	tolérant	audio: 5Kb - 1Mb vidéo:10Kb - 5Mb	Oui, 100's ms
Audio/vidéo enregistré	tolérant	idem	Oui, quelques s
Jeux interactifs	tolérant	quelques Kbps	Oui, 100's ms
Applis financières	sans perte	élastique	Oui et non

13

Services proposés dans Internet

Service TCP :

- orienté connexion : connexion nécessaire entre le client et le serveur
- transport fiable entre le processus émetteur et récepteur
- contrôle de flot : l'émetteur ne submerge pas le récepteur
- contrôle de congestion : réduit le débit de l'émetteur quand le réseau est congestionné
- ne propose pas :
  - de garantie de délai,
  - de bande passante minimale

Service UDP :

- transfert de données non fiable
- ne propose pas
  - de connexion,
  - de fiabilité,
  - de contrôle de flot,
  - de contrôle de congestion,
  - de garantie temporelle,
  - de bande passante
- beaucoup plus simple que TCP (UDP=IP) donc plus rapide
- pas de limitation du débit

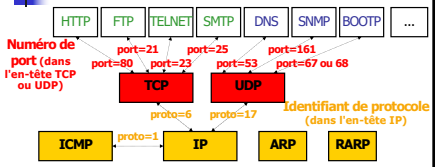
14

Principales applications Internet

Application	Protocole applicatif	Protocole de transport
e-mail	SMTP [RFC 821,2821]	TCP
Accès distant	telnet [RFC 854]	TCP
Web	HTTP [RFC 2068,2616]	TCP
Transfert de fichiers	FTP [RFC 959]	TCP
Streaming multimedia	propriétaire	TCP ou UDP
Serveur Fichiers	NFS	TCP ou UDP
Voix sur IP	propriétaire	En général UDP

15

Principales applications Internet



Tous les ports sur
http://www.iana.org/assignments/port-numbers

16

Applications de connexion à distance

Principes

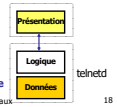
telnet, rlogin, rsh, ssh, X11

17

Connexions à distance

- Application permettant à un utilisateur de se connecter (prendre partiellement le contrôle)
  - sur un ordinateur distant (à partir d'un terminal local)
  - pourvu que cet utilisateur dispose d'un accès autorisé à cette machine
  - exécution de commandes saisies localement au clavier sur une machine distante
  - les environnements local et distant peuvent être hétérogènes (windows-->unix, ...)

Émulation de terminaux :



18

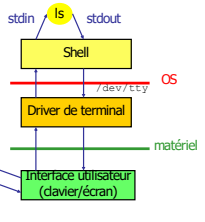
Connexions à distance

- Plusieurs protocoles
  - `telnet` : le standard (existe sur de nombreuses plate-formes)
  - `rlogin` : uniquement entre machines unix
  - `ssh` : sécurisé (authentification + cryptage), peut transporter le DISPLAY
- Besoin de l'application : inter-activité
  - tout ce qui est tapé au clavier sur le client est envoyé sur la connexion au serveur
  - tout ce qui est envoyé par le serveur vers le client, sur la connexion, est affiché dans le terminal

19

Connexions locales

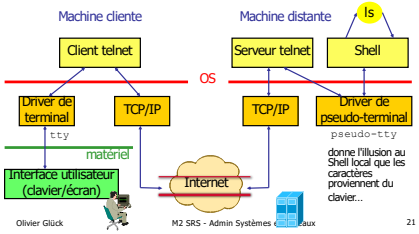
- Fonctionnement d'une connexion locale



20

Connexions distantes

- Fonctionnement d'une connexion distante



21

Telnet : protocole et application

TELecommunication NETwork protocol

- un des premiers standard de l'Internet : RFC 854,855 (1983)
- connexion TCP sur le port 23 côté serveur
- authentification sur le shell distant (mot de passe en clair)
- quand un caractère est tapé au clavier, il est envoyé au serveur qui renvoie un "écho" du caractère ce qui provoque son affichage dans le terminal local
- prise en compte de l'hétérogénéité
  - mécanisme de négociation d'options à la connexion (codage des caractères ASCII sur 7 ou 8 bits ?)
  - exemple : `telnet` d'une machine Windows vers une machine Unix --> tous les caractères ASCII n'ont pas la même signification

22

Exécution de Telnet

- Les différentes exécutions possibles (côté client)
  - sans argument (paramètrer sa connexion distante)

```
telnet
```

  - par le nom de la machine distante (DNS+port 23)

```
telnet nom_du_serveur
```

  - par l'adresse IP de la machine distante (port 23)

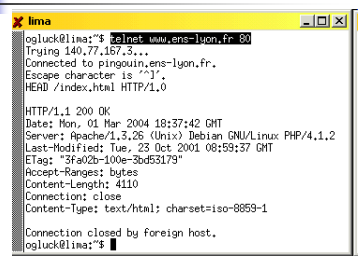
```
telnet adr_IP_du_serveur
```

  - accès à un autre service (connexion sur un autre port)

```
telnet adr_IP_du_serveur numéro_port
```


23

Exécution de Telnet



24

RLOGIN : principe

Remote LOGIN (service `login` dans `inetd.conf`)

- Application standard d'unix BSD (RFC 1282) (dec 1991)
- Connexion TCP sur le port 513 côté serveur
- Plus simple que `telnet` (que sous Unix)
- Idée : lors de la connexion, les paramètres du terminal local sont envoyés au site distant (pas de négociation)
- Intérêts de `rlogin` par rapport à `telnet`
  - permet à l'administrateur de définir un ensemble de machines "équivalentes" sur lesquelles les noms d'utilisateurs et les droits d'accès sont partagés
    - exemple : un utilisateur a un login X sur m1 et Y sur m2
  - permet des accès automatiques sans saisir de mot de passe
  - permet d'exporter sur la machine distante une partie de l'environnement local (type du terminal `$TERM`, taille de la fenêtre) : un terminal distant a alors un comportement similaire à un terminal local (couleurs…)

32

RLOGIN : authentification

- Authentification
  - si un mot de passe est nécessaire, il circule en clair
- Authentification automatique
  - pour ne pas avoir à saisir de mot de passe, il faut
    - soit que la machine cliente soit dans le fichier `/etc/hosts.equiv` de la machine distante
    - soit que le couple (machine cliente, user) soit dans le fichier `$HOME/.rhosts` de la machine distante
    - le démon `rlogind` examine d'abord si le fichier `/etc/hosts.equiv` permet une authentification automatique, puis si tel n'est pas le cas, il regarde le fichier `$HOME/.rhosts`

33

RLOGIN : authentification

- le fichier `$HOME/.rhosts` permet d'éviter l'authentification de certains couples (machine cliente/utilisateur)

```
ogluck@192.168.69.1# cat .rhosts
192.168.69.2 ogluck
```

- le fichier `/etc/hosts.equiv` contient les machines "équivalentes" ou des entrées de type `.rhosts`

```
ogluck@192.168.69.2# cat /etc/hosts.equiv
192.168.69.1 # autorise tout le monde
192.168.69.1 ogluck # que ogluck
+ ogluck # ogluck depuis n'importe où
```


34

RSH : principe

Remote SHell

- Connexion TCP port 514 - le pendent de `rlogin`
- Exécution de commandes sur une machine distante de façon transparente

```
rsh host cmd
```

  - authentification automatique comme avec `rlogin`
  - tout se passe comme si l'exécution était locale
    - l'entrée standard et la sortie standard de `cmd` sont directement connectées à la socket cliente
    - avantage : peut être utilisé directement dans un programme (pas de saisie de mot de passe)
    - quand `cmd` se termine sur le site distant, le processus `rsh` client se termine
    - une séquence `Ctrl-c` termine le processus distant `cmd`

35

RSH : principe

- Exemple

```
ogluck@192.168.69.1# rsh 192.168.69.2 ls
interfaces
iperf-1.7.0
iperf-1.7.0-source.tar.gz
iperf.deb
```

- Fonctionnement du démon `rshd` quand une requête arrive
  - 1- lecture sur la socket jusque `'\0'` (octet nul) ; la chaîne lue est interprétée comme un numéro de port
  - 2- une deuxième connexion est établie vers le client vers ce numéro de port pour transmettre `stderr` (permet de distinguer `stderr` et `stdout` dans les `>`)

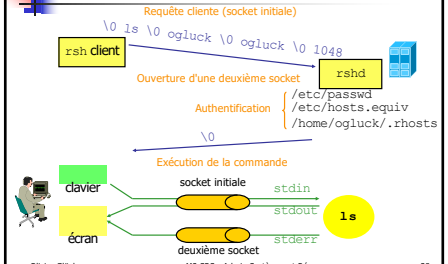
36

RSH : principe

- 3- récupération de l'@ IP cliente pour déterminer un nom éventuel (requête DNS) pour l'authentification
- 4- lecture sur la socket initiale
  - du `username` sur la machine cliente (`user_l`)
  - du `username` sur la machine distante (`user_d`)
  - de la ligne de commande à exécuter
- 5- le démon authentifie l'exécution distante
  - il vérifie que `user_d` est bien dans `/etc/passwd`
  - si `user_l=user_d`, regarde dans `/etc/hosts.equiv`
  - sinon regarde dans `$HOME/.rhosts`
- 6- une fois `user_d` authentifié, le démon renvoie `'\0'` au client puis passe la ligne de commande au shell local

37

RSH : principe



38

SSH : un shell distant sécurisé

Secure SHell

- Les communications sont cryptées
- Authentification à base de clés
- Un des seuls protocoles de connexion à distance qui passe les pare-feux de nos jours
- Permet de transporter des fenêtres graphiques via le tunnel SSH (multiplexage de plusieurs flux dans la connexion)
- Connexion TCP sur le port 22 côté serveur
- Pas encore de RFC (ietf-internet-draft)

39

SSH : syntaxes

- Connexions à distance (style `rlogin`)

```
ssh -l user hostname
ssh user@hostname
```

- Exécution de commande à distance (style `rsh`)

```
ssh -l user hostname cmd
ssh user@hostname cmd
```

- Copie de fichiers à distance (style `rcp`)

```
scp file1 file2 user@hostname:
scp -r dir user@hostname:/tmp
```

- `ssh` et `scp` remplacent `rlogin`, `rsh`, `rcp`, ...

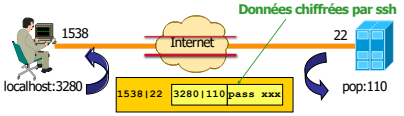
40

SSH : tunnels (*port forwarding*)

- `telnet pop.univ-lyon1.fr 110` ---> non sécurisé !
- On peut faire passer n'importe quoi dans un tunnel ssh :

```
ssh -L 3280:pop.univ-lyon1.fr:110 pop.univ-lyon1.fr
telnet localhost 3280
```

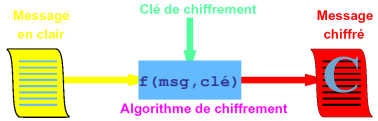
- Tout ce qu'on envoie sur `localhost/3280` arrive sur `pop/110` mais les données sont chiffrées (encapsulées dans le protocole `ssh`)



41

SSH : chiffrement

- Principe du chiffrement



- La qualité de la sécurité dépend
  - du secret de la clé
  - de la longueur de la clé (plus il y a de bits, plus il est difficile d'essayer toutes les clés)
  - de la difficulté d'inversion de l'algorithme de chiffrement

42

SSH : chiffrement

- Deux types d'algorithmes
  - symétrique : même clé privée secrète partagée utilisée pour le chiffrement et le déchiffrement
    - l'émetteur et le récepteur doivent se mettre d'accord sur **la** clé à utiliser
  - asymétrique : utilisation d'une clé publique pour le chiffrement et d'une clé privée pour le déchiffrement

`Mchiffré=f(Mclair, Cpublique)` et `Mclair=g(Mchiffré, Cprivée)`

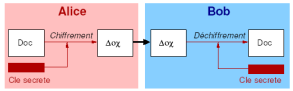
  - pour qu'un émetteur envoie un message chiffré, il suffit qu'il connaisse **la** clé publique du destinataire
  - pb : comment être sûr que la clé publique est bien celle du destinataire escompté ?
  - certificat : association d'une clé publique et d'un nom de destinataire signée par un tiers de confiance

43

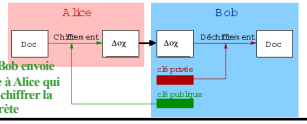
Chiffrement symétrique et asymétrique

source : M. Herrb

Symétrique : DES / AES



Assymétrique : RSA (DSA - El-Gamal)



Le client ssh Bob envoie sa clé publique à Alice qui l'utilise pour chiffrer la clé secrète

44

SSH : chiffrement

- Dans SSH :
  - algorithme asymétrique pour l'authentification (généralement RSA : basé sur l'arithmétique modulo)
  - algorithme symétrique pour les communications
    - utilisation de RSA pour échanger la clé de l'algorithme symétrique
    - chiffrement et déchiffrement moins coûteux

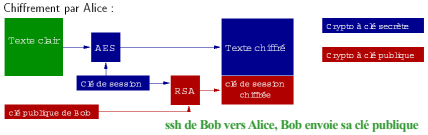
45

Chiffrement pratique

source : M. Herrb

Fonctions à clé publique très coûteuses → utilisation d'une **clé de session**

Chiffrement par Alice :



Longueur des clés sûres (2003) :

clé secrète : 128 bits
clé publique/privée : 1024 bits

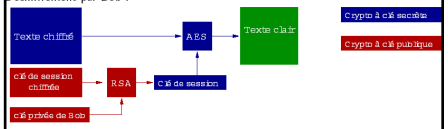
ssh de Bob vers Alice, Bob envoie sa clé publique à Alice, Alice fabrique une clé secrète (la clé de session) pour l'envoyer chiffrée à Bob. C'est cette clé secrète qui sera utilisée ensuite pour chiffrer/déchiffrer de manière symétrique avec AES.

46

Déchiffrement pratique

source : M. Herrb

Déchiffrement par Bob :



47

SSH : authentification

- 4 méthodes sont essayées dans l'ordre par `sshd`
  - authentification automatique (souvent désactivée sur le serveur car considérée *insecured*)
    avec `/etc/hosts.equiv` ou `/etc/ssh/shosts.equiv`
    avec `~/.rhosts` ou `~/.shosts`
  - authentification automatique forte "améliorée"
    idem (avec fichiers `rhosts` ou `hosts.equiv`) mais
    combinée avec une authentification des *hosts* par RSA : le serveur vérifie la clé en provenance de la machine cliente
    Mot de passe demandé lors de la première connexion puis ajout du couple (@IP, clé publique) du serveur dans le fichier du client
    Si la clé publique change pour cette @IP alors on redemande une authentification forte par mot de passe
    `/etc/ssh/ssh_known_hosts` ou `~/.ssh/ssh_known_hosts`

48

SSH : authentification

- 4 méthodes sont essayées dans l'ordre par `sshd`
  - authentification automatique forte par RSA (asymétrique)
    - le client génère un couple (clé_pub, clé_pri) avec `ssh-keygen` et copie la clé publique dans le fichier `~/.ssh/authorized_keys` sur le serveur ; liste les clés publiques autorisées (équivalent du `~/.rhosts` !)
    - le client envoie au serveur sa clé publique ; si elle est dans le fichier, le serveur génère un nb aléatoire de 256 bits qu'il chiffre avec la clé envoyée par le client ; le client déchiffre avec sa clé privée puis la renvoie avec hachage MD5 ; le serveur calcule le hachage MD5 et vérifie
  - authentification par saisie du mot de passe mais ce dernier est chiffré par RSA avant d'être envoyé au serveur

49

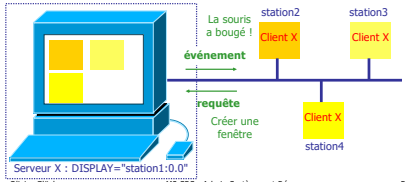
X : multi-fenêtrage réparti

- Système de multi-fenêtrage sous Unix
  - appelé X ou X Window System ou X11
  - ensemble de programmes réalisant l'interface Homme/Machine basé sur l'utilisation des périphériques (clavier, souris, écran, …)
- X est constitué de plusieurs entités
  - un serveur X : gère le matériel (clavier, écran, …) et leur utilisation par les applications graphiques ; accessible sur le port TCP 6000+`n` où `n` est le numéro de DISPLAY
  - des clients X : applications graphiques qui nécessitent un serveur X (`xemacs`, `xterm`, `xcalc`, `xv`, …)
  - le protocole X : fait communiquer les clients et le serveur

50

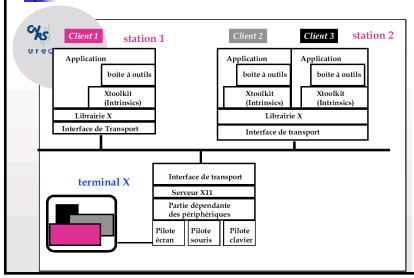
X : multi-fenêtrage réparti

- Système réparti : permet de travailler sur plusieurs machines simultanément
  - les clients X peuvent s'exécuter sur des machines distantes (3 connexions TCP dans l'exemple)



51

X : multi-fenêtrage réparti



52

X : multi-fenêtrage réparti

- Chaque client X peut définir ses caractéristiques
  - spécifications standards
    - fontes, géométrie de la fenêtre, background, foreground, borderwidth, couleurs…
  - spécifications particulières à l'application
    - affichage ou non d'un ascenseur…
- Gestion de fenêtres : *Window Manager*
  - un client X particulier qui gère
    - déplacement/redimensionnement de fenêtre
    - créer/détruire/iconifier des fenêtres
    - lancer ou terminer des applications X

53

X : multi-fenêtrage réparti

- Le protocole X permet au serveur X de contrôler l'autorisation des accès
  - Quels clients X peuvent se connecter au serveur X ?
  - La commande `xhost`

```
ogluck@lima:~$ xhost
access control enabled,only authorized clients can connect
ogluck@lima:~$ echo $DISPLAY
140.77.13.102:0.0
ogluck@lima:~$ xhost + ble
ble being added to access control list
ogluck@lima:~$ rlogin ble
ogluck@ble:~$ export DISPLAY=140.77.13.102:0.0
ogluck@ble:~$ xterm &
ogluck@ble:~$ exit
Connection to ble closed.
ogluck@lima:~$ xhost - ble
ble being removed from access control list
```

Qui est le serveur X ?

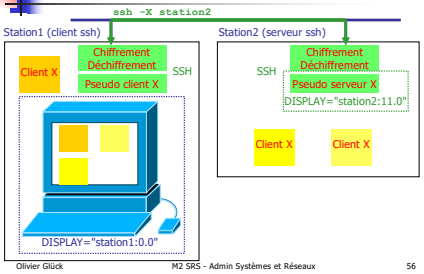
54

SSH : X11 et TCP forwarding

- X11 Forwarding
  - permet d'avoir une encapsulation chiffrée du protocole X11 dans la connexion `ssh` avec une gestion automatique de la variable `$DISPLAY`
  - si la variable `$DISPLAY` du client ssh est positionnée, `ssh -X` permet au serveur d'exporter les fenêtres graphiques lancées à partir de la connexion `ssh` vers le `$DISPLAY` du client (un "*proxy X server*" est créé sur la machine serveur pour transférer les connexions X vers le client via la session `ssh`)
- Possibilité de rediriger n'importe quel port TCP (dépend de la configuration de ssh)

55

SSH : X11 forwarding



56

Applications de transfert de fichiers

Protocoles de transfert de fichiers
FTP : File Transfer Protocol
TFTP : Trivial File Transfer Protocol

57

Protocoles de transfert de fichiers

- Copie intégrale d'un fichier d'un système de fichiers vers un autre en environnement hétérogène
- L'hétérogénéité concernant les fichiers est dépendante d'un système à l'autre
  - de la façon de représenter les noms de fichier (longueur, caractère espace,…)
  - des droits d'accès au fichier (lecture, écriture, exécution, propriétaire, …)
  - de la représentation des données contenues dans le fichier (saut de ligne…)
    - --> mode `ascii` : transfert au format NVT avec conversion au format local (`TYPE A`)
    - --> mode `binary` : transfert sans conversion (`TYPE I`)

58

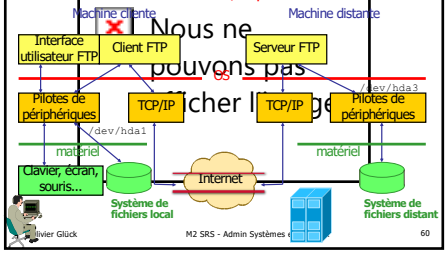
Protocoles de transfert de fichiers

- Plusieurs protocoles
  - copie de fichiers à distance : `rcp`, `scp`
  - protocole de transfert de fichiers avec accès aux systèmes de fichiers local et distant : `ftp`, `tftp`, `sftp`
- Type client/serveur
  - le client (initiateur de la connexion) interagit avec l'utilisateur, le système de fichiers local et les protocoles réseau
  - le serveur (héberge les fichiers distants) interagit avec les protocoles réseau et le système de fichiers distant
- Ne pas confondre avec les protocoles d'accès aux fichiers distants : NFS (RPC), SMB (Microsoft)

59

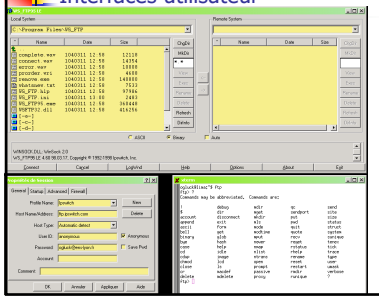
Protocoles de tranfert de fichiers

- Transfert de fichiers vers/depuis un hôte distant



60

Interfaces utilisateur



61

FTP : File Transfer Protocol - RFC 959

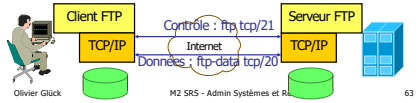
- Standard TCP/IP pour le transfert de fichiers
- Connexion TCP sur le port 21 côté serveur
- Contrôle d'accès au serveur distant (login,mdp)
  - le mot de passe circule en clair
- Particularité de FTP par rapport à TELNET… : utilisation de 2 connexions TCP

```
ogluck@lima:~$ cat /etc/services | grep ftp
ftp-data        20/tcp
ftp             21/tcp
tftp            69/udp
sftp            115/tcp # FTP over SSH
ftps-data       989/tcp #FTP over SSL (data)
ftps            990/tcp # FTP over SSL
```


62

FTP : Connexions contrôle et données

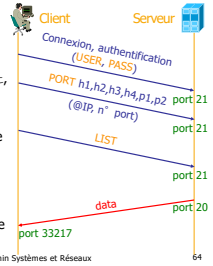
- Les clients FTP contactent le serveur FTP sur le port TCP/21
- Ouverture de 2 connexions TCP parallèles :
  - Contrôle : échange des commandes et des réponses entre le client et le serveur - "contrôle hors-bande"
  - Données : transfert des fichiers de données vers/depuis l'hôte distant (sur le port TCP/20 côté serveur)
- Le serveur FTP maintient un "état" : répertoires courants local et distant, username



63

FTP : Connexions contrôle et données

- Scénario d'une connexion
  - le client FTP se connecte sur le port 21 du serveur
  - le port 21 sert à envoyer des commandes au serveur FTP (`put`, `get`, `cd`, …)
  - si une commande nécessite que des données soient reçues ou transmises (`ls`, `get`, `put`, …), le client envoie une commande `PORT` au serveur indiquant un port (p1*256+p2) sur lequel le serveur va créer une connexion `ftp-data` depuis son port 20
  - la connexion `ftp-data` est close dès que le transfert est terminé



64

FTP : Connexions contrôle et données

- Connexion contrôle (`ftp`) :
  - échange des requêtes/réponses (dialogue client/serveur)
  - permanente, full-duplex, besoin de fiabilité (et faible délai !)
  - initiée par le client
- Connexion données (`ftp-data`) :
  - envoi de fichier ou liste de fichiers/répertoires (données)
  - temporaire, full-duplex, besoin de débit (et fiabilité !)
  - initiée par défaut par le serveur
    - ouverture active (`connect()`) du serveur vers le client (depuis le port 20 vers le port ??)
  - la connexion est fermée dès que le caractère `EOF` est lu

65

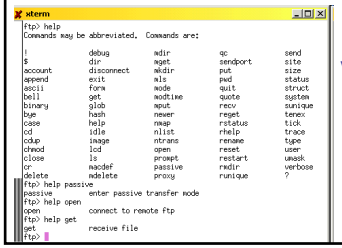
FTP : Connexions contrôle et données

- Active transfer & Passive transfer
  - Active transfer : la connexion `ftp-data` est initiée par le serveur
    - --> problème de firewall ou de NAT : impossibilité de créer la connexion à partir du serveur même s'il connaît le numéro de port du client
  - Passive transfer : `ftp-data` initiée par le client
    - intégré dans les navigateurs, paramétrable sur certains clients
    - fonctionnement : le client envoie la commande `PASV` au lieu de `PORT` sur le port 21 (RFC 1579 : Firewall-Friendly FTP) ce qui revient à demander au serveur de faire un `listen()` sur le port 20

66

Commandes du client FTP

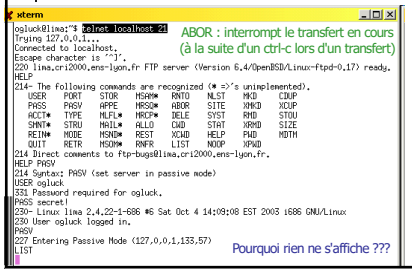
- Ne pas confondre avec les commandes du protocole FTP !



Varie d'un client à l'autre !

67

Requêtes du protocole FTP



68

Requêtes du protocole FTP

RETR <filename>
Déclanche la transmission par le serveur du fichier <filename> sur le canal de données.
STOR <filename>
Déclanche la réception d'un fichier qui sera enregistré sur le disque sous le nom <filename>. Si un fichier avec le même nom existe déjà il est remplacé par un nouveau avec les données transmisses.
APPE <filename>
Déclanche la réception d'un fichier qui sera enregistré sur le disque sous le nom <filename>. Si un fichier avec le même nom existe déjà, les nouvelles données lui sont concaténée.
REST <offset>
Redémarrage en cas d'échec d'un transfert précédent. L'offset précise le numéro du dernier octet reçu.
ABOR : abandon d'un transfert en cours.

69

Requêtes du protocole FTP

PWD : impression du répertoire courant.
LIST : catalogue du répertoire courant (canal donnée).
NLST : catalogue succint (canal donnée).
CWD <repname> : changement de répertoire courant pour <repname>.
MKD <repname> : création du nouveau répertoire <repname>.
RMD <repname> : suppression du répertoire <repname>.
DELE <filename> : suppression du fichier <filename>.
RNFR <filename1> : définit le nom actuel d'un fichier à renommer.
RNTO <filename2> : définit le nouveau nom d'un fichier à renommer.
STAT : status courant de la session FTP.
STAT <repname> : équivalent à LIST mais réponse sur le canal de contrôle
HELP : affiche l'aide sur les opérations du site.
NOOP : no operation.

70

Réponses du protocole FTP

- Les réponses sont de la forme

`status_code description`

Le code est un nombre sur trois chiffres signifiant :

Status	Signification	Status	Signification
		x0z	Erreur de syntaxe
1yz	Réponse positive préliminaire (une autre réponse suivra)	x1z	Réponse Informative (HELP…)
2yz	Réponse positive finale (une autre requête est possible)	x2z	Relatif à une connexion
3yz	Réponse positive intermédiaire (une autre requête doit suivre)	x3z	Relatif à l'authentification
4yz	Réponse négative temporaire (la même requête peut réussir plus tard)		
5yz	Réponse négative définitive (la requête n'est pas acceptée)	x5z	Relatif au système de fichier

71

Exemples de réponses

- `125 Data connection already open`
- `150 Opening BINARY mode data connection`
- `200 Command successful`
- `214 Help message`
- `220 lima.cri2000.ens-lyon.fr FTP server (Version 6.4/OpenBSD/Linux-ftpd-0.17) ready`
- `226 Transfer complete`
- `230 User ogluck logged in`
- `331 Passwd required for ogluck`
- `425 Can't open data connection`
- `452 Error writing file`
- `500 Command not understood`
- `550 No files found`

72

Exemple de dialogue FTP

```
ogluck@lima:~$ ftp -d -v localhost
Connected to localhost.
220 lima.cri2000.ens-lyon.fr FTP server (Version 6.4/OpenBSD/Linux-ftpd-0.17) ready.
Name (localhost:ogluck): ogluck
---> USER ogluck
331 Password required for ogluck.
Password:
---> PASS XXXX
230- Linux lima 2.4.22-1-686 #6 Sat Oct 4 14:09:08 EST 2003 i686 GNU/Linux
230 User ogluck logged in.
---> SYST
215 UNIX Type: L8 (Linux)
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> lcd /tmp
Local directory now /tmp
ftp> cd FTP
---> CWD FTP
250 CWD command successful.
ftp> ls
---> PORT 127,0,0,1,133,83
200 PORT command successful.
---> LIST
150 Opening ASCII mode data connection for '/bin/ls'.
total 4
-rw-r--r--   1 ogluck       2692 Mar 12 17:12 ftp.log
226 Transfer complete.
```


73

Exemple de dialogue FTP

```
ftp> get ftp.log
local: ftp.log remote: ftp.log
---> TYPE I
200 Type set to I.
---> PORT 127,0,0,1,133,84
200 PORT command successful.
---> RETR ftp.log
150 Opening BINARY mode data connection for 'ftp.log' (2692 bytes).
226 Transfer complete.
2692 bytes received in 0.00 secs (45326.0 kB/s)
ftp> put ftp.log ftp2.log
local: ftp.log remote: ftp2.log
---> PORT 127,0,0,1,133,85
200 PORT command successful.
---> STOR ftp2.log
150 Opening BINARY mode data connection for 'ftp2.log'.
226 Transfer complete.
2692 bytes sent in 0.00 secs (90651.9 kB/s)
ftp> mkdir TOTO
---> MKD TOTO
257 "TOTO" directory created.
ftp> ls
---> TYPE A
200 Type set to A.
---> PORT 127,0,0,1,133,86
200 PORT command successful.
```


74

Exemple de dialogue FTP

```
---> LIST
150 Opening ASCII mode data connection for '/bin/ls'.
total 12
drwxr-x---    2 ogluck       4096 Mar 12 17:17 TOTO
-rw-r--r--    1 ogluck       2692 Mar 12 17:12 ftp.log
-rw-r-----    1 ogluck       2692 Mar 12 17:16 ftp2.log
226 Transfer complete.
ftp> passive
Passive mode on.
ftp> get ftp2.log
local: ftp2.log remote: ftp2.log
---> TYPE I
200 Type set to I.
---> PASV
227 Entering Passive Mode (127,0,0,1,133,87)
---> RETR ftp2.log
150 Opening BINARY mode data connection for 'ftp2.log' (2692 bytes).
226 Transfer complete.
2692 bytes received in 0.00 secs (53651.1 kB/s)
ftp> bye
---> QUIT
221 Goodbye.
ogluck@lima:~$
```


75

TFTP : Trivial File Transfer Protocol

- Transfert de fichiers au-dessus d'UDP, port 69

```
ogluck@lima:~$ grep tftp /etc/services
tftp            69/udp
ogluck@lima:~$ grep tftp /etc/inetd.conf
tftp            dgram   udp     wait    nobody
  /usr/sbin/tcpd  /usr/sbin/in.tftpd /tftpboot
```

- Pourquoi TFTP ?
  - TFTP, c'est en gros FTP sans pouvoir lister les répertoires distants et ne nécessitant pas de mot de passe pour récupérer ou déposer des fichiers
  - protocole léger donc facilement implantable par des systèmes sans disque (en ROM) qui utilisent TFTP au boot pour récupérer un fichier de configuration… (terminaux X, imprimantes réseau, routeurs Cisco…)
  - UDP car ces systèmes n'implantent pas forcément TCP

77

TFTP : Trivial File Transfer Protocol

- Protocole léger - RFC 1350
  - pas de contrôle d'accès
  - 5 types de messages seulement
  - fiabilité assurée par acquittement positif avec timer de retransmission sur l'émetteur et le récepteur
    - les messages DATA font 512 octets max ; ils sont numérotés et sont aussitôt acquittés par un ACK
- Comme il n'y a pas d'authentification, les accès sur le serveur sont limités aux répertoires passés en arguments du démon `tftpd` (`/tftpboot` par défaut)

78

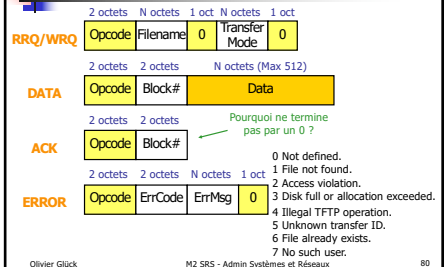
TFTP : types de messages

- Les 5 types de messages

Opcode	Opération	Description
1	RRQ	Read request
2	WRQ	Write request
3	DATA	Data
4	ACK	Acknowledgment
5	ERROR	Error (sert aussi d'ACK)

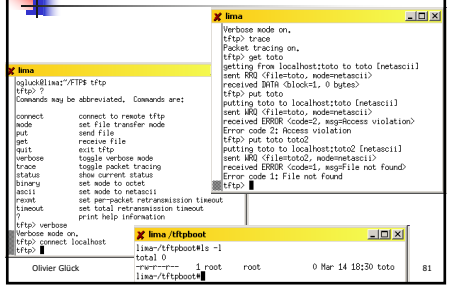
79

TFTP : format des messages



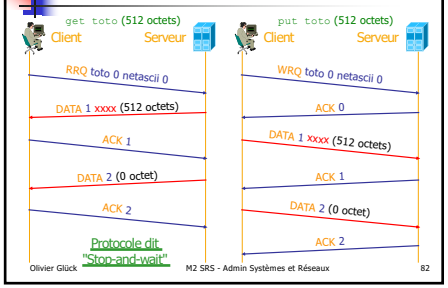
80

TFTP : commandes utilisateurs



81

TFTP : exemples d'échanges



82

Accès aux fichiers distants

Les protocoles NFS et SMB

83

Accès aux fichiers distants

- Différences avec le transfert de fichiers
  - l'accès aux fichiers distants est complètement transparent pour l'utilisateur
  - tout se passe comme si le système de fichiers distant était local
  - l'utilisateur peut éditer le fichier, le modifier, … ; les modifications seront répercutées sur le système de fichiers distant
- Les deux principaux protocoles
  - NFS : *Network File System* (Unix/Sun-RPC)
  - SMB : *Server Message Block* (issu du monde Microsoft)

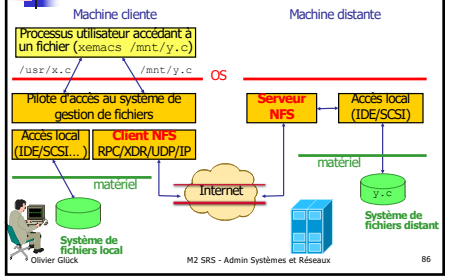
84

NFS : Network File System

- Présenté par Sun en 1985 pour permettre à ses stations sans disque d'accéder à un système de gestion de fichiers distant (RFC 1094)
- Utilise les appels de procédures distantes Sun-RPC (qui sont issues des travaux sur NFS)
  - a priori, les client et serveur NFS devraient être des processus utilisateur s'exécutant au-dessus de RPC/XDR/UDP/IP
  - en fait, le client et serveur NFS s'exécutent dans le noyau
    - le client pour rendre transparent l'accès à un fichier via NFS (pas de différence d'utilisation d'un fichier local et d'un fichier distant)
    - le serveur pour des raisons d'efficacité

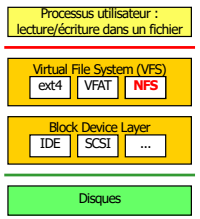
85

NFS : principe de fonctionnement



86

Les éléments d'accès aux fichiers



- Manipule des chemins, descripteurs, déplacements
- Manipule des Files, Dentries, Inodes, déplacements
- Masque les différences à l'application (API uniforme, ...)
- Manipule des blocks
- Matériel-dépendant

Le choix entre ext4, NTFS,VFAT, NFS, ... se fait lors de l'ouverture du fichier

87

NFS : en pratique...

Client NFS

```
root@192.168.69.1# tail -1 /etc/fstab
192.168.69.2:/home  /nfshome  nfs  defaults,noauto 0 0
root@192.168.69.1# mkdir /nfshome
root@192.168.69.1# mount /nfshome
root@192.168.69.1# ls -l /nfshome
drwxr-xr-x    2 1003     5000         1024 fév 16 13:32 olivier
root@192.168.69.1# df
Filesystem           1k-blocks      Used Available Use% Mounted on
/dev/hda2              3898108   2766592    930304  75%   /
192.168.69.2:/home    16128636   1493328  13815996  10%   /nfshome
```

/nfshome

open(),read(),write()

```
root@192.168.69.2# cat /etc/exports
# /etc/exports: the access control list for filesystems which
# may be exported to NFS clients.  See exports(5).
/home 192.168.69.0/255.255.255.0(rw,root_squash,async)
```

/home

Serveur NFS

88

NFS : en pratique...

Client NFS

```
root@192.168.69.1# rpcinfo -u 192.168.69.2 showmount
program 100005 version 3 ready and waiting
root@192.168.69.1# rpcinfo -u 192.168.69.2 nfs
program 100003 version 3 ready and waiting
root@192.168.69.1# showmount -a 192.168.69.2
All mount points on 192.168.69.2:
192.168.69.1:/home
root@192.168.69.1# showmount -d 192.168.69.2
Exported directories on 192.168.69.2:
/home
root@192.168.69.1# showmount -e 192.168.69.2
Export list for 192.168.69.2:
/home 192.168.69.0/255.255.255.0
```

/nfshome

```
root@192.168.69.2# showmount
Hosts on 192.168.69.2:
192.168.69.1
```

/home

Serveur NFS

89

NFS et la sécurité

- Principe d'authentification :
  - `(uid, gid)local` est mappé en `(uid, gid)distant` --> équivalence entre les droits locaux et distants
  - problème pour `root` :
    - quels droits possède le `root` d'une machine cliente sur les fichiers exportés par un serveur NFS ?
    - par défaut, `root` est mappé en `nobody` pour des raisons de sécurité (sinon il faut mettre `no_root_squash` dans `/etc/exports`)
- Règle de non transitivité de NFS :
  - si A exporte `/home` à B ; si B monte `A:/home` dans `/home2` et exporte `/home2` à C alors C n'aura pas accès au `/home` de A
  - sinon il n'y aurait aucun contrôle possible des règles d'exportation donc pas de sécurité... (B pourrait exporter à C alors que A ne veut pas autoriser C)

90

NFS : les procédures distantes (v2)

- `GETATTR` : retourne les attributs d'un fichier (type, taille, permissions, propriétaire, date de dernière modification, ...)
- `SETATTR` : modifie les attributs d'un fichier
- `STATFS` : retourne l'état d'un SGF (espace libre, ...)
- `LOOKUP` : recherche le fichier dont le nom et en argument, retourne un *file handle* et les attributs (appelée à l'ouverture du fichier)
- `READ` : lecture dans le fichier d'au plus N octets à partir d'un certain offset (N doit être inférieur à 8192)
- `WRITE` : écriture dans le fichier de N octets à partir d'un certain offset, les données se trouvant dans le tampon @
- `CREATE`/`REMOVE`/`RENAME` : créer, supprimer, renommer
- `LINK`/`SIMLINK`/`READLINK` : création/lecture d'un lien
- `MKDIR`/`RMDIR`/`READDIR` : manipulation des répertoires

91

MOUNT : les procédures distantes

- Procédures distantes utilisées par `mount` et `showmount`
  - `MNT` : montage d'une partition distante, retourne un *file handle* correspondant au répertoire monté (racine)
  - `DUMP` : retourne la liste de tous les *file system* montés
  - `UMNT` : supprime un point de montage
  - `UMNTALL` : supprime tous les points de montage pour ce client
  - `EXPORT` : retourne les informations sur les *file system* qui sont exportés

92

NFS : procédures non-idempotentes

- Toutes les opérations précédentes ne sont pas idempotentes : deux exécutions identiques successives ne donnent pas le même résultat
- Ne sont pas idempotentes `CREATE`, `REMOVE`, `RENAME`, `LINK`, `SIMLINK`, `MKDIR`, `RMDIR`
- Solution :
  - le serveur utilise un cache des requêtes/réponses récentes pour les procédures non-idempotentes
  - quand une nouvelle requête arrive, le serveur regarde dans son cache si la réponse est déjà présente, auquel cas il renvoie cette réponse sans re-exécuter la proc.

93

NFS : un serveur sans état

- Le serveur NFS est sans état : entre deux requêtes, il ne conserve aucune information sur
  - les accès précédents à un fichier donné,
  - les fichiers ouverts, ...
  - le `LOOKUP` correspond au `open()` mais il n'y a pas de procédure correspondant au `close()`
  - après un `LOOKUP`, le serveur ne sait pas si le client va effectivement "utiliser" le fichier ou non
- Pourquoi sans état ?
  - en cas de crash du serveur NFS
    - permet de simplifier son redémarrage
    - transparent pour le client : il n'a pas besoin de réitérer certaines requêtes en cas de crash

94

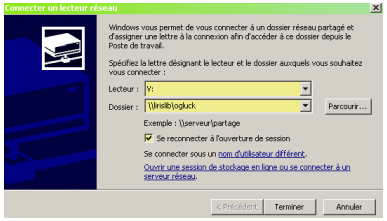
NFS au dessus de TCP

- En principe, pour plus de réactivité, NFS utilise les RPC sur UDP (dans le cadre d'un LAN)
- Mais pour étendre NFS aux WAN, les dernières versions permettent d'exécuter NFS sur RPC/TCP
- Caractéristiques de NFS sur TCP
  - le serveur fait une ouverture passive sur le port TCP/2049
  - quand un client monte une partition NFS, cela se traduit par une ouverture active --> une connexion TCP par point de montage (soit une par *file system*)
  - toutes les applications qui utilisent ce système de fichiers partagent la même connexion TCP
  - plus résistant aux pannes du serveur NFS : le client essaie régulièrement de rétablir la connexion et conservent les requêtes RPC en attente

95

SMB : Server Message Block

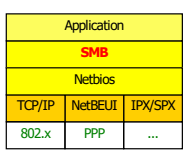
- Protocole de Microsoft et Intel permettant le partage de ressources (disques, imprimantes...) à travers un réseau (1987)

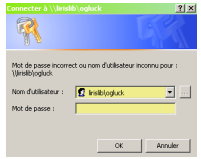


96

SMB : Server Message Block

- SMB est prévu pour être utilisé sur l'interface NetBIOS
  - utilise les noms NetBIOS (15 caractères + 1 pour le type)
  - utilise le mécanisme datagram de NetBIOS par *broadcast* comme service de nommage (NOM-->@MAC, pas d'adresse de niveau 3)





97

SMB : Server Message Block

- Chaque machine (client ou serveur) possède un nom sur 15 caractères
- SMB ajoute un 16ième caractère pour distinguer
  - les serveurs de fichiers et d'imprimantes, les clients, ...
- Notion de domaine
  - un ensemble d'utilisateurs (avec nom et mot de passe) et de serveurs (avec des droits d'accès)
  - un *primary domain server* contient la base de données des utilisateurs et de leurs mots de passe
- Un serveur partage une ou plusieurs ressources
  - fichiers, imprimantes, ...
  - à chaque triplet (domaine, serveur, ressource) correspond un nom unique `\\server\resource_name`

98

SMB : Server Message Block

- Deux niveaux de protection des accès :
  - au niveau de chaque utilisateur : basé sur le nom des utilisateurs (user, passwd), permet de gérer l'accès aux ressources voire aux éléments d'une ressource
  - au niveau de chaque ressource : un mot de passe commun à tous les utilisateurs est associé à une ressource pour y autoriser l'accès
- CIFS : *Common Internet File System*
  - dernière version de SMB proposant un meilleur passage à l'échelle (extensibilité)
  - divulgation du protocole SMB par Microsoft à l'IETF en 1996 sous ce nom --> a permis l'apparition de Samba

99

SMB : Server Message Block

samba opening windows to a wider world

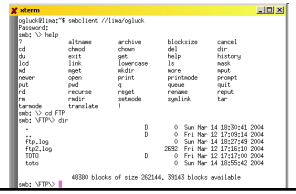
- Samba : implémentation de SMB sous Unix qui permet un partage de ressources entre les mondes Unix et Windows ; Samba permet de
  - partager un disque Unix pour des machines Windows
  - accéder à un disque Windows depuis une machine Unix
  - partager une imprimante Unix pour des machines Windows
  - utiliser une imprimante Windows à partir d'un hôte Unix
- Le serveur Samba sur la machine Unix émule un domaine SMB

100

SMB : Server Message Block

- Commandes Unix liées au serveur Samba
  - `smbpasswd` : permet de changer le mot de passe d'un utilisateur SMB
  - `smbclient` : permet d'interroger un serveur samba depuis Unix

```
smbclient -L host # liste les ressources offertes par le serveur
smbclient //host/ressource # permet l'accès à la ressource
```



101

Gestion d'utilisateurs distants

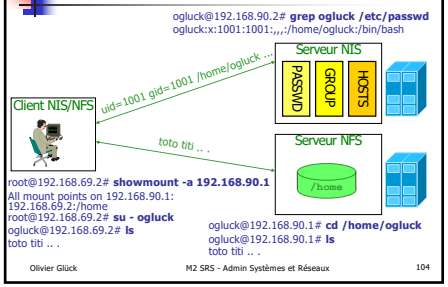
102

NIS : un annuaire répliqué

- NIS : *Network Information System*
  - introduit par SUN en 1985 (*Yellow Pages* (`yp`) à l'origine)
  - n'est pas un standard de l'Internet mais est largement utilisé
  - une base de données distribuée qui permet le partage d'informations système (`/etc/passwd`, `/etc/hosts`, ...)
- Objectif : réduire le temps d'administration d'un parc de machines
  - simplifier la gestion des comptes, des mots de passe et les travaux d'administration dans le monde Unix
  - typiquement, il suffit de créer un nouvel utilisateur sur le serveur NIS pour que chaque machine client NIS ait accès aux informations de *login* de cet utilisateur

103

NIS : un exemple courant



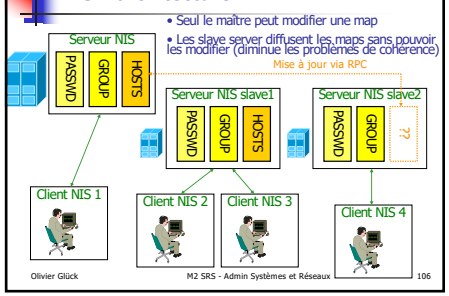
104

NIS : architecture

- Architecture : découpage en domaines
  - modèle Client/Serveur au dessus des SUN-RPC
  - un **domaine NIS** contient
    - un serveur NIS maître qui maintient les "*maps*" (informations contenues dans la base)
    - zéro, un ou plusieurs serveurs NIS esclaves :
      - permet de décharger le serveur principal et d'être plus résistant aux pannes
      - le maître réplique ses informations vers les serveurs secondaires
    - des clients NIS qui peuvent interroger les serveurs maître ou secondaires

105

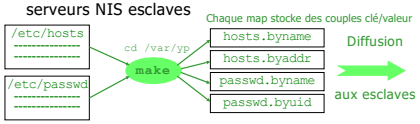
NIS : architecture



106

NIS : en pratique...

- Les maps sont stockées sur le serveur dans `/var/yp/nom-de-domaine`
- Quand le fichier source d'une map est modifié (sur le serveur), il faut régénérer la map associée et éventuellement propager les modifications aux serveurs NIS esclaves



107

NIS : en pratique...

- Savoir si les services sont en place

```
root@ 192.168.69.2# rpcinfo -u 192.168.90.2 ypserv
program 100004 version 1 ready and waiting
program 100004 version 2 ready and waiting
root@ 192.168.69.2# rpcinfo -u 192.168.69.2 ypbind
program 100007 version 1 ready and waiting
program 100007 version 2 ready and waiting
root@ 192.168.69.2# ypwhich
192.168.90.2
```

- Contrôle de l'accès au serveur NIS

```
root@ 192.168.90.2# cat /etc/ypserv.securenets
#This file defines the access rights to your NIS server
255.0.0.0       127.0.0.0
255.255.255.0   192.168.69.0
```


112

NIS : évolutions

- Défauts des NIS
  - pas d'authentification des clients NIS : il suffit de connaître le nom de domaine pour interroger le serveur et connaître le contenu de ses maps
  - les maps sont transmises en totalité même en cas de faible modification de leurs contenus
  - pas adapté aux WAN (*broadcast*...)
- NIS+ un successeur éphémère sans succès qui a été officiellement abandonné au profit de LDAP
- Cependant, les NIS sont encore largement utilisés dans le cadre d'un réseau local simple

114

Partie 2 : Applications de l'Internet de type Client/Serveur (suite1)

Olivier GLÜCK
Université LYON 1/UFR d'Informatique
Olivier.Gluck@ens-lyon.fr
http://www710.univ-lyon1.fr/~ogluck

115

Plan de la partie 2

- Introduction / Rappel
- Connexions à distance (telnet/rlogin/rsh/ssh/X11)
- Applications de transfert de fichiers (FTP/TFTP)
- Accès aux fichiers distants (NFS/SMB)
- Gestion d'utilisateurs distants (NIS)
- **DNS : un annuaire distribué**
- LDAP : un annuaire fédérateur sécurisé
- La messagerie électronique (SMTP/POP/IMAP)

116

DNS : un annuaire distribué des adresses de l'Internet

Le système DNS
Une base de données distribuée
Notions de zones et domaines
Les différents types de serveurs
Résolutions récursives et itératives
Cache DNS, Format des messages DNS
Commandes et fichiers liés au DNS

117

DNS : Domain Name System

- Gens : plusieurs identifiants
  - #sécu, nom, #Passeport
- Hôtes, routeurs :
  - adresse IP (32 bits)
  - "nom" :
    - `www.google.com`
    - `www.education.gouv.fr`
- Problème résolu par le DNS : Comment relier les adresses IP utilisées pour acheminer les paquets aux noms utilisés par les utilisateurs ou les applications ?

118

DNS : Domain Name System

- C'est une base de données **distribuée**
  - implantée dans une hiérarchie de serveurs de noms
- C'est un protocole applicatif
  - les hôtes, routeurs, serveurs de noms communiquent pour effectuer la traduction
  - DNS est utilisé par d'autres protocoles applicatifs mais n'est pas utilisé directement par l'application comme SMTP...
  - modèle Client/Serveur : un émetteur interroge un serveur de noms (serveur DNS)
  - port 53/UDP (ou 53/TCP pour les mises à jour)
  - RFC 1034, 1035, 2181, ...

119

Les services fournis par le DNS

- Le service principal : la traduction d'adresses
- Autres services :
  - permettre le "*Host aliasing*" : donner un pseudonyme à une machine qui a un nom peu parlant
  - permettre le "*Mail server aliasing*" : un serveur Web et un serveur Mail peuvent avoir le même pseudonyme même s'ils n'ont pas la même adresse IP (2 machines ≠)
  - permettre la répartition de la charge : un nom de serveur Web ou Mail peut correspondre à plusieurs adresses IP (serveurs Web ou Mail répliqués) avec un système de rotation dans les réponses du serveur DNS
- Pour l'utilisateur, le DNS n'est qu'une boîte noire mais en réalité très compliquée
  - une requête DNS peut impliquer plusieurs serveurs de noms répartis dans le monde entier

120

Un système centralisé ?

- Pourquoi pas de DNS centralisé ? Un seul serveur contiendrait toutes les correspondances requises par les applications de l'Internet
  - dimension de l'Internet : trop de correspondances à gérer, nombre de requêtes au serveur trop important
  - tolérance aux pannes : si le serveur DNS tombe, tout l'Internet aussi !
  - volume de trafic impossible à supporter par un seul serveur
  - délais de réponse : il faut faire en sorte que la réponse soit la plus proche possible du demandeur
  - problème lié à la maintenance et aux mises à jour perpétuelles de la base

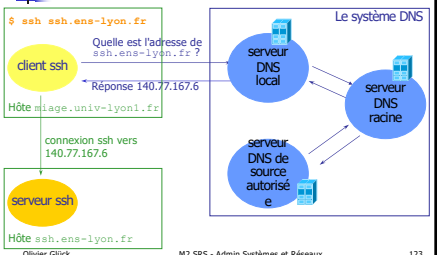
121

Un système distribué

- Aucun serveur ne connaît toutes les correspondances nom <--> adresse IP
  - si un serveur ne connaît pas une correspondance, il interroge un autre serveur jusqu'à atteindre le serveur détenant l'information désirée
- Trois types de serveur DNS
  - les serveurs de noms locaux (à qui s'adressent les requêtes locales ; en charge de la résolution)
  - les serveurs de noms racine (sont censés savoir comment s'approcher de la réponse)
  - les serveurs de noms de source autorisée (contiennent les correspondances "officielles")

122

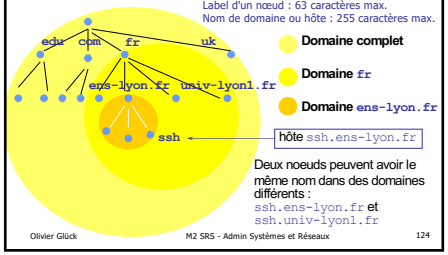
Un système distribué



123

La notion de domaine DNS

Un domaine est un sous-arbre entier de l'espace de nommage



124

La notion de domaine DNS

- Le premier niveau de l'arbre
  - *Top Level Domain* (TLD)
  - géré par l'ICANN (*Internet Corporation for Assigned Names and Numbers*)
  - on distingue deux catégories de TLD

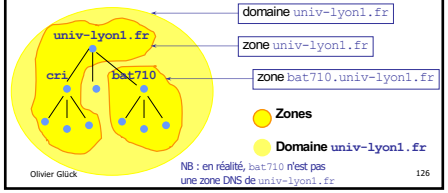
    les "*Generic TLD*" : `.com`, `.org`, `.gov`, `.gouv`, `.net`, ...

    les "*Countries TLD*" : `.fr`, `.uk`, `.us`, `.jp`, ... (240 en tout)
- La gestion des autres niveaux est laissée aux entités correspondantes (AFNIC pour `.fr`)
  - zone DNS : un sous-arbre de l'arbre administré séparément par un organisme qui gère la délégation des noms et sous-domaines de la zone

125

La notion de zone DNS

- Une zone = une administration centralisée avec au moins un serveur DNS (généralement 1 primaire et 1 secondaire)
  - le secondaire (redondance) met à jour ses données à partir du primaire
- Une zone doit connaître les adresses des serveurs DNS des zones subordonnées

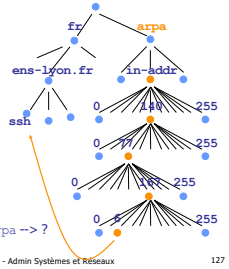


126

La résolution de noms inverse

- Retrouver le nom canonique à partir de l'adresse IP
- Le domaine `arpa` : un domaine particulier géré par l'ICANN permettant la résolution inverse
- Résolution `ssh.ens-lyon.fr` --> ?
- Résolution inverse

`6.167.77.140.in-addr.arpa` --> ?



127

Les différents types de serveur DNS

- Les serveurs de noms locaux
  - chaque organisation a un serveur de noms local
    - serveur DNS par défaut de la zone
    - contient parfois les correspondances relatives à la zone de l'organisation
  - toutes les requêtes DNS en provenance de cette organisation vont vers ce serveur de nom local
- Les serveurs de noms racine [RFC 2870]
  - il existe actuellement 13 serveurs racine dans l'Internet (liste sur `http://gnso.icann.org/gtld-registries/`)
  - chaque serveur DNS local connaît un serveur de noms racine qu'il peut interroger s'il ne connaît pas une correspondance
  - un serveur de noms racine connaît au moins les serveurs de source autorisée du premier niveau (`.fr.`, …)

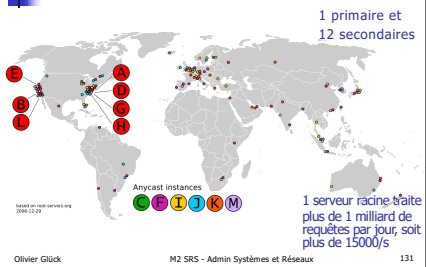
128

Les différents types de serveur DNS

- Un serveur de noms racine qui ne connaît pas une correspondance interroge un autre serveur de noms le rapprochant de la réponse, généralement le serveur de noms de source autorisée qui connaît la correspondance
- Les serveurs de noms de source autorisée
  - chaque hôte est enregistré auprès d'au moins deux "authoritative servers" (le primaire et le secondaire), qui stocke son adresse IP et son nom
  - un serveur de noms est dit de source autorisée pour un hôte s'il est responsable de la correspondance nom/@ pour cet hôte (serveur primaire de la zone)
  - un serveur de noms local n'est pas forcément de source autorisée (ex. `bat710.univ-lyon1.fr`)

129

Les différents types de serveur DNS



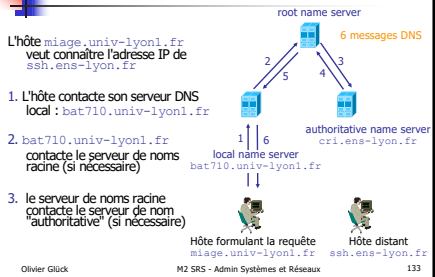
131

Résolution de noms récursive/itérative

- Requête récursive
  - la machine qui demande la résolution de nom contacte un serveur DNS et attend que ce dernier lui retourne la réponse désirée
- Requête itérative
  - le serveur de noms contacté fournit en réponse le nom d'un autre serveur DNS à contacter pour avancer dans la résolution
  - "je ne connais pas ce nom mais demande à ce serveur"
- Dans le cheminement d'une résolution de nom, certaines requêtes peuvent être itératives, d'autres récursives

132

Principe d'une résolution récursive

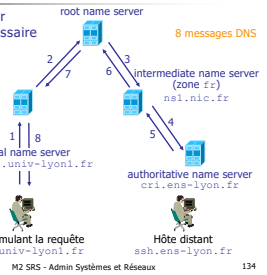


133

Principe d'une résolution récursive

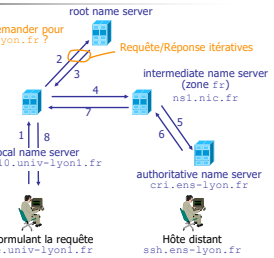
Cas où un serveur intermédiaire est nécessaire

- Le serveur de noms racine ne connaît pas forcément le serveur de source autorisée qui le renseignera sur la correspondance recherchée
- Passage par un ou plusieurs serveurs de noms intermédiaires



134

Principe d'une résolution itérative



- Généralement les requêtes sont toutes récursives sauf celle entre le serveur de noms local et le serveur de noms racine
- --> permet de moins solliciter le serveur racine (moins exigeant en termes de ressources)

135

La mise en mémoire cache DNS

- Idée générale des caches : réduire le temps de réponse
  - ici, le temps de réponse d'une résolution de nom, en diminuant le nombre de messages DNS en transit nécessaires !
  - le serveur de noms (quelconque) stocke dans son cache les informations récentes (en particulier les enregistrements de type NS)
    - comme la mémoire n'est pas infinie et que les données du cache peuvent ne plus être valables au bout d'un certain temps, les données expirent du cache après un certain temps TTL (environ 2 jours)
  - un serveur qui mémorise dans son cache un enregistrement DNS n'a pas autorité dessus --> spécifie "*no authoritative*" dans la réponse

136

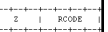
Les messages DNS [RFC 1034, 1035]

- Un message de réponse DNS contient un ou plusieurs RR (*Resource Record*)
  - l'unité de stockage d'une correspondance dans le cache
  - `RR = (Nom, Type, Classe, TTL, Valeur)`
  - `Type` (16 bits) : type de l'enregistrement ; la signification de `Nom` (N octets) et `Valeur` dépend du type
  - `Classe` (16 bits) : famille de protocoles ; `Classe=1` pour Internet (`IN`)
  - `TTL` (32 bits) : temps en secondes pendant lequel il faut conserver l'enregistrement dans le cache
  - `Valeur = RDLENGTH` (16 bits) + `RDATA` (N octets)
- Un message de requête DNS contient des questions du type `(Nom, Type, Classe)`

138

Les messages DNS [RFC 1034, 1035]

- Un unique format pour les Requêtes/Réponses
- 12 octets d'en-tête DNS
  - un identifiant qui permet au client d'associer la réception d'une réponse à une requête formulée
  - les fanions : 2 octets



`QR` : 0 si requête, 1 si réponse
`OPCODE` (4 bits) : 0 (QUERY), 1 (Inverse QUERY), ...
`AA` : 1 si le serveur de noms qui répond a autorité sur la réponse sollicitée
`TC` : le message a été tronqué --> utiliser TCP
`RD` : 1 dans la requête si le client souhaite que le serveur de noms effectue une récursion s'il ne dispose pas de la traduction demandée
`RA` : 1 dans la réponse si le serveur de noms contacté assure la récursion
`AD` et `CD` (dans `Z`) : "*Authentic Data*" et "*Checking Disabled*" (DNSSEC)
`RCODE` (4 bits) : 0 (pas d'erreur), 3 (Name error), 5 (Refused), ...

139

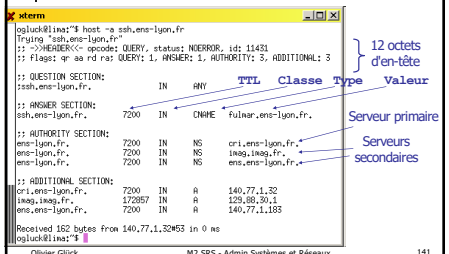
Les messages DNS [RFC 1034, 1035]

512 octets max. avec UDP

Identification	Fanions	12 octets d'en-tête
Nombre de questions	Nombre de réponses (RR)	
Nombre de "authority servers"	Nombre de RR supplémentaires	
Questions (nombre variable de questions)		QUESTION SECTION
Réponses (RRs répondant à la demande)		ANSWER SECTION
Noms des serveurs de source autorisée (RRs des serveurs primaire/secondaire)		AUTHORITY SECTION
Adresses des serveurs de source autorisée (nb. variable de RRs)		ADDITIONAL SECTION

140

Les messages DNS [RFC 1034, 1035]



141

Les messages DNS [RFC 1034, 1035]

- **Type=A** (val=1) : sert à décrire une correspondance
  `Nom`=nom d'hôte (canonique), `Value`=@IPv4
- **Type=AAAA** (val=28, RFC 1886) : idem mais adresse IPv6
  `Nom`=nom d'hôte, `Value`=@IPv6
- **Type=PTR** (val=12) : sert à la résolution inverse
  `Nom`=un nom de la zone `arpa`, `Value`=nom canonique (valeur pointée)
- **Type=NS** (val=2) : sert à associer un nom de domaine à un serveur de noms de source autorisée
  `Nom`=domaine, `Value`=nom du serveur de noms
- **Type=CNAME** (val=5) : sert à définir un alias pour un hôte
  `Nom`=un alias, `Value`=nom canonique (le vrai nom)

142

Les messages DNS [RFC 1034, 1035]

- **Type=MX** (val=15) : alias réservés aux serveurs mail permettant d'associer plusieurs serveurs de mail avec différentes priorités à une même adresse (RFC 974)
  `Nom`=un alias, `Value`=nom canonique d'un serveur de mail
- **Type=SOA** (val=6) : sert à donner des infos sur la zone gérée
  `Nom`=nom d'une zone, `Value`=informations sur la zone
- **Type=ANY** (val=255) : utilisé dans les questions pour indiquer n'importe quel type (*)
- **Type=AXFR** (val=252) : utilisé dans les questions pour demander le transfert d'une zone entière (mise à jour d'un serveur secondaire…)
- **Type=HINFO** (val=13) : sert à indiquer les CPU et OS de l'hôte interrogé

143

Les messages DNS [RFC 1034, 1035]

- Exemples

```
ssh.ens-lyon.fr.            CNAME   fulmar.ens-lyon.fr.
ens-lyon.fr.                NS      cri.ens-lyon.fr.
ens-lyon.fr.                NS      ens.ens-lyon.fr.
cri.ens-lyon.fr.            A       140.77.1.32
relaissmtp.ens-lyon.fr.     CNAME   pluvier.ens-lyon.fr.
ens-lyon.fr.                MX      20 pluvier.ens-lyon.fr.
ens-lyon.fr.                MX      30 pluvier2.ens-lyon.fr.
listes.ens-lyon.fr.         MX      20 pluvier.ens-lyon.fr.
fulmar.ens-lyon.fr.         A       140.77.167.6
6.167.77.140.in-addr.arpa.PTR       fulmar.ens-lyon.fr
```

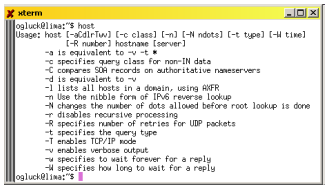

144

Routage de courrier et DNS [RFC 974]

- Centraliser la réception des messages sur une machine qui a un système plus robuste
  - anti-virus, anti-spam, …
  - seule machine accessible sur le port 25 depuis l'extérieur via le pare-feu
- Les MX permettent ensuite de répartir la charge sur différents serveurs de mail et de disposer de serveurs de secours
  - en cas de saturation, le serveur SMTP peut aiguiller les messages via un autre serveur SMTP interne

145

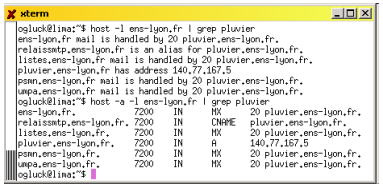
La commande `host`




```
ogluck@lima:~$ host ssh.ens-lyon.fr
ssh.ens-lyon.fr is an alias for fulmar.ens-lyon.fr.
fulmar.ens-lyon.fr has address 140.77.167.6
```


146

La commande `host`

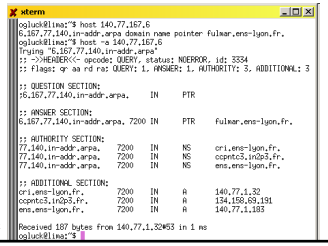


- Pour connaître les serveurs de source autorisée d'une zone :
  `host -a nom_zone` **ou** `host -t ns nom_zone`

147

Les messages DNS - type PTR

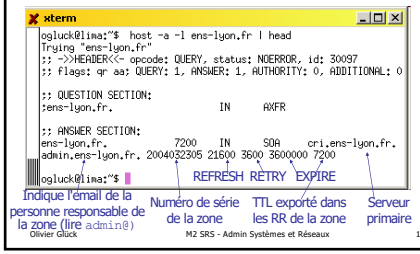
`6.167.77.140.in-addr.arpa` est un pointeur vers `fulmar.ens-lyon.fr`



148

Les messages DNS - types AXFR, SOA

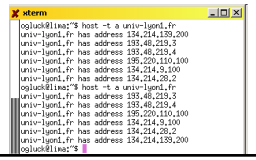
Demande du contenu de toute la zone `ens-lyon.fr`



149

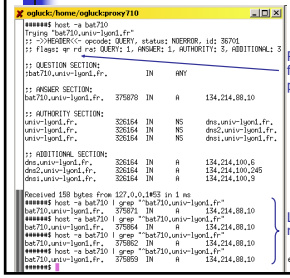
Le DNS Round-robin

- Les alias : plusieurs noms correspondent à une même adresse IP
- Le Round-robin : à un même nom correspond plusieurs adresses IP --> permet d'avoir de la redondance (plusieurs RRs de type A)
- Le DNS change l'ordre à chaque nouvelle requête pour répartir la charge



150

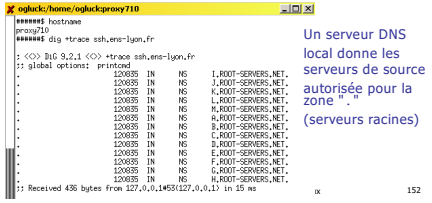
La mise en mémoire cache



151

La commande `dig`

- Rend les mêmes services que `host` mais est encore plus bas niveau : permet en particulier de voir l'ensemble des requêtes/réponses

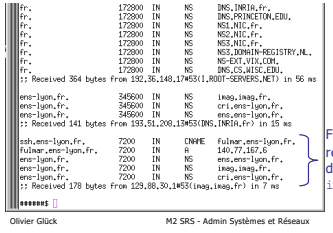


152

La commande `dig`

- Suite...

Le serveur racine `I` donne les serveurs de source autorisée pour la zone "`fr.`"



153

Du côté client… - le *resolver*

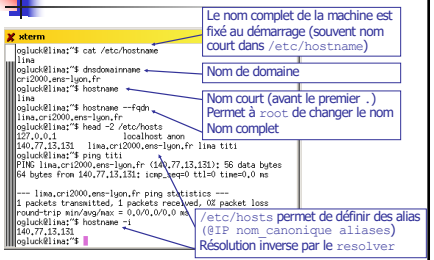
- Le *resolver* a en charge les résolutions de noms (inverse ou pas) chaque fois que cela est nécessaire --> `man resolver`
- Deux fichiers de configuration lui sont associés
  - `/etc/resolv.conf` permet de paramètrer les requêtes DNS effectuées (`man resolv.conf`)
  - `/etc/host.conf` permet de paramètrer le *resolver* (`man host.conf`), en particulier ordre de résolution

    `order hosts,bind,nis`

    `/etc/nsswitch.conf` est consulté en premier s'il existe
- Extrait de l'API du *resolver* pour les applications
  - `gethostbyname(name)`
  - `gethostbyaddr(addr)`

154

Du côté client… - les commandes

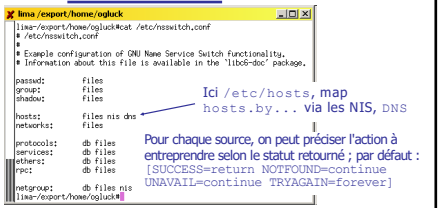


155

Le fichier `/etc/nsswitch.conf`

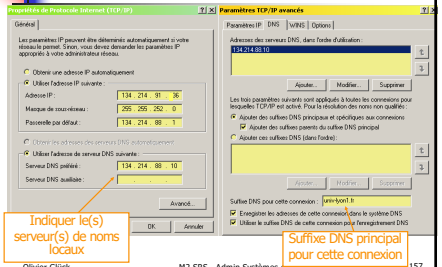
Permet de spécifier l'ordre des méthodes de résolutions (ligne `hosts` pour la résolution de noms)

`man nsswitch.conf`



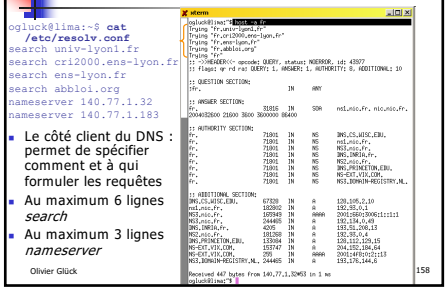
156

Configuration d'un poste de travail



157

Le fichier `/etc/resolv.conf`

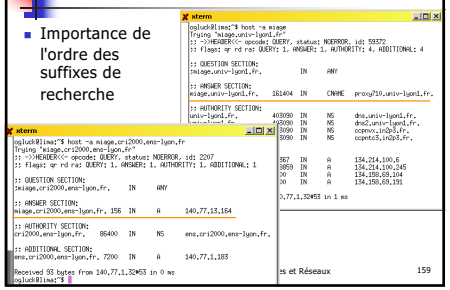


- Le côté client du DNS : permet de spécifier comment et à qui formuler les requêtes
- Au maximum 6 lignes *search*
- Au maximum 3 lignes *nameserver*

158

Le fichier `/etc/resolv.conf`

- Importance de l'ordre des suffixes de recherche



159

Le serveur de noms named (BIND)

- BIND : Berkeley Internet Name Domain
  - http://www.isc.org/sw/bind
  - implantation d'un serveur DNS du domaine publique
- Le démon répondant aux requêtes DNS est named
  - fichier de configuration : named.conf
  - il faut y associer les fichiers décrivant les zones administrées (syntaxe master files : voir RFC 1035)
  --> dans /etc/namedb ou /etc/bind
- Des utilitaires
  - rndc permet de contrôler à distance le fonctionnement de named (avec authentification)
  - named-checkconf et named-checkzone permettent de vérifier la syntaxe des fichiers de zones ou config.

160

Le fichier named.conf

man named.conf

```
######$ cat /etc/bind/named.conf
// This is the primary configuration file
// for the  BIND DNS server named.
options {
        // répertoire de travail de named
        directory "/var/cache/bind";
        // si le serveur n'a pas la réponse
        // il forward à un autre
        forward first;
        forwarders {
        134.214.88.23;
        134.214.88.10;
                };
};
// prime the server with knowledge
// of the root servers
zone "." {
        // copie m.a.j. au démarrage
        type hint;
        file "/etc/bind/db.root";
};
// be authoritative for the localhost
// forward and reverse zones, and
// for broadcast zones as per RFC 1912
zone "localhost" {
        type master; // autorité pour la zone
        file "/etc/bind/db.local";
};
zone "127.in-addr.arpa" {
        type master;
        file "/etc/bind/db.127";
};
zone "0.in-addr.arpa" {
        type master;
        file "/etc/bind/db.0";
};
zone "255.in-addr.arpa" {
        type master;
        file "/etc/bind/db.255";
};
// add entries for other zones below here
// ...........................
```


161

Les fichiers décrivant une zone

RFC 1035

```
######$ cat /etc/bind/db.root
;       This file holds the information on root name servers needed to
;       initialize cache of Internet domain name servers
;       This file is made available by InterNIC ...
;           Sur ftp://ftp.rs.internic.net/domain/named.root
.                       3600000  IN  NS    A.ROOT-SERVERS.NET.
A.ROOT-SERVERS.NET.     3600000      A     198.41.0.4
;
.                       3600000      NS    B.ROOT-SERVERS.NET.
B.ROOT-SERVERS.NET.     3600000      A     128.9.0.107
;
; formerly C.PSI.NET
;
.                       3600000      NS    C.ROOT-SERVERS.NET.
C.ROOT-SERVERS.NET.     3600000      A     192.33.4.12
; ...................................
```


162

Les fichiers décrivant une zone

```
######$ cat /etc/bind/db.local
; BIND data file for local loopback interface
$TTL    604800
@       IN      SOA     localhost. root.localhost. (
                              1         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                         604800 )       ; Negative Cache TTL
;
@       IN      NS      localhost.
@       IN      A       127.0.0.1
######$ cat /etc/bind/db.127
; BIND reverse data file for local loopback interface
$TTL    604800
@       IN      SOA     localhost. root.localhost. (
                              1         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                         604800 )       ; Negative Cache TTL
;
@       IN      NS      localhost.
1.0.0   IN      PTR     localhost.
```

@ désigne le nom de la zone (ici localhost)

@ désigne le nom de la zone (ici 127.in-addr.arpa)

1.0.0.127.in-addr.arpa

163

Partie 2 : Applications de l'Internet de type Client/Serveur (suite2)

Olivier GLÜCK
Université LYON 1/UFR d'Informatique
Olivier.Gluck@ens-lyon.fr
http://www710.univ-lyon1.fr/~ogluck

164

Plan de la partie 2

- Introduction / Rappel
- Connexions à distance (telnet/rlogin/rsh/ssh/X11)
- Applications de transfert de fichiers (FTP/TFTP)
- Accès aux fichiers distants (NFS/SMB)
- Gestion d'utilisateurs distants (NIS)
- DNS : un annuaire distribué
- **LDAP : un annuaire fédérateur sécurisé**
- La messagerie électronique (SMTP/POP/IMAP)

165

LDAP : un annuaire fédérateur sécurisé

166

Problématique résolue par LDAP

- Permettre la fusion de multiples BD dans un unique annuaire informatique
  - base Microsoft Excel du personnel administratif
  - base Microsoft Access du personnel enseignant
  - base Microsoft Excel des numéros de téléphone
  - base `/etc/passwd` des comptes Unix des utilisateurs
  - base `/etc/aliases` (ou Sympa) de listes de Mail
  - base Samba des utilisateurs Windows
  - autres bases MySQL, Oracle, maps NIS,…
- Comment envoyer un mail à l'ensemble du personnel administratif sachant que l'administrateur système recevra uniquement une liste de (Nom, Prénom) ?

167

Le concept d'annuaire

- Annuaire informatique
  - service permettant d'accéder à des informations relatives à des personnes, des machines (ou autres ressources) de manière organisée
  - objectif : maintenir de façon cohérente et contrôlée une grande quantité de données
- Système de gestion de base de données (SGBD)
  - le schéma des données stockées est défini pour résoudre un certain problème ; il est connu des applis
  - les objets sont généralement complexes, stockés dans différentes tables ayant des relations entre elles
  - un langage spécifique permet la lecture et mise à jour des tables (requêtes SQL, …)

168

Le concept d'annuaire

- Différences annuaire/SGBD - dans un annuaire :
  - pas de liens de dépendances entre les objets stockés
  - les objets peuvent être distribués sur plusieurs annuaires pour assurer une meilleure disponibilité
  - le schéma de stockage des données est standardisé pour assurer un partage des données
  - les applications de l'annuaire n'ont pas besoin de connaître la structure interne des données stockées
  - un annuaire est principalement consulté en lecture et optimisé pour cela

169

L'annuaire LDAP

- LDAP : *Lightweight Directory Access Protocol*
- Héritier de l'annuaire X500 (proposée par l'ISO)
  - standard conçu par les opérateurs télécom pour interconnecter leurs annuaires téléphoniques
  - X500 adapté à Internet --> LDAP (même modèle de schéma, …)
- Proposé à l'IETF en 1995
  - standard d'annuaire sur TCP/IP
    - le standard ne concerne pas le contrôle d'accès aux données de l'annuaire
  - Version 3 actuellement [RFC 2251]
  - Aussi : RFC 2252 à 2256, RFC 2829 à 2830, RFC 2849

170

L'annuaire LDAP

- Objectifs
  - fournir aux utilisateurs des informations fiables, facilement accessibles
  - permettre aux utilisateurs de mettre à jour eux-mêmes leurs informations personnelles
  - rendre les informations accessibles de façon contrôlée
  - faciliter le nomadisme des utilisateurs
  - éviter la redondance d'informations : un seul annuaire pour l'ensemble des services
  - faciliter la gestion (administration) des postes de travail, des équipements réseau

sans remettre en cause les applications existantes !

171

L'annuaire LDAP

- Un modèle d'information : type des informations contenues dans l'annuaire
- Un protocole d'accès : comment accéder aux informations contenues dans l'annuaire
- Un modèle de nommage : comment l'information est organisée et référencée
- Un modèle fonctionnel : une syntaxe des requêtes permettant l'interrogation de la base et la mise à jour des informations
- Un modèle de duplication : comment la base est répartie sur différents serveurs (tolérance aux pannes, répartition de la charge du serveur, …)
- Un modèle de sécurité : comment contrôler l'accès aux données ainsi que leur transfert

172

Le protocole LDAP

- Il définit
  - les échanges de la connexion Client/Serveur
    - commandes de connexion au service : `bind`, `unbind`, `abandon` (le client abandonne la requête en cours)
    - commandes de mises à jour des entrées de l'annuaire : `add`, `delete`, `modifiy`, `rename`
    - commandes d'interrogation : recherche (`search`) et comparaison (`compare`) d'entrées
  - le format de transport des données
    - pas de l'ASCII comme SMTP, HTTP, …
    - encodage LBER : *Lightweight Basic Encoding Rules*

173

Le protocole LDAP

- Il définit
  - les échanges de la connexion Serveur/Serveur
    - la réplication (*replication service*), en cours de normalisation (*LDUP : LDAP Duplication Protocol*)
    - créer des liens entre différents annuaires (*referral service*) - défini dans LDAPv3
  - les mécanismes de sécurité
    - méthodes d'authentification pour se connecter à l'annuaire (qui peut se connecter à l'annuaire et comment)
    - mécanismes de règles d'accès aux données (une fois connecté, à quoi peut-on accéder et avec quels droits)
    - mécanismes de chiffrement des transactions

174

Le protocole LDAP

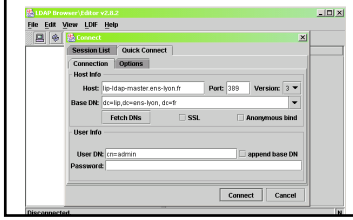
- LDAPv3 est conçu pour être extensible sans avoir à modifier la norme
  - permet l'ajout d'opérations (en plus des 9 de base)
  - permet l'ajout de paramètres associés à une opération
  - les mécanismes de sécurité sont définis dans une couche séparé : permet des méthodes d'authentification externes

```
ogluck@lima:/etc/ldap$ cat /etc/services | grep ldap
ldap 389/tcp  # Lightweight Directory Access Protocol
ldap 389/udp  # Lightweight Directory Access Protocol
ldaps 636/tcp # LDAP over SSL
ldaps 636/udp # LDAP over SSL
```


175

Se connecter à une base LDAP

Deux principaux éditeurs graphiques : GQ sous Unix (`http://biot.com/gq/`) et LDAP Browser\Editor sous Windows (`http://www.iit.edu/~gawojar/ldap/`)



176

Le modèle d'information

- Un annuaire est constitué de schémas LDAP qui vont déterminer les objets utilisables dans l'annuaire
- Un schéma LDAP
  - définit une liste des classes d'objets, les types des attributs et leur syntaxe répondant aux normes de *l'Object Management Group* (OMG)
  - standardisé (IANA) : pour l'interopérabilité entre logiciels
  - permet l'interfaçage avec les applications (Samba, …)

```
ogluck@lima:/etc/ldap$ ls /etc/ldap/schema/
README core.schema inetorgperson.schema krb5-kdc.schema
  nis.schema corba.schema cosine.schema  java.schema
  misc.schema openldap.schema
```

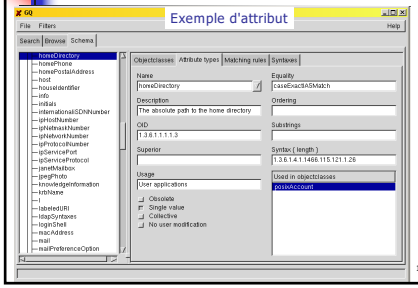

177

Le modèle d'information

- Un attribut est défini par
  - un nom, un identifiant unique (OID), mono/multi-valué, une syntaxe et des règles de comparaison (*matching rules*), une valeur (format+taille limite), modifiable ou non
- Les classes d'objets modélisent
  - des objets réels : un compte Unix (`posixAccount`), une organisation (`o`), un département (`ou`), un personnel (`organizationPerson`), une imprimante (`device`), …
  - ou abstraits : l'objet père de tous les autres (`top`), …
- Une classe d'objet est définie par
  - un nom, un OID, des attributs obligatoires, des attributs optionnels, un type (structurel, auxiliaire ou abstrait)

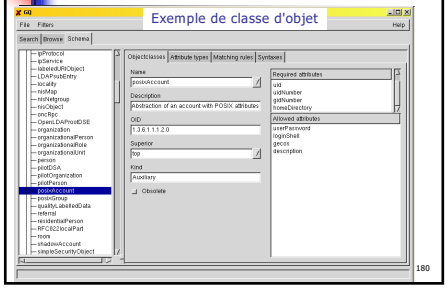
178

Le modèle d'information



179

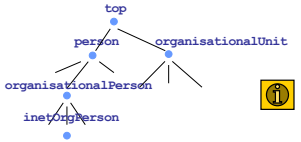
Le modèle d'information



180

Le modèle d'information

- Les classes d'objets forment une structure arborescente : tout en haut, l'objet `top`



- Chaque objet hérite des attributs de l'objet dont il est le fils
- Plus d'infos :
  http://www.it.ufl.edu/projects/directory/ldap-schema/
  http://ldap.akbkhome.com/

181

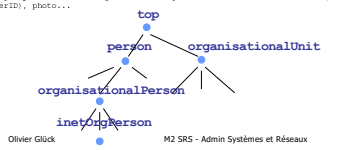
Le modèle d'information

```
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
```

L'objet person a comme attributs : commonName, surname, description, seeAlso, telephoneNumber, userPassword

L'objet fils organizationalPerson ajoute des attributs comme : organizationUnitName, title, postalAddress...

L'objet petit-fils inetOrgPerson lui rajoute des attributs comme : mail, labeledURI, uid (userID), photo...



182

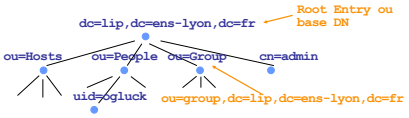
Le modèle de nommage

- Il définit comment sont organisées les entrées (=objets) de l'annuaire et comment elles sont référencées
- Structure arborescente contenant deux catégories d'objets
  - les conteneurs (une zone de rangement) : départ d'une nouvelle branche
    - peuvent contenir des conteneurs ou des feuilles
    - généralement, une sous-organisation de l'organisation (département, zone géographique, …)
  - les feuilles (véritables données) : terminaison des branches (généralement les machines, les utilisateurs, …)

183

Le modèle de nommage

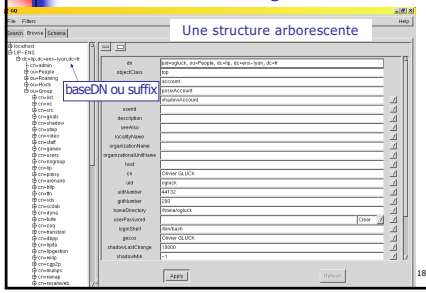
- Structure logique hiérarchique : le DIT (*Directory Information Tree*)
- Une entrée est identifiée par un nom unique : le DN (*Distinguish Name*)
- RDN - *Relative Distinguish Name*



184

Le modèle de nommage

Une structure arborescente



185

Le format LDIF

- *LDAP Data Interchange Format* (LDIF)
- Standard de représentation des entrées sous format texte --> permet de
  - faire des imports/exports de la base ou d'une partie
  - créer, ajouter, modifier,... un grand nombre d'entrées de façon automatisée

```
dn: uid=ogluck,ou=People,dc=lip,dc=ens-lyon,dc=fr
objectClass: top
objectClass: account
objectClass: posixAccount
objectClass: shadowAccount
uid: ogluck
uidNumber: 44132
gidNumber: 200
homeDirectory: /home/toto
cn: Olivier GLUCK
loginShell: /bin/bash
```


186

Le modèle fonctionnel

- Il décrit le moyen d'accéder aux données (syntaxe des requêtes) et les opérations que l'on peut leur appliquer
- Rappel des opérations de consultation/mise à jour
  - opérations de mise à jour des entrées de l'annuaire : `add`, `delete`, `modify`, `rename`
  - opérations d'interrogation : recherche (`search`) et comparaison (`compare`) d'entrées
  - --> il n'y a pas d'opération de lecture d'une entrée : pour connaître le contenu d'une entrée, il faut écrire une requête qui pointe sur cette entrée

187

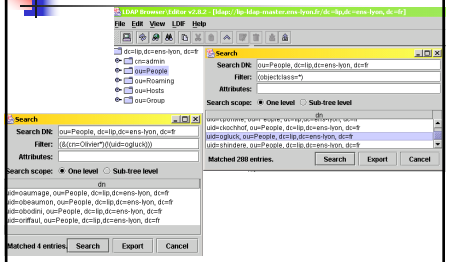
Le modèle fonctionnel

- Une requête est composée de 8 paramètres

`base object`	l'endroit de l'arbre où doit commencer la recherche
`scope`	la profondeur de la recherche
`derefAliases`	si on suit les liens ou pas
`size limit`	nombre de réponses limite
`time limit`	temps maxi alloué pour la recherche
`attrOnly`	renvoie ou pas la valeur des attributs en plus de leur type
`search filter`	le filtre de recherche
`list of attributes`	la liste des attributs que l'on souhaite connaître

188

Le modèle fonctionnel



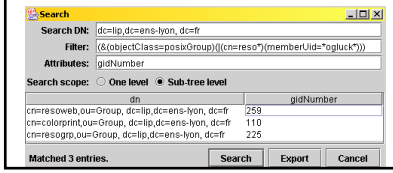
189

Le modèle fonctionnel

- Les filtres de recherche [RFC 2254]

(<operator>(<search operation>)(<search operation>)…)

```
(mail=*)  # existence
(uidNumber>=40000)  # comparaison
(|(ou=People)(ou=Group))  # OU
(&(cn=Olivier*)(!(uid=ogluck)))  # ET, contient, NON
(&(objectClass=posixGroup)(|(cn=reso*)(memberUid=*ogluck*)))
```



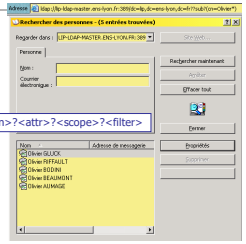
190

Les URLs LDAP [RFC 1959]

- Permet aux clients Internet d'avoir un accès direct aux annuaires LDAP
- Syntaxe :

ldap[s]://<host>:<port>/<base_dn>?<attr>?<scope>?<filter>

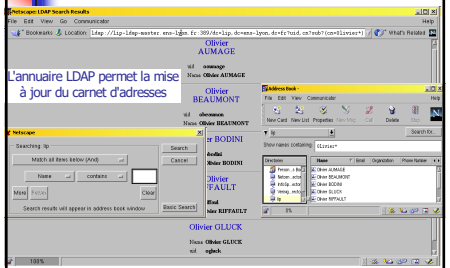
<base_dn> : point de départ de la recherche
<attr> : attributs consultés
<scope> : étendue de la recherche (base, one, sub)
<filter> : filtre de recherche (objectClass=*) par défaut



ldap://lip-ldap-master.ens-lyon.fr:389/dc=lip,dc=ens-lyon,dc=fr??sub?(cn=Olivier*)

191

Les URLs LDAP [RFC 1959]

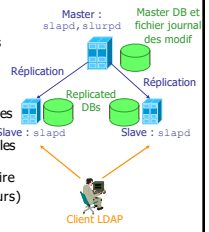


ldap://lip-ldap-master.ens-lyon.fr:389/dc=lip,dc=ens-lyon,dc=fr?uid,cn?sub?(cn=Olivier*)

192

Le modèle de duplication

- Il définit comment dupliquer l'annuaire sur plusieurs serveurs
  - améliorer le temps de réponse
  - être tolérant aux pannes
- Deux types de serveurs LDAP
  - *supplier server* (maître) : fournit les données
  - *consumer server* (esclave) : reçoit les données du maître
- Possibilité de partionner l'annuaire (éclatement sur plusieurs serveurs)
  - liens virtuels entre les différentes partitions (*referral service*)



193

Le modèle de sécurité

- Authentification pour se connecter au service
  - Anonymous authentification, Root DN/passwd authentification (administrateur), User DN/passwd
- Contrôle de l'accès aux données
  - droits d'accès aux données (fonctions de l'utilisateur authentifié) : lecture d'une valeur (`read`), modification (`write`), recherche (`search`), comparaison (`compare`), …
    - `search` : les données peuvent être une clé de recherche
    - `read` : permet de lire les données issues d'une recherche (par ex. `search` sur `cn` mais `read` seulement sur `Phone Number`)
  - règles définies sous forme d'ACLs (*Access Control List*) au niveau du sommet, d'un sous-arbre ou d'une entrée
- Chiffrement des transactions (LDAP+SSL, …)

194

Mettre en place un annuaire LDAP

- Il faut bien choisir les schémas
  - Quelles informations veut on stocker dans l'annuaire ?
    - --> choix des objets contenant les attributs désirés
  - Quelles sont les applications qui vont utiliser l'annuaire ?
    - Authentification des utilisateurs sous Unix, sous Windows (`samba`), gestion des groupes d'utilisateurs, listes de mail dynamiques (`sympa`), carnets d'adresses `Netscape`, … ?
- Il faut réfléchir à l'organisation du DIT
  - impacts sur la performance, les droits d'accès, …
- Puis dans un second temps
  - gestion centralisée sur un seul serveur ?
  - nombre de serveurs redondants ? Emplacement ?

195

OpenLDAP

- Logiciel LDAP du domaine public
- Le démon `slapd`
  - traite les requêtes LDAP
- Le démon `slurpd`
  - permet la réplication
- Des librairies LDAP
  - par exemple, pour authentifier les login via LDAP

  `libpam-ldap, libnss-ldap`
- Des utilitaires
  - `ldapadd  ldapdelete  ldapmodify`
    `ldapmodrdn  ldappasswd  ldapsearch`

196

Le fichier `/etc/ldap/slapd.conf`

- Permet de configurer le démon `slapd`
  - définition des schémas utilisés

    `include /etc/ldap/schema/inetorgperson.schema`
  - définition du *backend* (moteur de base de données utilisé pour ranger les données)

    `database ldbm` (`ldbm` par défaut, sinon `sql`, ...)
  - définition de la base de l'annuaire et de l'administrateur
    - le suffixe : racine de l'arbre

      `suffix "dc=lip,dc=ens-lyon,dc=fr"`
    - l'administrateur et son mot de passe

      `rootdn cn=admin,dc=lip,dc=ens-lyon,dc=fr`
      `rootpw toto`
    - le répertoire où la base est stockée

      `directory "/var/lib/ldap"`

197

Le fichier `/etc/ldap/slapd.conf`

- définitions des ACLs (`man slapd.access`)

```
# Format d'un ACL :
access to <what> [ by <who> <access> [ <control> ] ]+
<what>: *, un dn, un filtre LDAP, une liste d'attributs (attrs=...)
<who> : *, dn, anonymous, users (quelqu'un authentifié), self (le proprio), ...
<access> : none, auth, compare, search, read, write, ...
<control> : stop, continue, break (imbrication des règles...)
# Par défaut :
access to attrs=userPassword
        by dn="" write # l'admin
        by anonymous auth # droit de lecture uniquement lors du bind
        by self write # le propriétaire
        by * none
# The admin dn has full write access
access to *
        by dn="" write
        by * read # nécessaire d'avoir read pour le bind
```


198

Le fichier `/etc/ldap/slapd.conf`

- définition des réplicats
  - sur le serveur maître

    # fichier dans lequel `slapd` stocke les modifications pour `slurpd`
    `replogfile /var/lib/ldap/replog`
    # définition d'un réplicat
    `replica host=ldap.ens-lyon.fr:389 bindmethod=... ...`
  - sur un esclave

    # le dn autorisé à faire la mise à jour
    `updatedn "souvent slurpd"`
    # URL du maître
    `updateref ldap://master-ldap.ens-lyon.fr:389`
- ... `man slapd.conf`

199

Le fichier `/etc/ldap/ldap.conf`

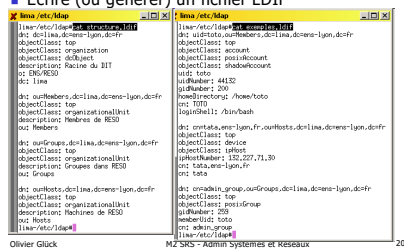
- Permet de donner des informations aux clients LDAP
  - `man ldap.conf`
  - peut aussi être fait dans `~/.ldaprc`
  - ou par des variables d'environnements

```
# base par défaut à contacter pour les opérations LDAP
BASE dc=lip,dc=ens-lyon,dc=fr
# en tant que qui le client se connecte à la base
BINDDN uid=ogluck,ou=People,dc=lip,dc=ens-lyon,dc=fr
# le serveur auquel se connecter
HOST ldap.ens-lyon.fr:389
# d'autres options de configuration...
```


200

Ajouter des entrées dans l'annuaire

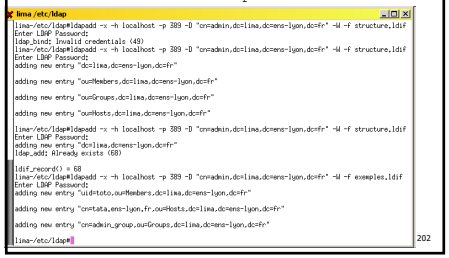
- Ecrire (ou générer) un fichier LDIF



201

Ajouter des entrées dans l'annuaire

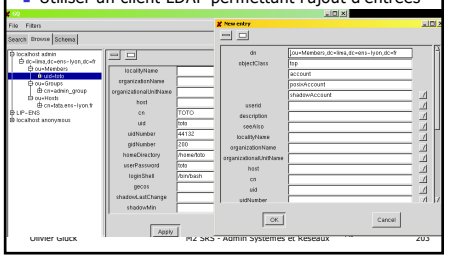
- Utiliser la commande `ldapadd`



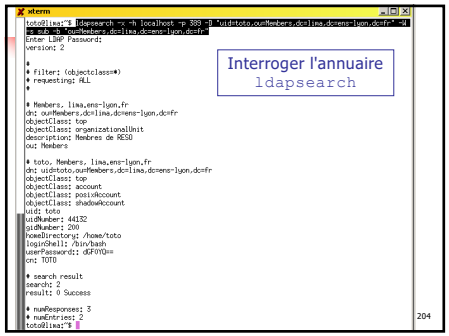
202

Ajouter des entrées dans l'annuaire

- Utiliser un client LDAP permettant l'ajout d'entrées

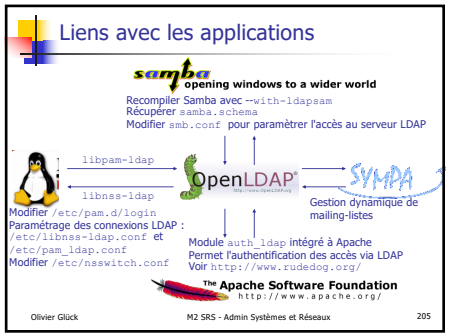


203



204

Liens avec les applications



205

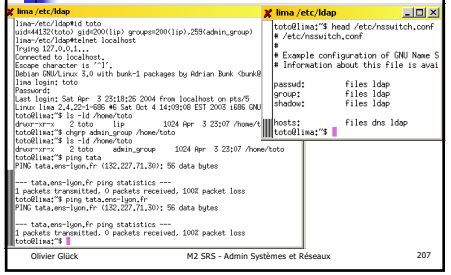
Authentification Unix via LDAP

- PAM - *Pluggable Authentification Modules*
  - permet de gérer la politique d'authentification des connexions sans recompiler quoi que ce soit
  - pour authentification via LDAP, rajouter la ligne `auth sufficient pam_ldap.so` dans le fichier `/etc/pam.d/login` qui signifie l'authentification via LDAP est suffisante
  - voir aussi `/etc/pam.d/ssh`, `/etc/pam.d/rsh` …
- Configurer l'accès à la base LDAP dans
  - `/etc/libnss-ldap.conf` et `/etc/pam_ldap.conf`
  - voir les pages `man` associées
- Indiquer dans `/etc/nsswitch.conf` l'ordre d'interrogation pour l'authentification
  - toujours laisser `files` en premier !

Olivier Glück M2 SRS - Admin Systèmes et Réseaux 206

206

Authentification Unix via LDAP



207

Authentification Unix via LDAP

```
toto@lima:~$ head -4 /etc/pam.d/passwd
#
# The PAM configuration file for the Shadow `passwd' service
#
password        sufficient pam_ldap.so
toto@lima:~$ ldapsearch -x -h localhost -p 389 -D "uid=toto,ou=Members,dc=lima,dc=ens-lyon,dc=fr" -W -LLL
 -s sub -b "uid=toto,ou=Members,dc=lima,dc=ens-lyon,dc=fr" "(cn=TOTO*)" userPassword
Enter LDAP Password:
dn: uid=toto,ou=Members,dc=lima,dc=ens-lyon,dc=fr
userPassword:: dG90bw==

toto@lima:~$ passwd
Enter login(LDAP) password:
New password:
Re-enter new password:
LDAP password information changed for toto
passwd: password updated successfully
toto@lima:~$ ldapsearch -x -h localhost -p 389 -D "uid=toto,ou=Members,dc=lima,dc=ens-lyon,dc=fr" -W -LLL
 -s sub -b "uid=toto,ou=Members,dc=lima,dc=ens-lyon,dc=fr" "(cn=TOTO*)" userPassword
Enter LDAP Password:
dn: uid=toto,ou=Members,dc=lima,dc=ens-lyon,dc=fr
userPassword:: e1NNRDVST1kFZcFRTWDIzVlpuOHVlNFBoNzZubHBlaG9RPQ==

toto@lima:~$
```


208

Authentification Samba via LDAP

- Dans `/etc/samba/smb.conf`

```
[global]
# paramétrage des connexions LDAP
   ldap server = localhost
   ldap port = 389
   ldap suffix = "dc=lima,dc=ens-lyon,dc=fr"
   ldap admin dn = "cn=admin,dc=lima,dc=ens-lyon,dc=fr"
   ldap ssl = no
```

- Après avoir créé une entrée `sambaAccount` dans l'annuaire pour `user_login`, il suffit de faire `smbpasswd user_login` pour que Samba mettent à jour les champs Samba dans l'annuaire

209

Partie 2 : Applications de l'Internet de type Client/Serveur (suite3)

Olivier GLÜCK
Université LYON 1/UFR d'Informatique
Olivier.Gluck@ens-lyon.fr
http://www710.univ-lyon1.fr/~ogluck

210

Plan de la partie 2

- Introduction / Rappel
- Connexions à distance (telnet/rlogin/rsh/ssh/X11)
- Applications de transfert de fichiers (FTP/TFTP)
- Accès aux fichiers distants (NFS/SMB)
- Gestion d'utilisateurs distants (NIS)
- DNS : un annuaire distribué
- LDAP : un annuaire fédérateur sécurisé
- **La messagerie électronique** (SMTP/POP/IMAP)
- Le protocole HTTP

211

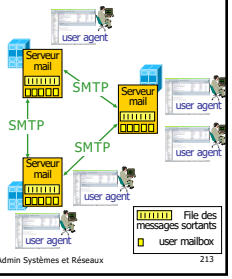
La messagerie électronique

Les différents composants
Configuration d'un agent utilisateur
Le protocole SMTP
Codage des messages et types MIME
Les protocoles d'accès

212

Courrier électronique : les composants

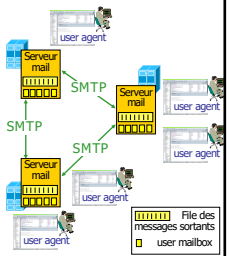
- 4 composants principaux :
  - des agents utilisateurs
  - des serveurs de mail
  - un protocole de transfert de mail : *Simple Mail Transfer Protocol* (SMTP)
  - un protocole d'accès à la boîte aux lettres (POP, IMAP, ...)
- Les agents utilisateurs :
  - composition, édition, lecture du courrier électronique
  - ex : Eudora, Outlook, elm, pine, Netscape Messenger
  - un agent utilisateur dialogue avec un serveur pour émettre/recevoir des messages



213

Courrier électronique : les composants

- Les messages entrants et sortants sont stockés sur le serveur
- La boîte aux lettres de chaque utilisateur contient les messages entrants (à lire)
- File d'attente de messages mail sortants (à envoyer)
- Protocole SMTP entre les serveurs de mail pour l'envoi des messages
  - modèle C/S : Client (serveur de mail émetteur) - Serveur (serveur de mail récepteur)
  - le client se connecte sur le port 25/TCP du serveur pour transférer son message



214

Courrier électronique : les composants

Nous ne pouvons pas afficher l'image.

- Les protocoles d'accès : consultation de sa boîte aux lettres (après authentification)
  - POP3 : *Post Office Protocol v3* [RFC 1939]
    - autorisation (agent <--> server) et téléchargement
  - IMAP4 : *Internet Message Access Protocol* v4 [RFC 3501]
    - plus de caractéristiques, plus complexe, plus récent
    - manipulation de messages stockés sur le serveur
  - HTTP (*Webmail*) : Hotmail , Yahoo! Mail, …

215

Courrier électronique : les composants

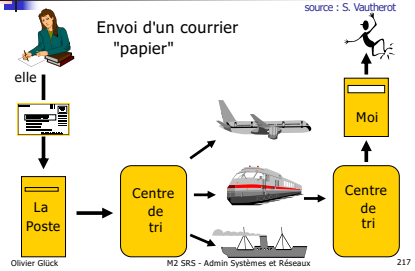
Nous ne pouvons pas afficher l'image.

- Dans les débuts du courrier électronique
  - il n'y avait pas de protocole d'accès
  - SMTP était juste prévu pour échanger du courrier entre le serveur de l'émetteur (client) et le serveur du récepteur (serveur) --> étapes 3-4-5 uniquement
- Pourquoi un protocole d'accès et une évolution de SMTP permettant au serveur sortant d'être à la fois client et serveur SMTP ?
- Pourquoi une file des messages sortants ?
- Pourquoi ne pas mettre le serveur sortant directement sur le poste utilisateur ?

216

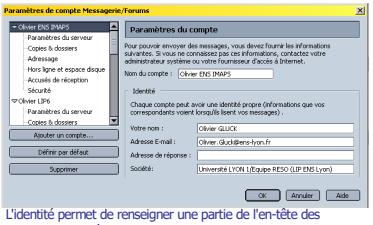
Analogie : le courrier "papier"

source : S. Vautherot



217

Configuration d'un agent utilisateur



L'identité permet de renseigner une partie de l'en-tête des messages envoyés

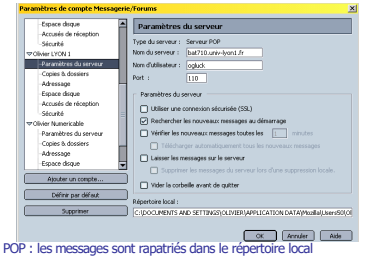
218

Configuration d'un agent utilisateur

Paramètrage du serveur sortant

219

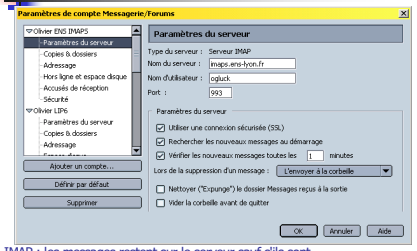
Configuration d'un agent utilisateur



POP : les messages sont rapatriés dans le répertoire local

220

Configuration d'un agent utilisateur



IMAP : les messages restent sur le serveur sauf s'ils sont supprimés, déplacés, ...

221

Le protocole SMTP [RFC 821]

- Transfert direct entre le serveur émetteur et le serveur récepteur (port 25/TCP)
- 3 phases de transfert
  - handshaking (établissement de la connexion)
  - transfert d'un ou plusieurs messages
  - fermeture de la connexion
- Les connexions sont **persistentes**
  - si plusieurs messages à destination du même serveur sont en attente sur le serveur émetteur, ils transiteront tous sur la même connexion TCP

222

Le protocole SMTP [RFC 821]

- Un message est composé d'un en-tête et d'un corps (RFC 822)
  - les champs de l'en-tête peuvent être positionnés soit par l'agent utilisateur émetteur, soit par le serveur entrant, soit par le serveur sortant
  - un champ d'en-tête est de la forme
    `nom_champ: valeur<CRLF>`
  - l'en-tête contient au minimum les champs `From` et `To`, très souvent le champ `Subject`
  - peut permettre de mettre en place des filtres…

plus d'infos : `http://www.cru.fr/messagerie/accents.html`

223

Le protocole SMTP [RFC 821]

- Une succession de Commande/Réponse
  - Commande SMTP : texte ASCII
  - Réponse SMTP : code d'état (status) + phrase
- Un message peut contenir plusieurs objets ; ils sont alors envoyés dans un message "multipart" (contrairement à HTTP : 1 objet = 1 réponse)
- Le serveur SMTP utilise `CRLF.CRLF` pour reconnaître la fin d'un message
- Les messages (en-tête ET corps) sont transférés en ASCII 7 bits (US-ASCII)

224

Le codage des messages

- SMTP est prévu pour transférer des caractères US-ASCII sur 7 bits --> problème de la représentation des caractères accentués, du transfert des octets (images…)
- Pour transférer une image ou du texte accentué, l'agent utilisateur émetteur/récepteur doit encoder/décoder le contenu du message
- Encodage `quoted-printable` :
  - généralement utilisé pour transférer du texte
  - permet le transfert des caractères ASCII étendus (codés sur 8 bits >128) comme les caractères accentués :
    - ils sont codés par les 3 caractères US-ASCII suivants : `=xx` où `xx` est le code hexadécimal du caractère à encoder
    - du coup, il faut coder le caractère `=` différemment : `=20`

225

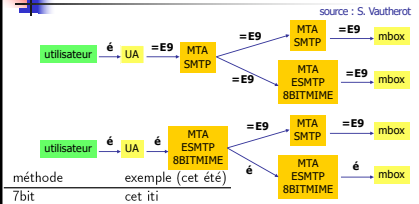
Le codage des messages

- Encodage `base64` :
  - généralement utilisé pour transférer des flux d'octets
  - permet le transfert des images ou autre série d'octets en tant que caractères ASCII NVT :
    - 3 octets (24 bits) sont transférés en tant que 4 caractères ASCII NVT : les 3 octets sont découpés en 4 fois 6 bits
    - bourrage avec le caractère = si pas aligné sur 4 caractères
    - permet de ne pas transférer plus de bits que le contenu initial (excepté le bourrage)
  - ESMTP [RFC 1425] : une évolution de SMTP qui permet le transfert des messages sans passer au format ASCII NVT
    - transfert de blocs de données sur 8 bits (flux d'octets)
    - spécifié par `Content-Transfer-Encoding: 8bit` ou `Binary` dans l'en-tête

226

Le codage des messages

source : S. Vautherot



méthode	exemple (cet été)
7bit	cet iti
8bit	cet été
quoted-printable	cet =E9t=E9
base64	Y2V0IOI06Qo=
binary	cet été

Le codage est fait par l'UA et/ou le MTA selon les possibilités (supposées) de l'agent suivant

227

Les types MIME [RFC 2045, 2056]

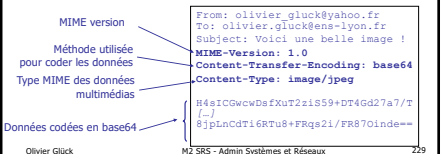
- MIME : *Multi-purpose Internet Mail Extensions*
- Permet l'échange de fichiers multimédias entre machines quelconques en spécifiant dans l'en-tête
  - le type du fichier en vue d'un traitement par l'agent utilisateur destinataire
  - le codage des données du fichier
- Les commandes MIME ont été intégrées dans HTTP1.0
- Un type MIME est composé
  - d'un type général (text, image, audio, video, application…)
  - et d'un sous-type (`image/gif`, `image/jpeg`, `application/pdf`, `application/rtf`, `application/msword`, `text/plain`, `text/html`)
- En perpétuelle évolution
- La machine cliente doit ensuite associer l'exécution d'une application à chaque type MIME

228

Les types MIME [RFC 2045, 2056]

`Content-Type: type/subtype; parameters`

- Lignes supplémentaires dans l'en-tête du message pour déclarer un type MIME et un encodage
- Content-type est généralement positionné à partir de l'extension du document demandé (`/etc/mime.types`)



229

Le type Multipart

```
From: olivier_gluck@yahoo.fr
To: olivier.gluck@ens-lyon.fr
Subject: Voici une belle image mais avec du texte !
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary=98766789

--98766789
Content-Transfer-Encoding: quoted-printable
Content-Type: text/plain

Cher Olivier,
Voici une photo de nos dernieres vacances !
--98766789
Content-Transfer-Encoding: base64
Content-Type: image/jpeg

H4sICGYRMTQAA3NsaWRlcy5wcwDsfXuT2ziS59+DT4Gd275a
56o7LlgSJbFNiWpSqsfw6rvLxPgSxIlVnk64i54ftRKl67/T
[…]
8jpLnCdTi6RTu8+FRqs2i/RTuy56plYbYVsa1fdvUjHrtV6g
RTf4/hy67fgIIVDfeR+rtYuNFR87oinde==
--98766789--
```

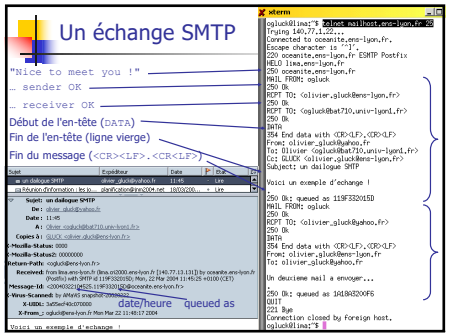

230

Les commandes SMTP

Commande	Description
`HELO nom_client`	identifie le client SMTP ; établit la connexion
`MAIL From: <@exp>`	identifie l'expéditeur du message
`RCPT To: <@dest>`	désigne le destinataire du message
`DATA`	indique le début du message (en-tête+corps)
`QUIT`	termine la connexion
`NOOP`	pas d'opération ; force le serveur à répondre
`RSET`	réinitialisation de la saisie de données (DATA)

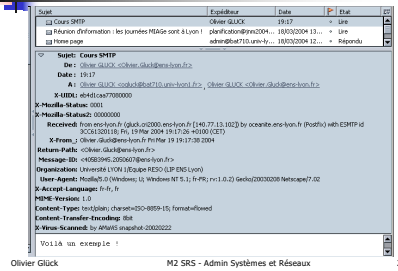
231

Un échange SMTP



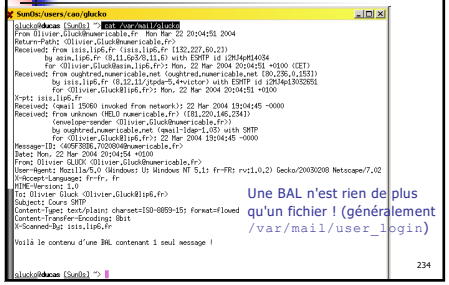
232

Exemple d'en-tête



233

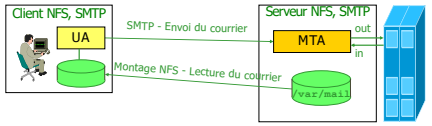
Exemple de contenu d'une bal



234

L'accès à sa boîte aux lettres

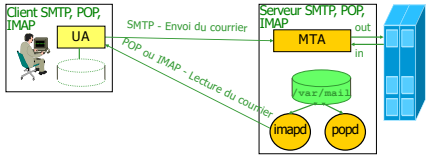
- Par lecture directe du fichier (en local ou par montage NFS)



235

L'accès à sa boîte aux lettres

- En utilisant un protocole spécifique (POP, IMAP) ou le protocole HTTP qui traverse généralement les pare-feus



236

Le protocole POP3 [RFC 1939]

Phase d'autorisation
- Commandes client :
  - **user**: déclare username
  - **pass**: password
- Deux réponses possible du serveur :
  - **+OK**
  - **-ERR**

Phase de transaction
- **list**: liste les numéros de messages et leur taille
- **retr**: rappatrie un message à partir de son numéro
- **dele**: efface un message
- **quit**

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

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


237

Le protocole POP3 [RFC 1939]

- POP3 est extrêmement simple
  - permet uniquement de télécharger des messages depuis le serveur en laissant éventuellement une copie de ceux-ci dans la BAL de l'utilisateur
  - pas adapté aux utilisateurs nomades
    - impossible de gérer des répertoires sur le serveur
    - impossible de gérer les messages en les laissant sur le serveur

--> IMAP répond à cette problématique au prix d'un protocole beaucoup plus complexe

238

Le protocole IMAP [RFC 3501]

- IMAP permet la gestion distante des messages
  - associe un message à un répertoire distant sur le serveur
  - permet à l'utilisateur de faire une recherche dans les messages sur le serveur
  - permet de ne consulter que des extraits de messages (par exemple que l'en-tête ou que la partie texte d'un message *multipart*…)
  - contrairement à POP3, IMAP conserve des informations d'état sur chaque utilisateur (noms des répertoires, listes des messages qu'ils contiennent…)

Plus d'infos : `http://www.imap.org/`

`http://cri.univ-lyon2.fr/doc/ImapMaisCEstTresSimple.html`

239

L'accès Webmail

- Pas de protocole d'accès spécifique
  - l'utilisateur utilise un navigateur Web comme agent utilisateur pour consulter/envoyer ses courriers
- Utilise le protocole HTTP (ou HTTPS) pour communiquer avec les serveurs SMTP/IMAP
  - le serveur HTTP exécute des scripts qui utilisent
    - le protocole IMAP pour communiquer avec le serveur IMAP et ainsi manipuler les messages distants de l'utilisateur
    - le protocole SMTP pour traduire une demande d'envoi d'un message de la part de l'utilisateur
- Avantages
  - adapté aux utilisateurs itinérants
  - pas besoin d'un agent utilisateur particulier, seule une connexion Internet avec Navigateur Web est nécessaire

Plus d'infos : `http://www.cru.fr/http-mail/critere.html`

240

Les alias

- Adresse d'un destinataire : `bal@nom_domaine`
- Problème :
  - `bal` n'est pas forcément le login de l'utilisateur
  - `nom_domaine` n'est pas forcément le nom du serveur de mail contenant les BAL
  - `bal` peut représenter plusieurs destinataires (listes)
- Il faut faire des alias (souvent `/etc/aliases`)

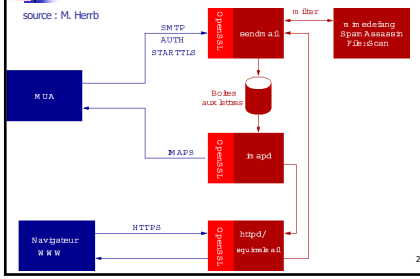
`Olivier.Gluck` --> `/var/mail/ogluck`

`ens-lyon.fr` --> `mailhost.ens-lyon.fr`

241

Une bonne archi de serveur MAIL

source : M. Herrb



243

HTTP : le protocole du Web

Intro : Web, URL et Formulaires

Format des requêtes/réponses

Durée de vie des connexions, Cookies

Différentes versions de HTTP, Proxy

Les requêtes clientes, les réponses du serveur

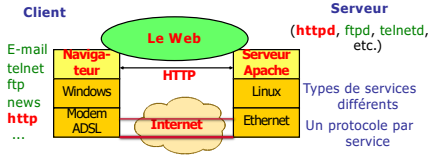
Les en-têtes, les types MIME

CGI, GET/POST, Format URL-encodé

244

Les services d'Internet

- Un service = une **application** qui utilise un **protocole** et un numéro de **port**
- Fonctionnement en mode **Client/Serveur** au dessus de TCP/IP



245

World Wide Web

- Architecture pour accéder à des documents liés entre eux et situés sur des machines reliées par Internet
- Architecture basée sur 3 concepts :
  - la localisation --> **URL**
  - le protocole --> **HTTP**
  - le langage --> **HTML**
- Popularité due à :
  - interfaces graphiques conviviales
  - très grande quantité d'informations
  - grande diversité des informations

246

Le jargon du Web

- Une page Web :
  - contient des "objets"
  - désignée par une adresse (URL)
- La plupart des pages Web contiennent :
  - du code HTML de base
  - des objets référencés
- L'URL a au moins deux composantes :
  - le nom d'hôte contenant la page Web
  - le chemin d'accès sur l'hôte
- L'Agent Utilisateur pour le Web est le *browser* :
  - MS Internet Explorer
  - Netscape Communicator
  - ...
- Le serveur Web :
  - Apache (domaine public)
  - MS Internet Information Server

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

247

Origines du Web

- Naissance au CERN : besoin d'échanges de documents, rapports, croquis, photos... entre des grosses équipes internationales pour des expériences demandant de longs investissements de mise en œuvre
  - mars 89 : Tim Berners-Lee : réseau de documents
  - septembre 90 : 1er prototype (mode texte)
  - décembre 91 : démonstration publique à la conférence Hypertext'91 de San Antonio

248

Envol du Web

- Février 93 : 1ère interface graphique *Mosaic* (Marc Andreesen)
- 1994 : M. Andreesen crée *Netscape Comm. Corp.* (développements logiciels pour le web)
- 1994 : création du W3C (**WWW Consortium**) par le CERN et le MIT (Tim Berners-Lee président) (développements du Web, standards...)
- 1996 : apparition des feuilles de styles (CSS)

249

Fonctionnement du Web

- Le client (navigateur ou *browser*) dialogue avec un serveur Web selon le protocole HTTP
- Le serveur vérifie la demande, les autorisations et transmet l'information
- Le navigateur interprète le fichier reçu et l'affiche (le navigateur, un *plug-in* ou un *helper*)
- A ce schéma de base, peuvent s'ajouter :
  - des **contrôles** par compte individuel, par domaine, par adresse IP...
  - des **exécutions** de code coté serveur et/ou coté client

250

Adressage des documents

- Il faut nommer, localiser et accéder à une page :
--> 3 questions : Quoi ? Où ? Comment ?
- Solution :
  - URL - *Uniform Resource Locator* : Adresse universelle de ressource
  - en 3 parties : le protocole (comment), le nom DNS (où) et le nom du document (quoi)
- URL --> URI (*Universal Resource Identifier*)
  - un sur-ensemble des URLs
- URL classique (simplifiée) :

`http://www.monsite.fr/projet/doc.html`

251

Adressage des documents

- Différentes composantes d'une URL :

`proto://host_name:port/path/extra_path?arguments`

  - la racine "`/`" de `path` est définie par la configuration du serveur Web
(**Attention :** à ne pas confondre avec la racine du système de fichiers sur le serveur)
  - `/path` peut contenir une étiquette (point d'ancrage)

`http://www.monsite.fr/projet/doc.html#label`

  - `extra_path` (après `.cgi` par ex.) et `arguments` permettent de passer des informations à des programmes s'exécutant sur le serveur

252

Adressage des documents

- URL relative :
  - un lien vers "`images/new.gif`" dans la page `http://www.monsite.fr/projet/doc.html`
est un lien vers
`http://www.monsite.fr/projet/images/new.gif`
  - le navigateur client reconstruit l'URL absolue pour faire la requête
  - la balise HTML `<BASE href="url">` permet de positionner la racine pour les URLs relatives du document contenant cette balise

253

Vision côté client

- Le Web est un ensemble de pages (documents) pouvant contenir des liens vers d'autres pages n'importe où dans le monde
- Consultation des pages via un navigateur
- L'utilisateur suit ces liens par simple click --> notion d'hypertexte (information répartie)
- Le navigateur (*browser*)
  - analyse l'URL demandée
  - demande au DNS l'adresse IP du site distant
  - établit une connexion TCP vers le numéro de port de l'URL (80 par défaut)
  - formule la requête au serveur

254

Vision côté client

- Le navigateur (*browser*)
  - va rechercher la page demandée
  - interprète les commandes de formatage et de mise en forme (police, gras, couleurs...)
  - va rechercher et affiche des images
  - animation (code JavaScript, gifs...)
  - affiche la page correctement formatée
- Paramétrage à plusieurs niveaux
  - valeurs par défaut du navigateur
  - valeurs fixées dans le document
  - préférences de l'utilisateur (navigateur)
  - exemples : couleur des liens (visités ou non), du texte, fond de la page, polices...

255

Vision côté serveur

- Le serveur est en permanence à l'écoute des requêtes formulées par les clients (qui peuvent être très nombreux !)
- Il vérifie la validité de la requête...
  - adresse correcte (URL)
  - client autorisé à accéder au document
- ... et y répond : envoi du texte, des images, du code à exécuter sur le client, d'un message d'erreur, d'une demande d'authentification, ...
- Il peut exécuter un programme localement qui va générer une réponse HTML (pages dynamiques)

256

Pourquoi des formulaires ?

- Apporte de l'inter-activité avec l'utilisateur en proposant des zones de dialogue : un formulaire n'est qu'une interface de saisie !
- Selon les choix de l'utilisateur, il faut y associer un traitement
  - sur le client avec JavaScript par exemple
  - sur le serveur par l'intermédiaire de CGI, PHP, …
- Exemples typiques d'utilisation de formulaire
  - commandes, devis via Internet
  - moteurs de recherche
  - interactions avec une base de données

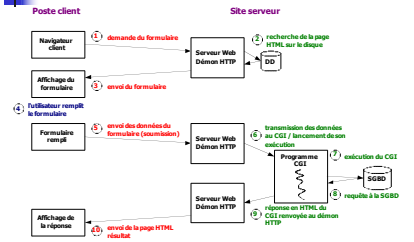
257

Principe du formulaire

- On décrit à l'aide de balises HTML les différents champs de saisie
- Chaque zone est identifiée par un nom symbolique auquel sera associée une valeur par l'utilisateur
- Quand le formulaire est soumis, les couples (nom/valeur) de toutes les zones sont transmis dans la requête HTTP au serveur
- A chaque zone de saisie peut être associé un traitement sur le client par l'intermédiaire d'un événement JavaScript

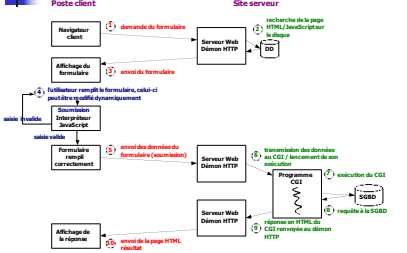
258

Le client est passif



259

Le client est actif



260

Caractéristiques de HTTP

- HTTP : Hyper Text Transfer Protocol
- Protocole régissant le dialogue entre des clients Web et un serveur (c'est le langage du Web !)
- Fonctionnement en mode Client/Serveur
- Une transaction HTTP contient
  - le type de la requête ou de la réponse (commande HTTP)
  - un en-tête
  - une ligne vide
  - un contenu (parfois vide)
- Très peu de type de requêtes/réponses
- Port standard : 80

261

Une transaction typique (1)

- 1 - le client contacte le serveur pour demander le document index.html

GET /~ogluck/index2.html HTTP/1.1

- 2 - le client envoie des informations d'en-tête pour informer le serveur de sa configuration et des documents qu'il accepte

User-Agent: Mozilla/4.0 (compatible;MSIE 6.0;Windows NT 5.1)
Host: www710.univ-lyon1.fr
Accept: image/gif, image/jpeg, */*

- 3 - le client envoie une ligne vide (fin de l'en-tête) et un contenu vide dans cet exemple

262

Une transaction typique (2)

- 4 - le serveur répond en commençant par indiquer par un code, l'état de la requête

HTTP/1.1 200 OK

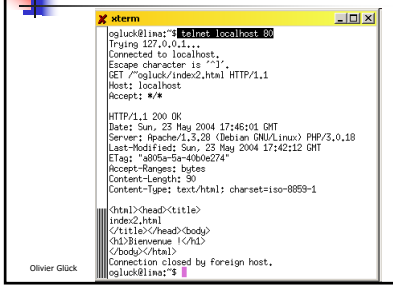
- 5 - le serveur envoie un en-tête qui donne des informations sur lui-même et le document demandé

Date: Sun, 23 May 2004 17:46:01 GMT
Server: Apache/1.3.28 (Debian GNU/Linux) PHP/3.0.18
Last-Modified: Sun, 23 May 2004 17:42:12 GMT
Content-Length: 90
Content-Type: text/html; charset=iso-8859-1

- 6 - puis une ligne vide (fin de l'en-tête) et le contenu du document si la requête a réussi

263

Une transaction typique (3)



264

Format des requêtes/réponses

- Format des requêtes
  - type de la requête (METHOD, URL, version HTTP)
  - en-tête
  - une ligne vide
  - un contenu éventuel
- Format des réponses
  - un code de réponse (version HTTP, code, description)
  - en-tête
  - une ligne vide
  - le contenu de la réponse

265

Durée de vie des connexions

- HTTP 1.0 (RFC 1945)
  - dès que le serveur a répondu à une requête, il ferme la connexion HTTP
- HTTP 1.1 (RFC 2068)
  - par défaut, la connexion est maintenue tant que le serveur ou le client ne décide pas de la fermer (`Connection: close`)
- HTTP est un protocole **sans état**
  - aucune information n'est conservée entre deux connexions
  - permet au serveur HTTP de servir plus de clients en un temps donné (gestion légère des transactions)
  - pour conserver des informations entre deux transactions, il faut utiliser un *cookie*, des champs cachés d'un formulaire, …

266

Cookies

- Moyen pour le serveur de stocker des informations chez le client pour palier au caractère sans état du protocole HTTP
- Cookie=une chaîne de caractères url-encodée de 4ko max stockée sur le disque dur du client
- Informations associées à un ensemble d'URL qui sont envoyées lors de toute requête vers l'une de ces URL
- Les *cookies* permettent de
  - propager un code d'accès (évite une authentification lors de chaque requête)
  - identification dans une base de données
  - fournir des éléments statistiques au serveur (compteurs de pages visitées, …)

267

Installation d'un Cookie sur le client

- Directive Set-Cookie dans l'en-tête de la réponse HTTP (envoyé lors de la première connexion)

Set-Cookie: nom=valeur; expires=date;
path=chemin_accès; domain=nom_domaine; secure

  - le couple nom/valeur est le contenu du cookie (seul champ obligatoire), sans espace ; et ,
  - le cookie devient invalide après la date indiquée
  - path=/pub signifie que le cookie est valable pour toutes les requêtes dont l'URL contient /pub
  - domain indique le nom de domaine (associé au serveur) pour lequel le cookie est valable
  - secure : le cookie n'est valable que lors d'une connexion sécurisée

268

Utilisation d'un Cookie par le client

- Chaque fois qu'un client va effectuer une requête, il vérifie dans sa liste de *cookies* s'il y en a un qui est associé à cette requête
- Si c'est le cas, le client utilise la directive Cookie dans l'en-tête de la requête HTTP

Cookie: nom1=valeur1; nom2=valeur2; …

- Le serveur peut insérer plusieurs directives Set-Cookie
- Dans la première spécification des *cookies* :
  - un client peut stocker un maximum de 300 *cookies*
  - un maximum de 20 *cookies* par domaine est permis
  - la taille d'un *cookie* est limitée à 4Ko

269

Différentes versions de HTTP (1)

- Version d'origine : HTTP 0.9
  - Une seule méthode : GET
  - Pas d'en-têtes
  - Une requête = une connexion TCP
- Amélioration en 2 étapes
  - HTTP 1.0 :
    - introduction des en-têtes (échange de "méta" info)
    - nouvelles possibilités : utilisation de caches, méthodes d'authentification, …
  - HTTP 1.1 :
    - mode **connexions persistantes** par défaut
    - introduction des serveurs virtuels -> la directive `Host` dans la requête est nécessaire

270

Différentes versions de HTTP (2)

- Intérêt des connexions persistantes
  - exemple d'une page d'accueil avec 5 images

  HTTP 0.9 : 6 connexions/déconnexions TCP/IP

  HTTP 1.1 : 1 seule connexion TCP/IP
- Intérêt d'un cache - amélioration des performances
  - les pages qui sont le plus souvent demandées sont conservées dans un cache
  - -> soulage le réseau
  - -> accès plus rapide
  - peut être utilisé localement ou par l'intermédiaire d'un serveur relais (*proxy*)

271

Connexions persistantes

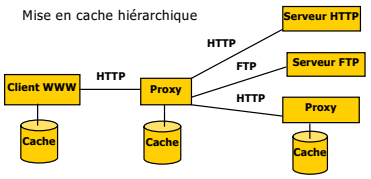
- Non-persistante
  - HTTP/1.0
  - le serveur interprète les requêtes, répond et ferme la connexion TCP
  - 2 RTTs sont nécessaires pour lire chaque objet
  - chaque transfert doit supporter le *slow-start*
  - exemple page contenant :
    - 1 fichier HTML
    - 10 images JPEG
- Persistante
  - par défaut dans HTTP/1.1
  - une seule connexion TCP est ouverte vers le serveur
  - le client envoie la requête de tous les objets requis dès qu'ils sont référencés dans le code HTML
  - moins de RTTs et moins de *slow-start*
  - deux versions : avec/sans pipeline

Mais la plupart des navigateurs de version 1.0 utilisent des connexions parallèles

272

Proxy

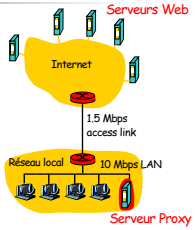
Mise en cache hiérarchique



273

Intérêt du cache Web

- Hypothèse : le cache est proche du client
- Réduction du temps de réponse
- Réduction du débit vers les serveurs distants



274

Les requêtes du client

- Rappel : Format d'une requête
  - une commande HTTP (METHOD), une URL qui identifie la ressource demandée, la version de HTTP
  - l'en-tête et une ligne vide
  - éventuellement un contenu (corps de la requête)
- Méthode GET
- Méthode POST
- Méthode HEAD
- D'autres méthodes qui ne sont pas souvent supportées par les serveurs

275

La méthode GET

- La méthode standard de requête d'un document
  - récupérer un fichier, une image, …
  - activer un script CGI en lui transmettant des données
- Le contenu de la requête est toujours vide
- Le serveur répond avec une ligne décrivant l'état de la requête, un en-tête et le contenu demandé
- Si la requête échoue, le contenu de la réponse décrit la raison de l'échec (fichier non présent, non autorisé, …)

276

La méthode GET et les CGI

- Comme le contenu d'une requête GET est vide, les données du formulaire sont transmises via l'URL après un ?
- Les champs sont séparés par un &

```
GET /cgi-bin/prog.cgi?email=toto@site.fr&pass=toto&s=login HTTP/1.1
```

- Ici, trois champs du formulaire sont transmis dans la requête
- Le mot de passe est transmis en clair !
- Permet de conserver dans un *bookmark* les données saisies dans le formulaire
- L'URL a une taille limitée (4Ko)

277

La méthode POST

- Elle permet de transmettre des données au serveur dans le corps de la requête
- Exemple

```
POST /cgi-bin/prog.cgi HTTP/1.1
User-Agent: Mozilla/4.0 (compatible;MSIE 6.0;Windows NT 5.1)
Host: localhost
Accept: */*
Content-type: application/x-www-form-urlencoded
Content-length: 36

email=toto@site.fr&pass=toto&s=login
```

- Le mot de passe est toujours transmis en clair !

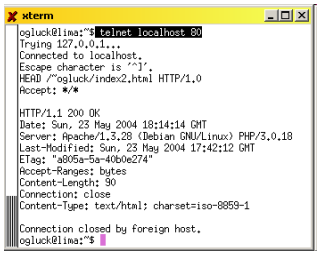
278

La méthode HEAD (1)

- Identique à GET mais permet uniquement de récupérer l'en-tête relatif à un document
- Permet de récupérer
  - la date de dernière modification du document (important pour les caches, JavaScript)
  - la taille du document (estimation du temps d'arrivée du document)
  - le type du document (le client peut sélectionner le type de documents qu'il accepte)
  - le type du serveur (permet de faire des requêtes spécifiques selon le type du serveur)
- Remarque : le serveur ne fournit pas nécessairement toutes ces informations !

279

La méthode HEAD (2)



280

Autres requêtes clientes

- PUT : permet de stocker le corps de la requête sur le serveur à l'URL spécifiée
- DELETE : suppression du document spécifié par l'URL
- OPTIONS : renvoie la liste des méthodes autorisées par le serveur
- TRACE : la corps de la requête entrante est renvoyée au client - utilisé pour faire du débug)
- ...

281

Les réponses du serveur

- Les codes de réponse
  - trois parties : version HTTP, code de statut, description textuelle du code

  HTTP/1.1 200 OK

  HTTP/1.1 404 Not Found
  - code=entier sur 3 chiffres classé selon des catégories
    - 100-199 : message d'information
    - 200-299 : succès de la requête cliente
    - 300-399 : la requête n'est pas directement serviable, le client doit préciser certaines choses
    - 400-499 : échec de la requête qui incombe au client
    - 500-599 : échec de la requête qui incombe au serveur (par ex. erreur d'exécution d'un CGI)

282

Quelques en-têtes de requêtes

- Identification du client

  From (adresse mail du client), Host (serveur, obligatoire en HTTP1.1), Referer (URL d'où l'on vient), User-Agent
- Préférences du client

  Accept (liste des types MIME acceptés), Accept-Encoding (compress|gzip|…), Accept-Langage, Accept-Charset
- Information pour le serveur

  Autorization (username:passwd encodé en base64), Cookie
- Conditions sur la réponse

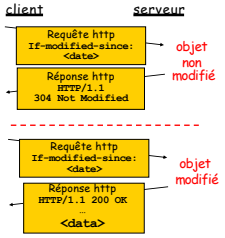
  If-Modified-Since (utile pour les caches), If-Unmodified-Since, If-Match (Etag)

283

Quelques en-têtes de requêtes

- Objectif : ne pas envoyer un objet que le client a déjà dans son cache
- Problème : les objets contenus dans le cache peuvent être obsolètes
- Le client spécifie la date de la copie cachée dans la requête http

  If-modified-since: <date>
- la réponse du serveur est vide si la copie cachée est à jour



284

Quelques en-têtes de réponses

- Informations sur le contenu du document

  Content-Type (type MIME du document), Content-Length (barre de progression du chargement), Content-Encoding, Content-Location, Content-Langage
- Informations sur le document

  Last-Modified (date de dernière modification), Allow (méthodes autorisées pour ce document), Expires (date d'expiration du document)
- En-tête générales

  Date (date de la requête), Server (type du serveur)

285

Transfert par morceaux en HTTP/1.1

- La réponse peut être envoyée en plusieurs morceaux (dans le cas des CGI par exemple car le serveur ne peut pas toujours déterminer la longueur totale de la réponse)

  Transfer-Encoding: Chunked
- Chaque morceau est constitué d'une ligne comportant la taille du morceau en hexadécimal puis des données
- Après les morceaux, une ligne contenant 0 et éventuellement des en-têtes supplémentaires

286

Les types MIME

- MIME : *Multi-purpose Internet Mail Extensions*
- Permet l'échange de fichiers multimédias entre machines quelconques en spécifiant le type du fichier
- Les commandes MIME ont été intégrées dans HTTP1.0
- Un type MIME est composé
  - d'un type général (text, image, audio, video, application…)
  - et d'un sous-type (image/gif, image/jpeg, application/pdf, application/rtf, text/plain, text/html)
- En perpétuelle évolution
- La machine cliente doit ensuite associer l'exécution d'une application à chaque type MIME
- Le serveur positionne Content-type à partir de l'extension du document demandé (`/etc/mime.types`)

287

CGI - Common Gateway Interface

- Interface de base qui définit la communication entre le serveur HTTP et un programme d'application
- CGI spécifie comment des navigateurs clients peuvent communiquer avec des programmes qui s'exécutent sur le serveur Web et qui génèrent des pages HTML dynamiques créées à la volée à partir du résultat des exécutions

288

Qu'est ce qu'un programme CGI ?

- Un programme
  - qui s'exécute sur la machine hébergeant le serveur HTTP
  - en langage compilé (binaire) ou interprété (script)
  - qui permet de
    - récupérer les données du formulaire à l'aide d'un *parser* : pour chaque champ, un couple NAME/VALUE est transmis au serveur
    - effectuer des traitements sur le serveur
      - lecture/écriture dans une base de données
      - stockage d'une info (compteurs, identifiant de connexion, …)
      - recherche d'une info
      - pied de page automatique (ex: date de dernière modification)
    - générer un résultat qui est renvoyé au client
      - page HTML, image, document postcript, …

289

Avantages/inconvénients

- Puissant mais dangereux
  - permet d'exécuter tout et n'importe quoi par le démon HTTP du serveur
- Un CGI doit s'exécuter rapidement
  - risque de surcharge du serveur
  - utilisateurs impatients : pendant que le CGI s'exécute, le client attend la réponse sans savoir pourquoi elle n'arrive pas…
  - possibilité d'envoyer dès le début de l'exécution une page qui permet d'indiquer à l'utilisateur que le résultat va arriver

290

Un premier exemple (1)

```
#! /bin/sh
# Date.cgi
echo 'Content-type: text/html'
echo ' '
#Création du corps du document
echo '<HTML><HEAD><TITLE>'
echo 'Date.cgi'
echo '</TITLE></HEAD><BODY>'
echo '<H1>Date sur le serveur</H1>'
echo -n "On est le `date +%D`, il est "
echo "`date +%H`h `date +%M`m"
echo '</BODY></HTML>'
```

Source du programme CGI

```
ogluck@lima:~/public_html/cgi-bin$ ./Date.cgi
Content-type: text/html

<HTML><HEAD><TITLE>
Date.cgi
</TITLE></HEAD><BODY>
<H1>Date sur le serveur</H1>
On est le 11/07/03, il est 11h
30m
</BODY></HTML>
```

Exécution du CGI sur le serveur

291

Un premier exemple (2)

Exécution du CGI depuis le client

292

Un premier exemple (3)

- Ce programme CGI n'utilise aucune donnée en provenance du client
- Il récupère simplement la date sur le serveur et affiche sur sa **sortie standard** le code d'une page HTML minimale contenant la date et l'heure
- La ligne "`Content-type: text/html`" est une information destinée au serveur pour la construction de l'en-tête HTTP constituant la réponse renvoyée au client (ici, il s'agit d'indiquer que le type des données générées par le CGI est une suite de commandes HTML)

293

Méthodes GET/POST (1)

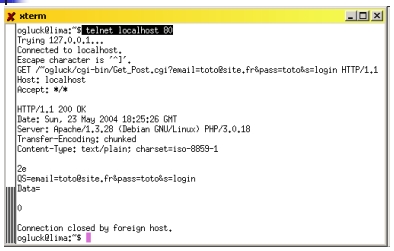
- Voici le code d'un petit script CGI en shell

```
#! /bin/sh
# Get_Post.cgi
echo 'Content-type: text/plain'
echo ' '
echo "QS=$QUERY_STRING"
read DATA
echo "Data=$DATA"
```

- Les résultats de l'exécution avec la méthode GET puis POST sont montrés dans les deux transparents suivants

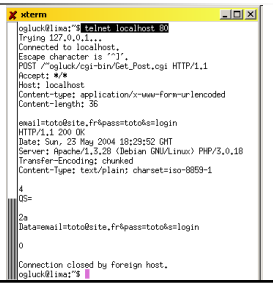
294

Méthodes GET/POST (2)



295

Méthodes GET/POST (3)



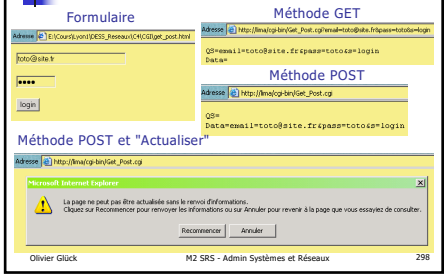
296

Méthodes GET/POST (4)

- Avec la méthode GET
  - les données relatives aux champs du formulaire sont transmises via l'URL (dans le type de la requête)
  - le programme CGI les récupère dans la variable d'environnement `QUERY_STRING`
  - il est possible de cliquer sur "Actualiser" pour retransmettre les données et de définir un *bookmark*
- Avec la méthode POST
  - les données relatives aux champs du formulaire sont transmises dans le corps de la requête HTTP
  - `Content-type` et `Content-length` sont positionnés
  - le programme CGI les récupère sur l'entrée standard
  - "Actualiser" et *bookmark* impossibles, données du formulaire non visibles dans les logs du serveur

297

Méthodes GET/POST (5)



298

Format URL encodé (1)

- Nécessité de coder les données de l'URL (méthode GET) sur le client pour construire la chaîne CGI pour respecter la RFC 2396 qui spécifie la syntaxe des URL
- Les caractères non-alphanumériques sont remplacés par %xx (xx=code ASCII du caractère en hexadécimal)
- Les caractères `; / ? : @ & = + $` et `,` sont réservés
  - `?` : début de QUERY_STRING
  - `&` : séparateur de champ
  - `=` : séparation entre le nom du champ et sa valeur
- Les espaces sont remplacés par des `+`

Format URL encodé (2)

- Format de la chaîne CGI

  nom_champ1=valeur1&nom_champ2=valeur2&…
- Cas des champs à valeurs multiples
  - exemple : listes à sélection multiples

  nom_liste=valeur1&nom_liste=valeur2&…
- La chaîne CGI
  - elle est construite par le client au format *URL-encoded* quand la requête est postée
  - elle est transmise au CGI tel quel via la variable d'environnement QUERY_STRING avec la méthode GET
  - elle est transmise au CGI tel quel via l'entrée standard avec la méthode POST