# *RESEAUX : Applications*



- **Comment accéder aux données ?**
- **Evolution des architectures**
- **Emulation de terminal, telnet**
- **Transfert de fichier, ftp, tftp**
- **Messagerie, smtp, pop3**
- **Web, http**

- **Performances**

# *Comment accéder aux données ?*

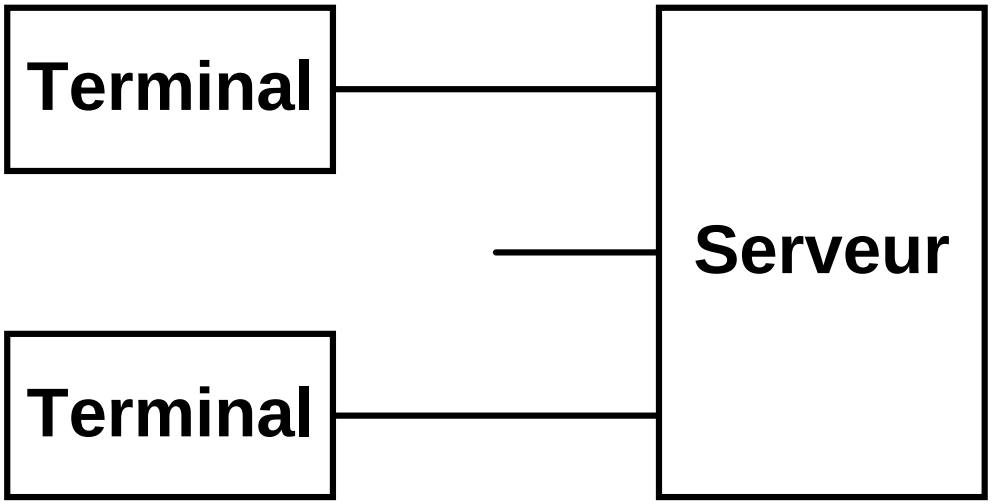


- *Time sharing*
- *File transfer*
- *Replication*
- *Synchronisation*
- *File access*
- *Data query*

# *Time sharing (temps partagé)*

- **Partage des ressources du serveur (CPU, mémoires, fichiers, …) entre plusieurs utilisateurs équipés de simples terminaux**

- **Terminal (clavier-écran) sans système de fichier**

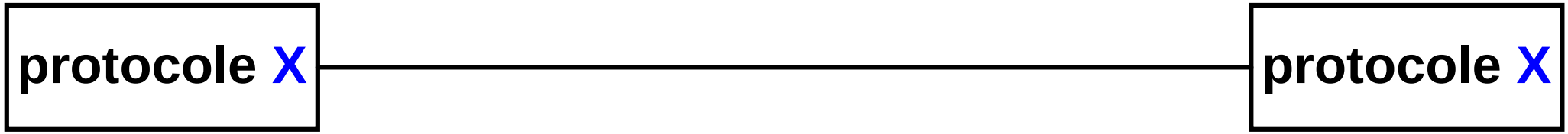



- **L'unicité des données est ainsi facile à garantir**

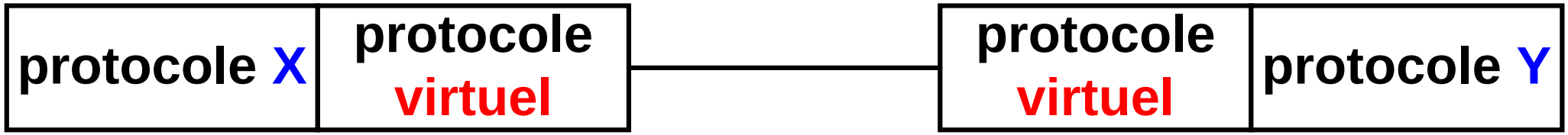
# *File transfer (transfert de fichier)*

- **Transfert des données entre systèmes de fichiers (ordinateurs) → copie(s) du fichier original**

- **Structure homogène et transfert via disquette**



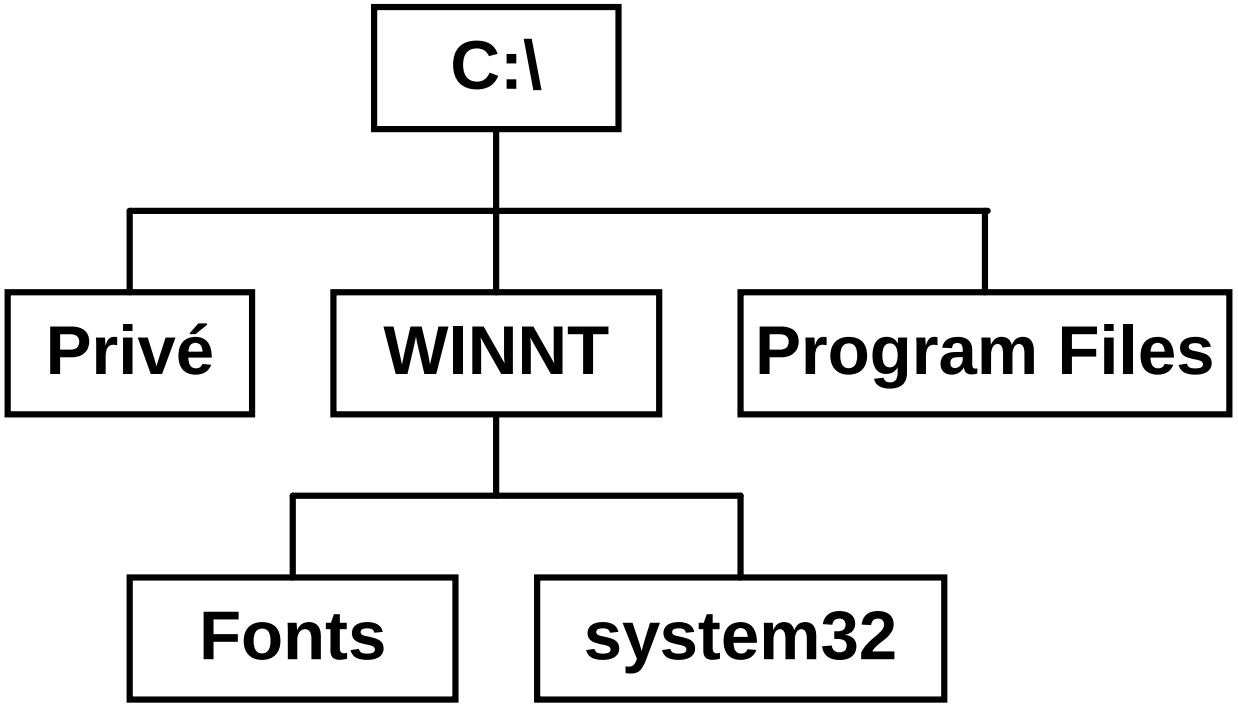

- **Structure hétérogène et protocole de transfert de fichier**



# *Fichiers et système de fichiers (1)*



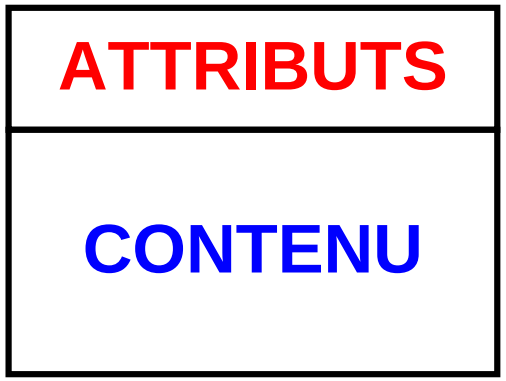
- **Le système de fichiers *(file system)* maintient une structure de répertoires (dossiers) pour stocker le nom et l'emplacement de chaque fichier sur le disque :**



# *Fichiers et système de fichiers (2)*



- **Chaque fichier est caractérisé par son contenu et ses attributs :**




**nom du ficher,**
**date, heure, ...**

- **Quelques systèmes de fichiers : VMS, FAT, NTFS**

- **Attibuts propres à chaque système de fichiers**

# *Réplication*

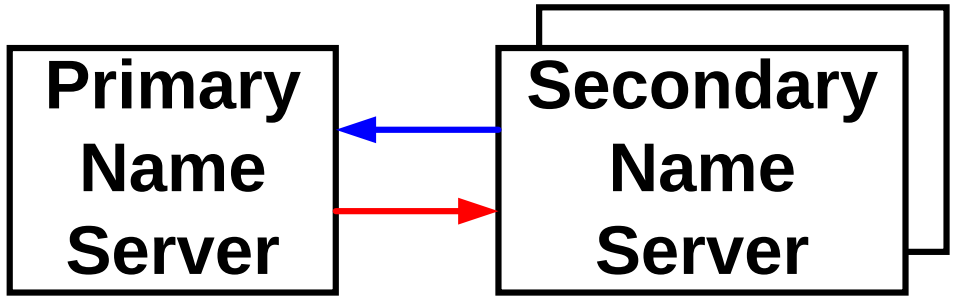


- **Illustration avec DNS *(Primary - Secondary)***

  **3600 (1h) refresh** — **intervalle entre 2 requêtes de transfert**

  **600 (10’) retry** — **intervalle si pas de réponse**

  **86400 (1j) expire** — ***sec. name server* répond alors que *prim. name server* est *down***

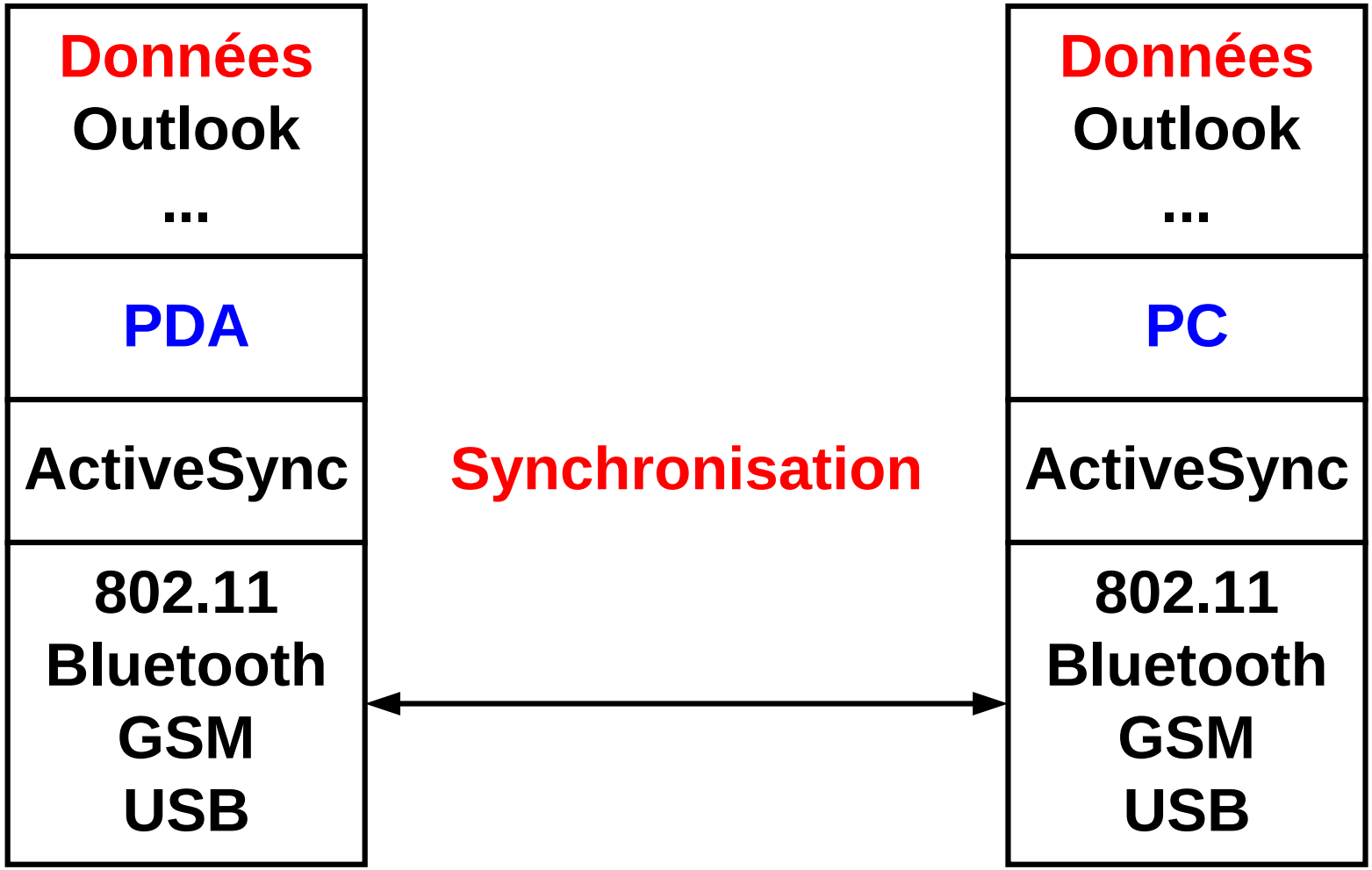



**zone transfers**

- **Atteindre une haute disponibilité**

# Synchronisation



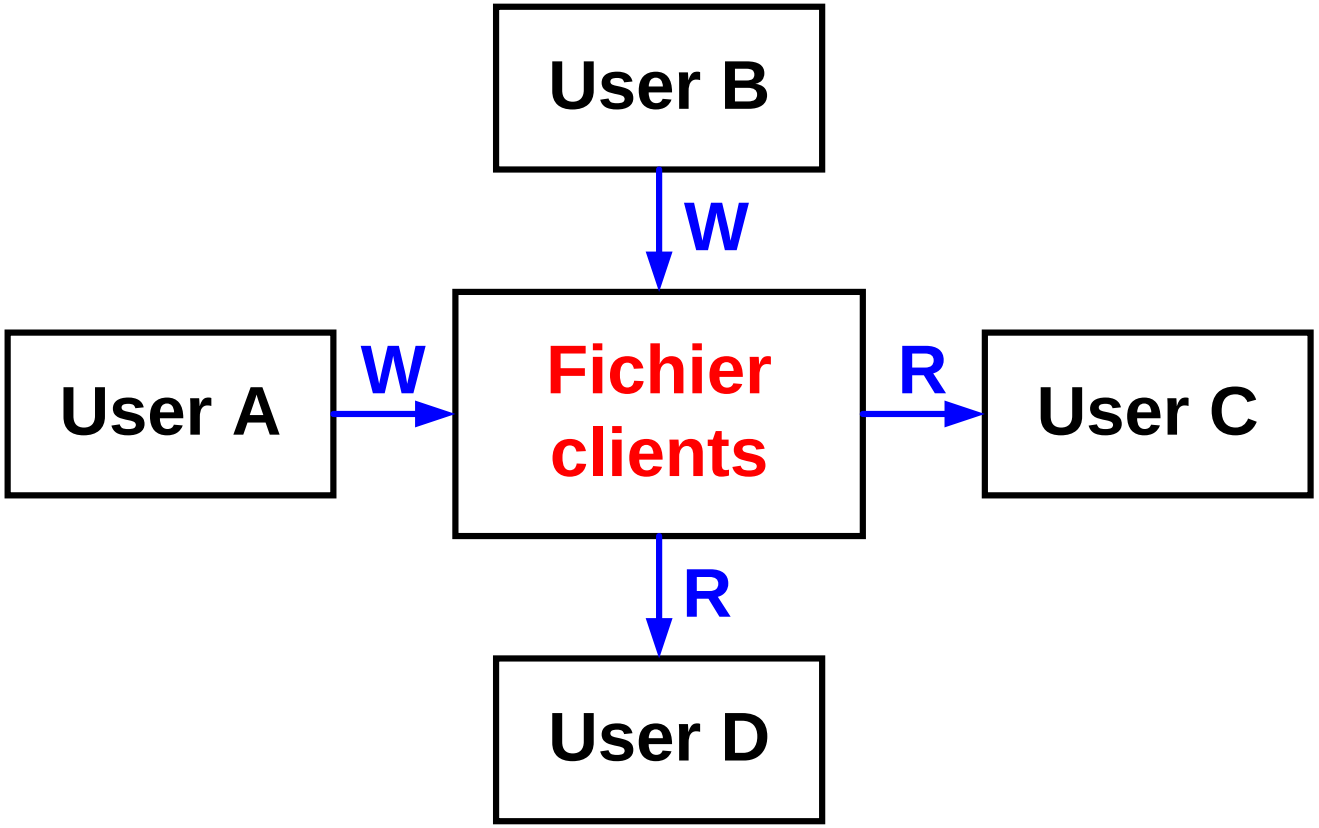



- Données synchronisées (data & heure du fichier)

# File access



- Plusieurs ordinateurs (processus) accèdent à un fichier unique




- Mécanisme de partage du fichier par bloc (écriture/lecture)

# *Data query*



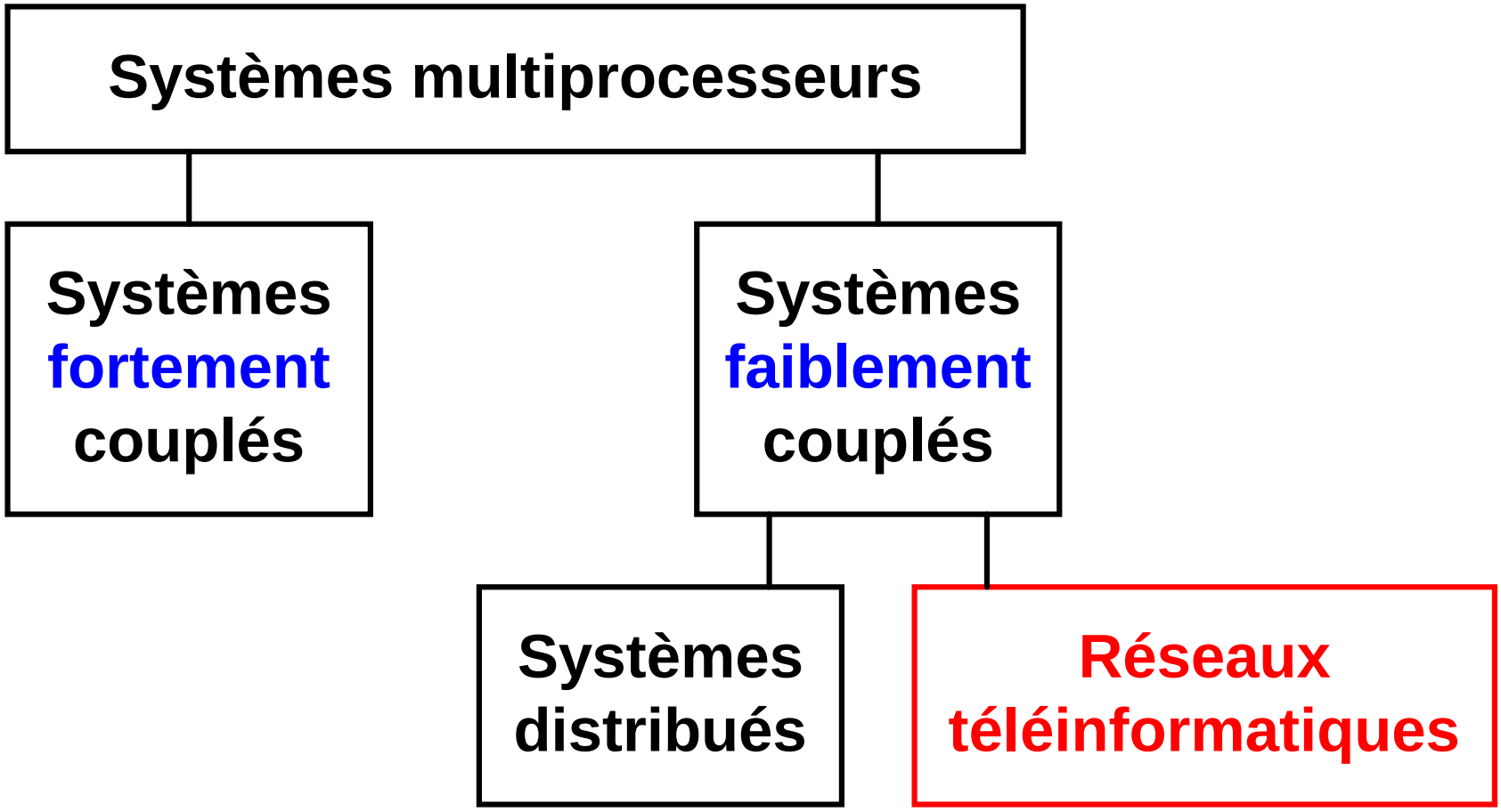
- **Demande structurée *(Structured Query Language)* de l'utilisateur (client)**

  **Rechercher tous les propriétaires du type de véhicule A qui l'utilisent plus de 20'000 km par an**

- **Réponse (éléments du fichier) du serveur**

# *Différents types de systèmes*





# *Evolution des architectures*

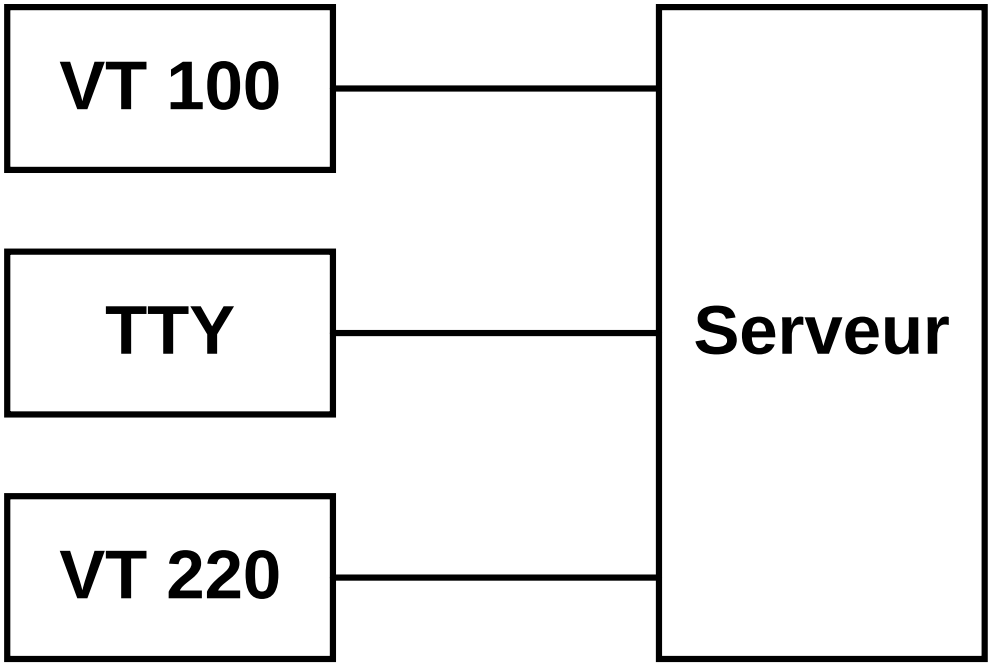


- ***Mainframe – minicomputer***
- ***Personal Computer***
- ***World-Wide Web***
- ***Thin client***
- ***Personal Digital Assistant***
- ***Client – server***

# *Mainframe – minicomputer (1)*



- **L’architecture des premiers ordinateurs était très centralisée; de simples terminaux alphanumériques donnaient accès aux ressources de l’ordinateur central**



# *Mainframe – minicomputer (2)*



- **Les normes étaient propriétaires comme terminal 3270 pour IBM ou VT100 pour Digital**
- **Certains systèmes utilisaient même des terminaux ASCII primitifs, proches du télex, appelés *teletype (TTY)***
- **Les applications dites mode caractère (CLI = *Command Line Interface*) avaient et ont encore leurs propres logiques d'interaction avec l'utilisateur**
- **On leurs reproche souvent leur manque de convivialité**

# *Personal Computer (PC)*



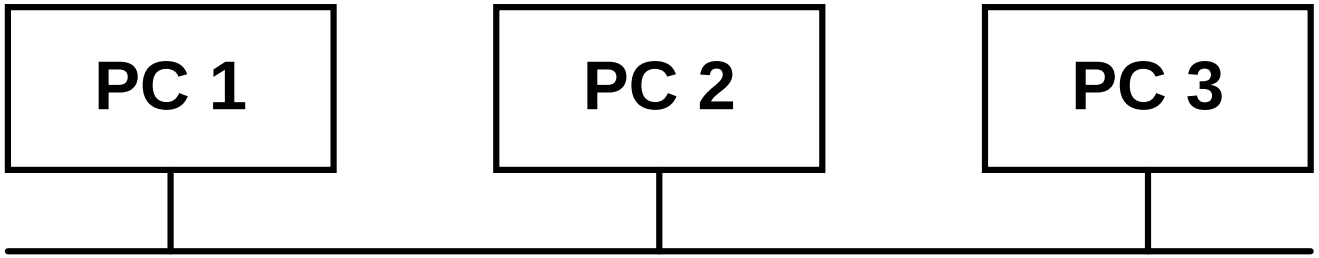
- **L'arrivée du PC puis du Macintosh marque le début d'une véritable révolution : chaque utilisateur dispose de ses proches ressources (CPU, mémoires, fichiers, …) qu'il doit gérer !**

- **Le système d'exploitation se gonfle à chaque version (DOS, Windows 3.1, …)**

- **L'interface graphique *(GUI : Graphical User Interface)* fait son apparition, les menus se standardisent et le "WYSIWYG" *(What You See Is What You Get)* devient argument de vente**

# *PCs en réseau (1)*






- **L'information doit pouvoir circuler dans l'entreprise**
- **Certaines ressources (imprimante laser, disque de grande capacité, unité de sauvegarde,…) doivent être partagées**
- **Comment garantir l'unicité des données ?**
- **De nouveaux produits comme Novell apparaissent**

# *PCs en réseau (2)*



- **Des extensions réseaux sont ajoutées au système d'exploitation : Windows 95, Windows NT, Windows 2000, Windows XP, …**
- **Les protocoles propriétaires (IPX de Novell, NetBEUI de Microsoft, …) sont remplacés par ceux de la famille TCP/IP**
- **L'interopérabilité avec le monde Unix est alors possible**

# *PCs en réseau (3)*



- **La gestion d'un réseau de PCs représente un travail important que de plus en plus d'entreprises souhaitent automatiser**

- **Le temps passé à installer divers logiciels (système d'exploitation, applicatifs, mises à jour, …) finit par représenter un coût annuel non négligeable**

- **Les règles de sécurité sont difficiles à faire respecter si chaque utilisateur dispose d'un accès à *internet* et possède le droit d'installer des logiciels**

# *World-Wide Web (1)*



- **Conçu au CERN (1989) pour le travail coopératif des physiciens, son impact dépasse vite le cadre universitaire**

- **Navigateur *(browser)***

  **Outil de navigation avec lequel l'utilisateur (client) accède à l'information désirée (URL)**

- ***Uniform Resource Locator (URL)***

  **Lien, pointeur sur ressource (document) disponible**

  **format = access-method://host[:port]/path/filename**

  **http://www.td.unige.ch/**

  **ftp://anonymous@ftp.switch.ch/mirror/**

  **telnet://info@nic.switch.ch**

  **file://s2.tdeig/labo/guide.txt**

# *World-Wide Web (2)*



- ***Server***

  **Ordinateur ou module logiciel capable de mettre à disposition un service : serveur de fichiers (ftp), serveur vidéo, serveur http, …**

- ***HyperText Transport Protocol* (HTTP)**

  **Protocole d'échange entre client et serveur**

- ***HyperText Markup Language* (HTML)**

  **Langage dans lequel les documents sont structurés**

# *World-Wide Web (3)*



- ***Page***

  **Un fichier ou document (HTML ou ASCII) capable d'être affiché par l'outil de navigation**

  **Page par défaut**

- ***Plug-in***

  **Extension de l'outil de navigation capable de supporter des applications (fichiers) comme PDF *(Portable Document Format)*, GIF *(Graphic Interchange Format)*, JPEG *(Joint Photographic Experts Group)*, MPEG *(Motion Picture Experts Group)*, VRML *(Virtual Reality Modeling Language)*, WAV *(WAVE)*, …**

# *World-Wide Web (4)*



- **Partage de l'information sous toutes ses formes (texte, son, image, …)**

- **Exécution de script du côté client (Javascript, …) et/ou du côté serveur (PHP, …)**

- ***Web Services***

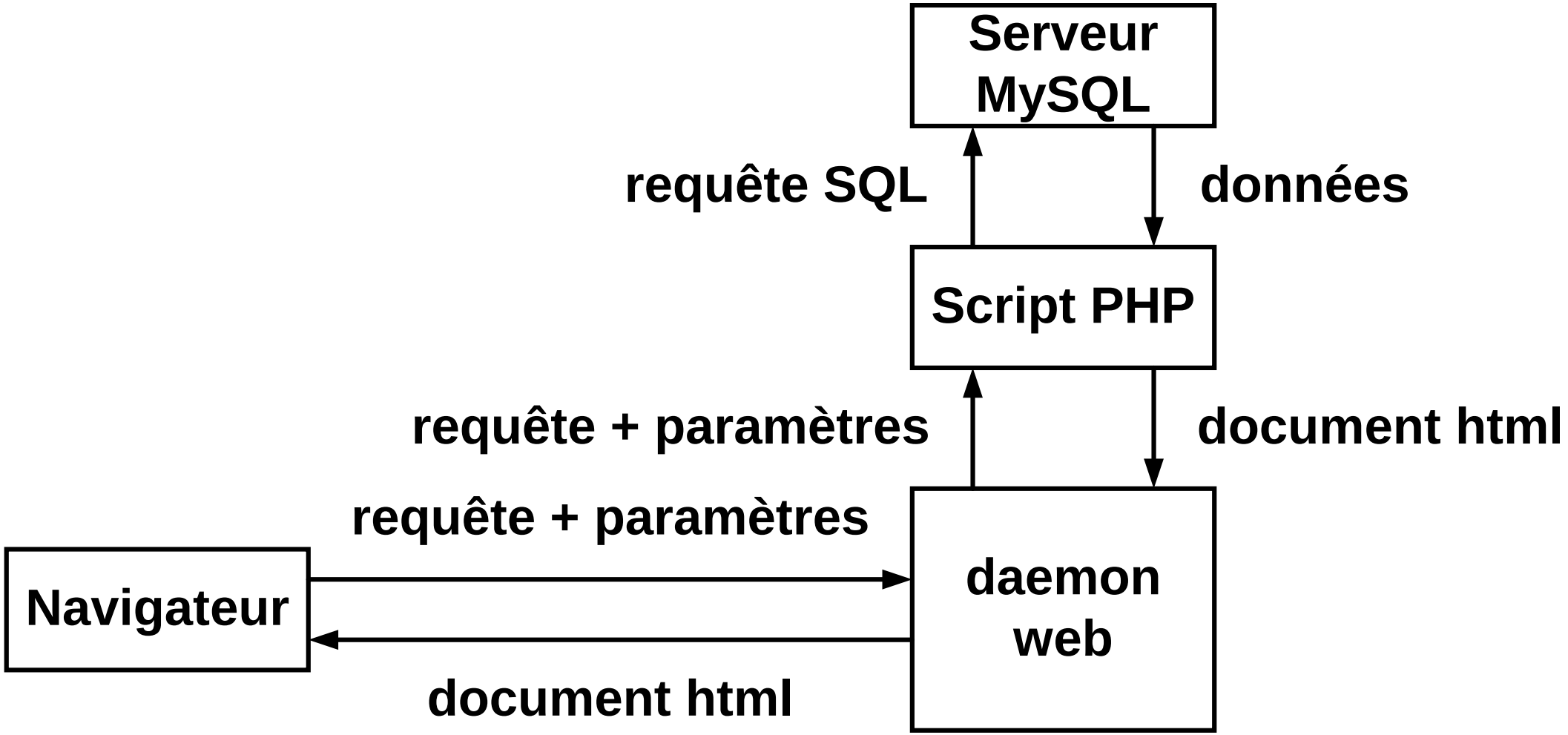
  **Sercices web accessibles via des applications qui permettent de publier, gérer et rechercher l'information**

  **Protocoles http, html et SOAP *(Simple Object Access Protocol)***

# *World-Wide Web (5)*



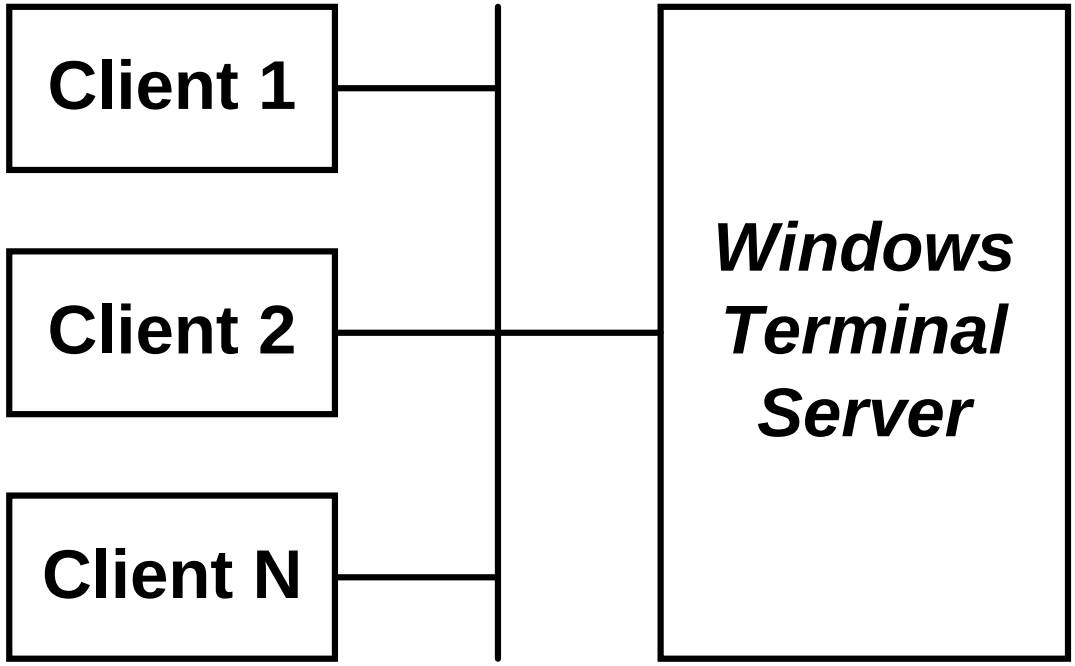
- **Configuration du travail de diplôme Dizon 2003 :**



# *Thin client (1)*



- **L'architecture *thin client* (client léger) n'est en fait qu'une évolution du modèle terminal "non-intelligent" - serveur capable d'offrir une interface graphique avec système de fenêtres**



# *Thin client (2)*



- **Minimum de fonctionalité (clavier - écran) côté client**
  **→ Administration simplifiée du poste client (DHCP, …)**

- **Chaque application (Word, …) n'est installée qu'une seule fois côté serveur**

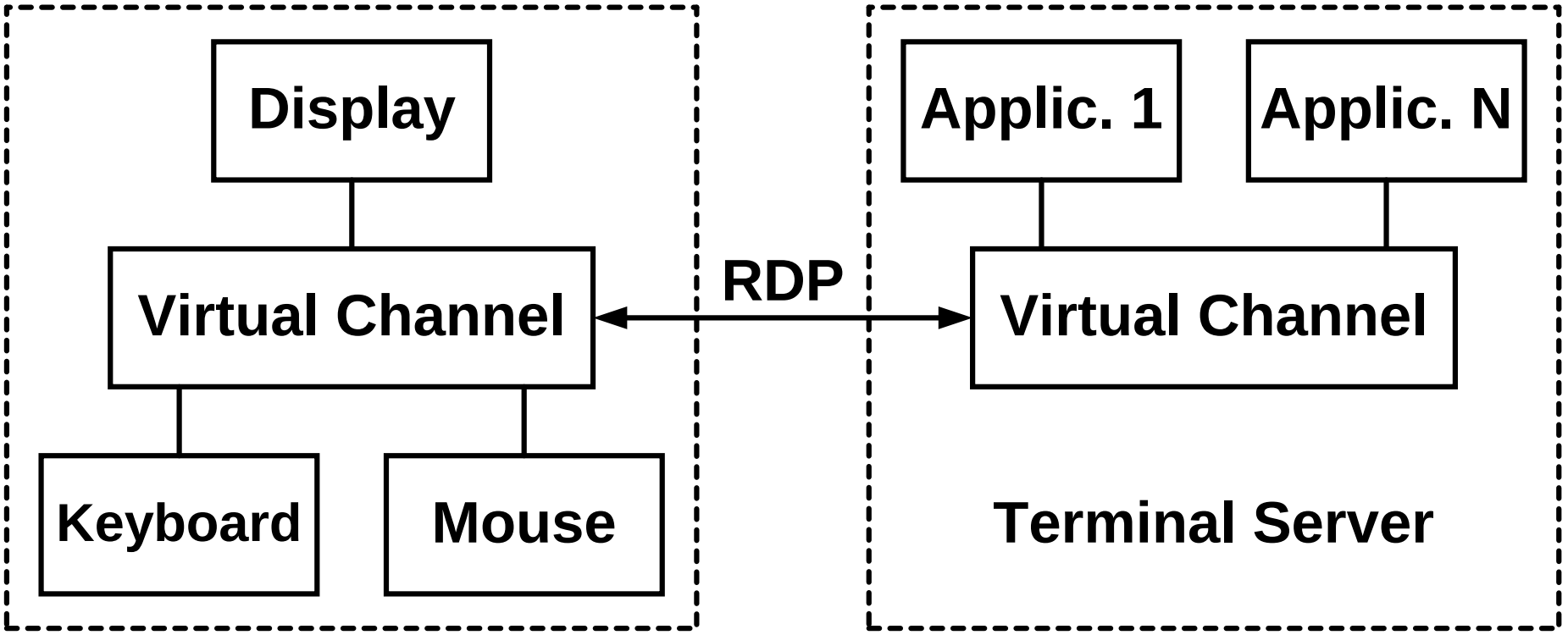
- **Aucune donnée personnelle mémorisée sur le poste qui ne possède pas de mémoire de masse**

- **Unicité des données garantie quel que soit le poste client (poste de travail dans l'entreprise, PC à domicile, portable, PDA, …)**

# *Thin client (3)*



- **Un protocole comme RDP *(Remote Desktop Protocol)* transmet les commandes (clavier, souris) ainsi que les écrans *(bitmap)* à afficher**




- **Mobilité : réseau sans fil, *wireless*, …**

# *Personal Digital Assistant (PDA)*



- **PalmOS** — **33 MHz Motorola 160x160**
  **USB, iRDA**

- **Pocket PC2002** — **206 MHz Intel 240x320 (64k)**
  **USB, iRDA, Bluetooth**
  **Ecran tactile**
  **Reconnaissance d'écriture**
  **Haut-parleur + microphone**
  **Enregistreur vocal**
  **Lecture MP3, …**
  **Terminal Services**

# *Modèle client – serveur (1)*



**Considérons l'interrogation d'une base de données (slide 23)**

**Le traitement peut être décomposé en 3 parties :**

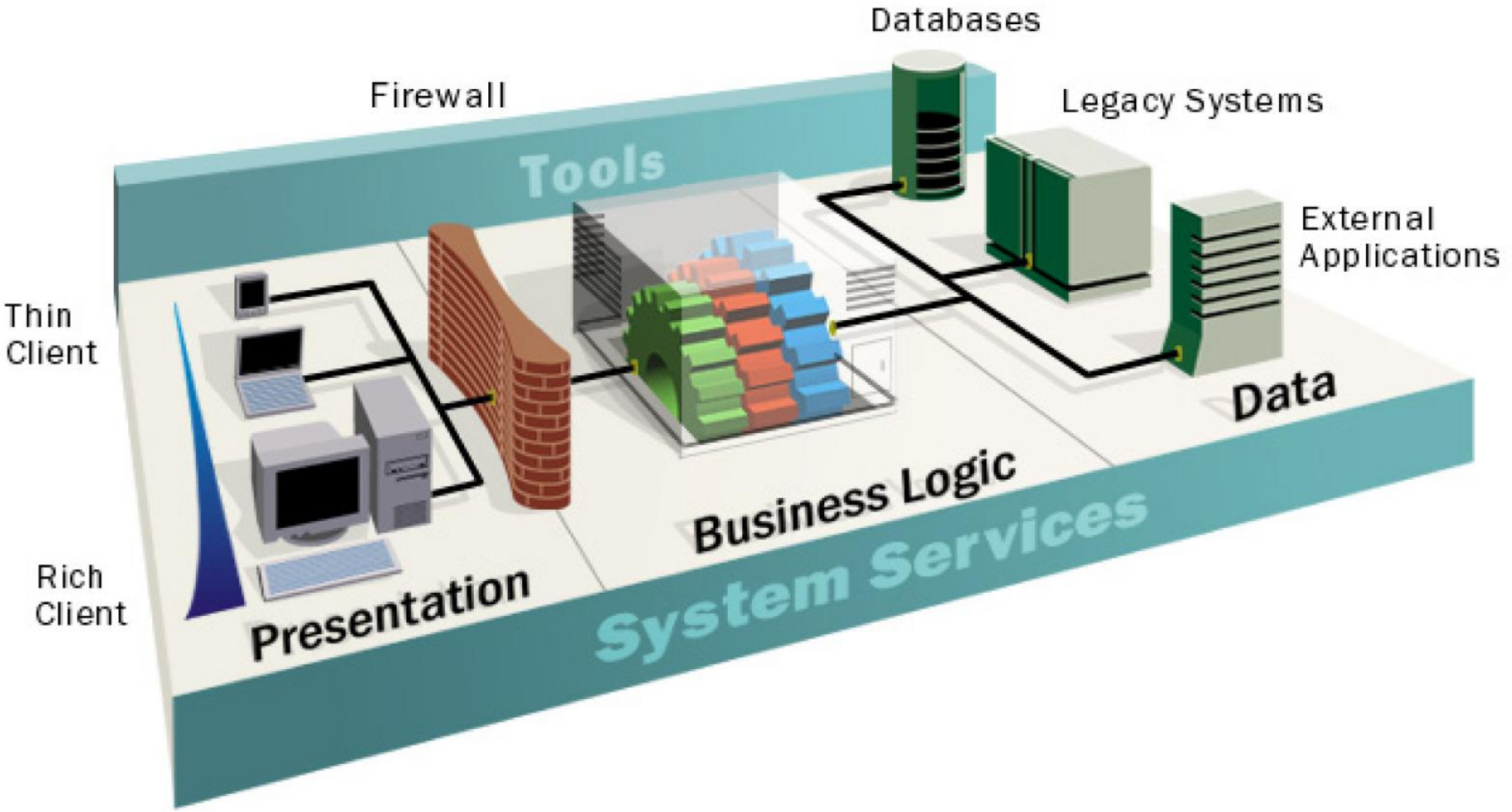

**1** **Dialogue avec l'utilisateur : masque d'écran, aide en ligne**

**2** **Application : préparation des requêtes d'interrogation de la base de données, calculs, préparation des résultats à afficher**

**3** **Accès à la base de données et aux fichiers**

# *Modèle client – serveur (2)*

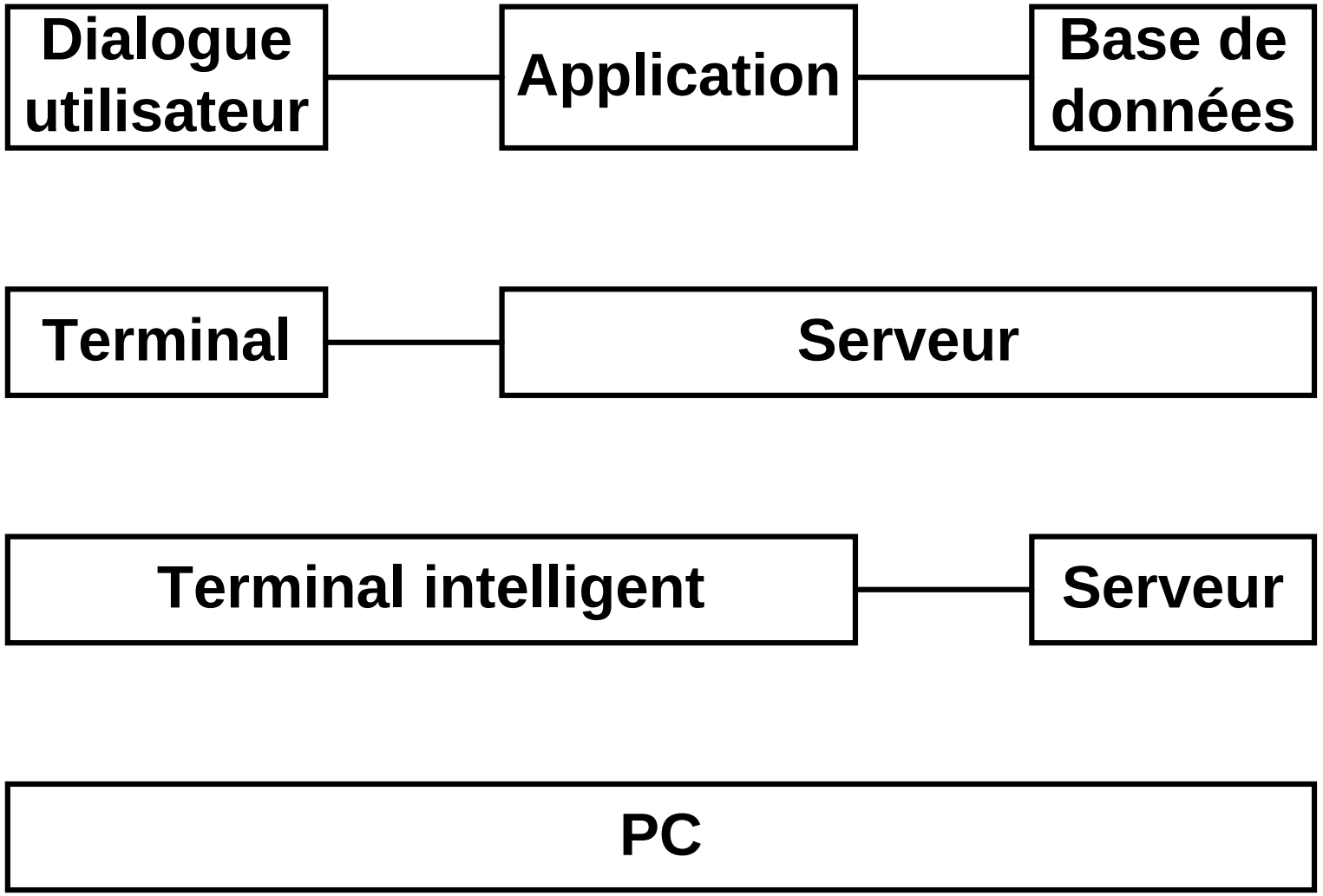
## Architecture client – serveur de la plate-forme Microsoft



# *Modèle client – serveur (3)*



**Diverses architectures sont possibles :**



# *Modèle client – serveur (4)*



- **Le modèle client-serveur relève de la problématique des architectures distribuées**

- **On cherche à :**
  - **Répartir les traitements sur différents processeurs**
  - **Minimiser le trafic sur le réseau**
  - **Offrir une interactivité suffisante**

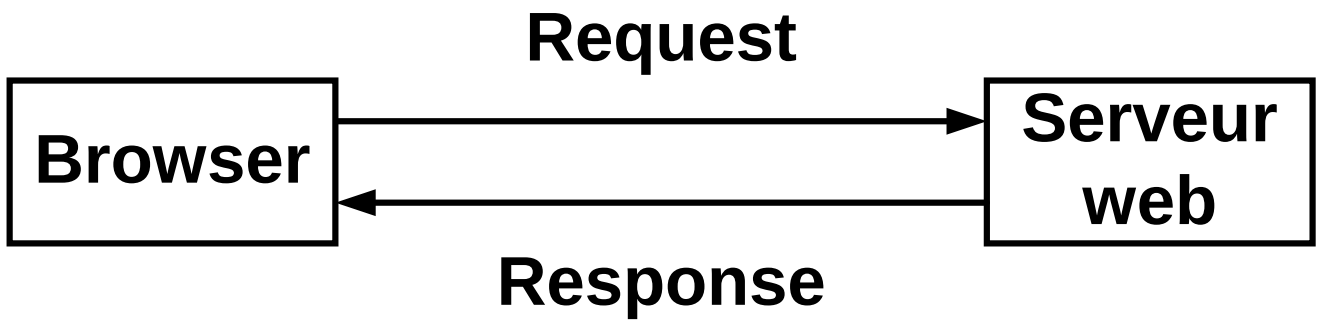
- **La relation client – serveur est de type maître-esclave et non d'égal à égal *(peer to peer)* comme dans le cas d'un traitement distribué général**

# *Modèle client – serveur (5)*



- **Les interfaces entre clients et serveurs doivent assurer des échanges entre modules indépendamment des architectures physiques**

- **Illustration avec le protocole HTTP *(HyperText Transfer Protocol)* utilisé entre navigateur *(browser)* et serveur *web* :**




→ **Get http://www.td.unige.ch/default.html**

← **Response**

# *Applications TCP/IP (1)*



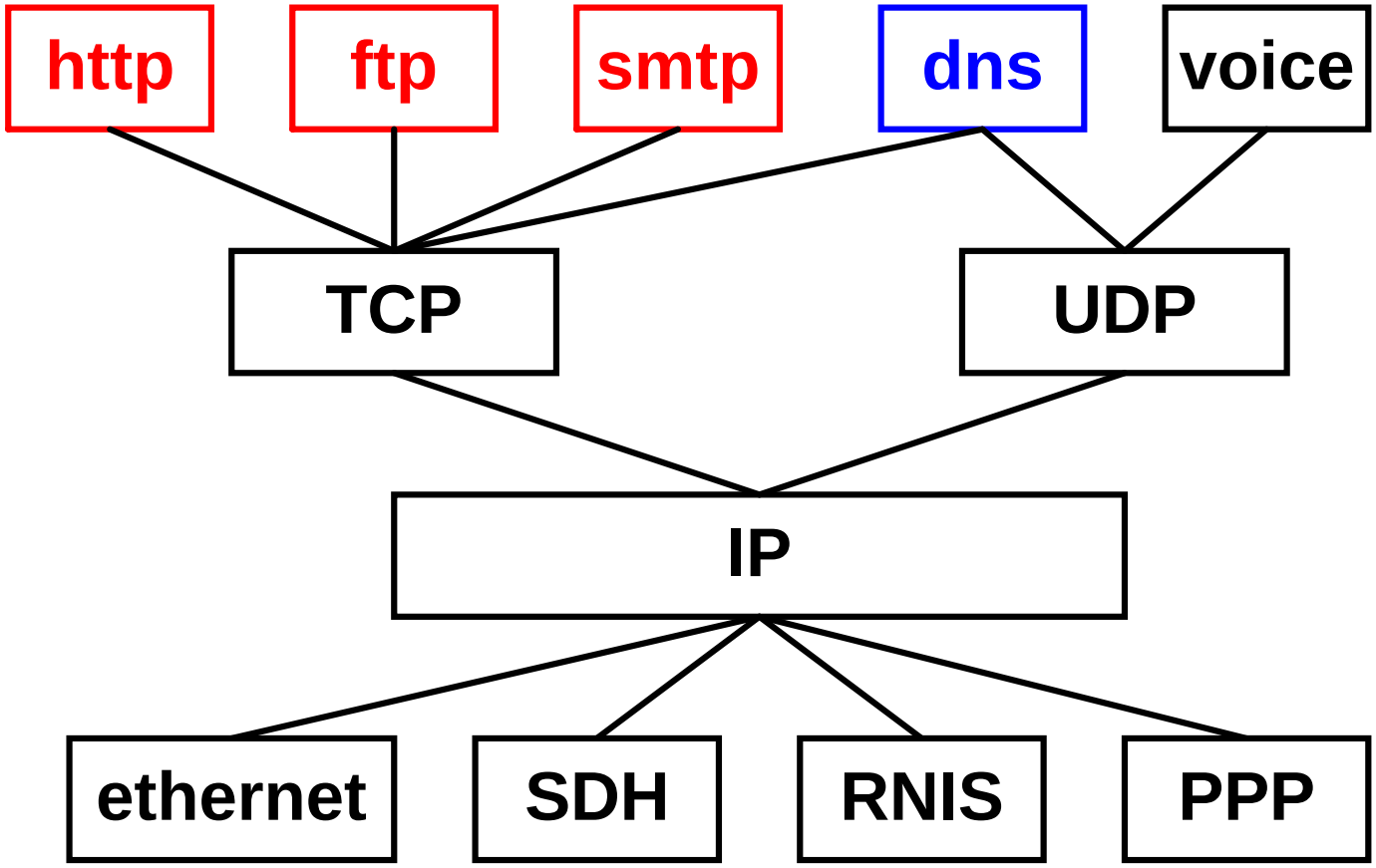
- **Emulation de terminal** **telnet**
- **Transfert de fichier** **ftp – tftp**
- **Messagerie** **smtp – pop**
- **Hypertext** **http – html**

**Par défaut, toutes ces applications n'intègrent aucun mécanisme de chiffrement**

***Username – password* sont transmis en clair**

# Applications TCP/IP (2)

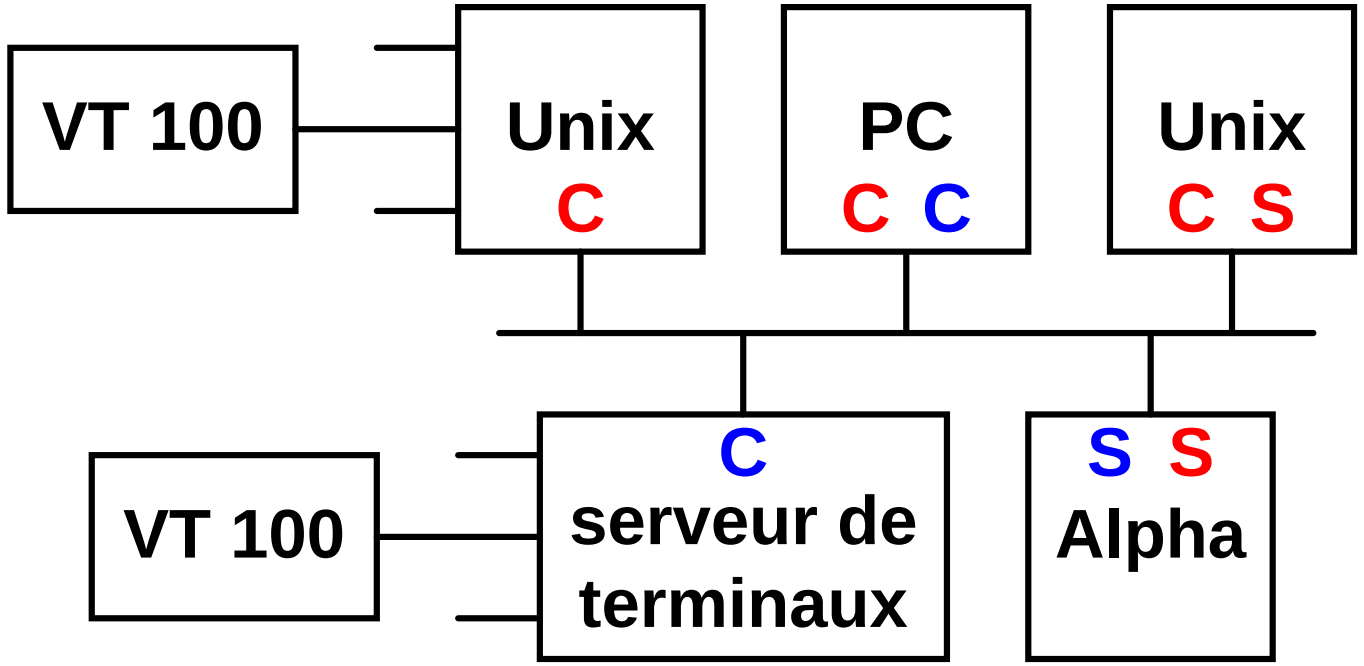




# *Emulation de terminal (illustration)*

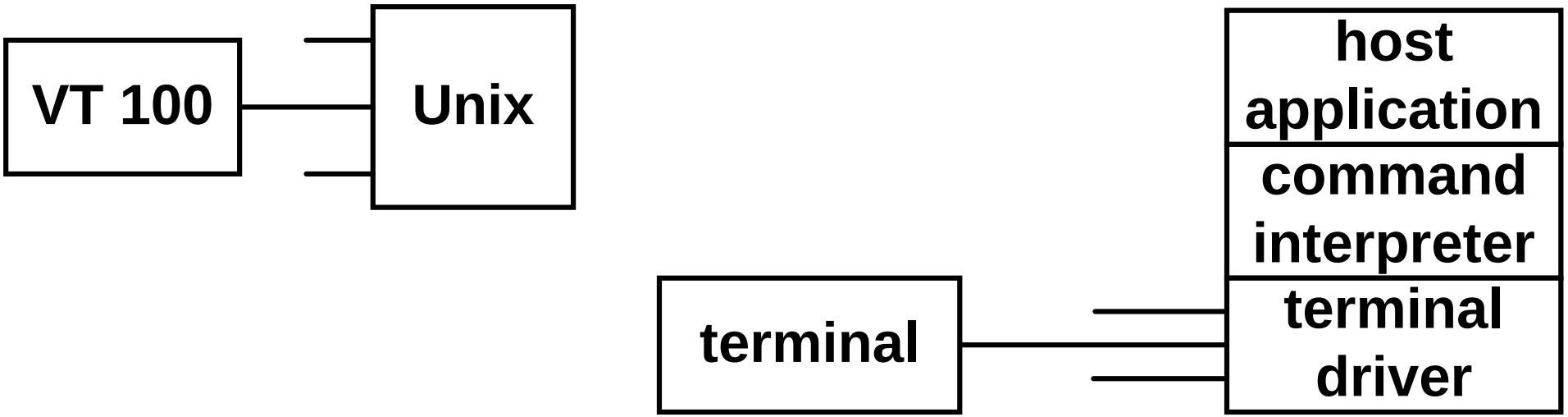


- **L'exemple suivant montre des configurations possibles dans un réseau hétérogène du service d'émulation de terminal basé sur le protocole public *telnet* et sur le protocle propriétaire DEC LAT :**



# Architecture Unix



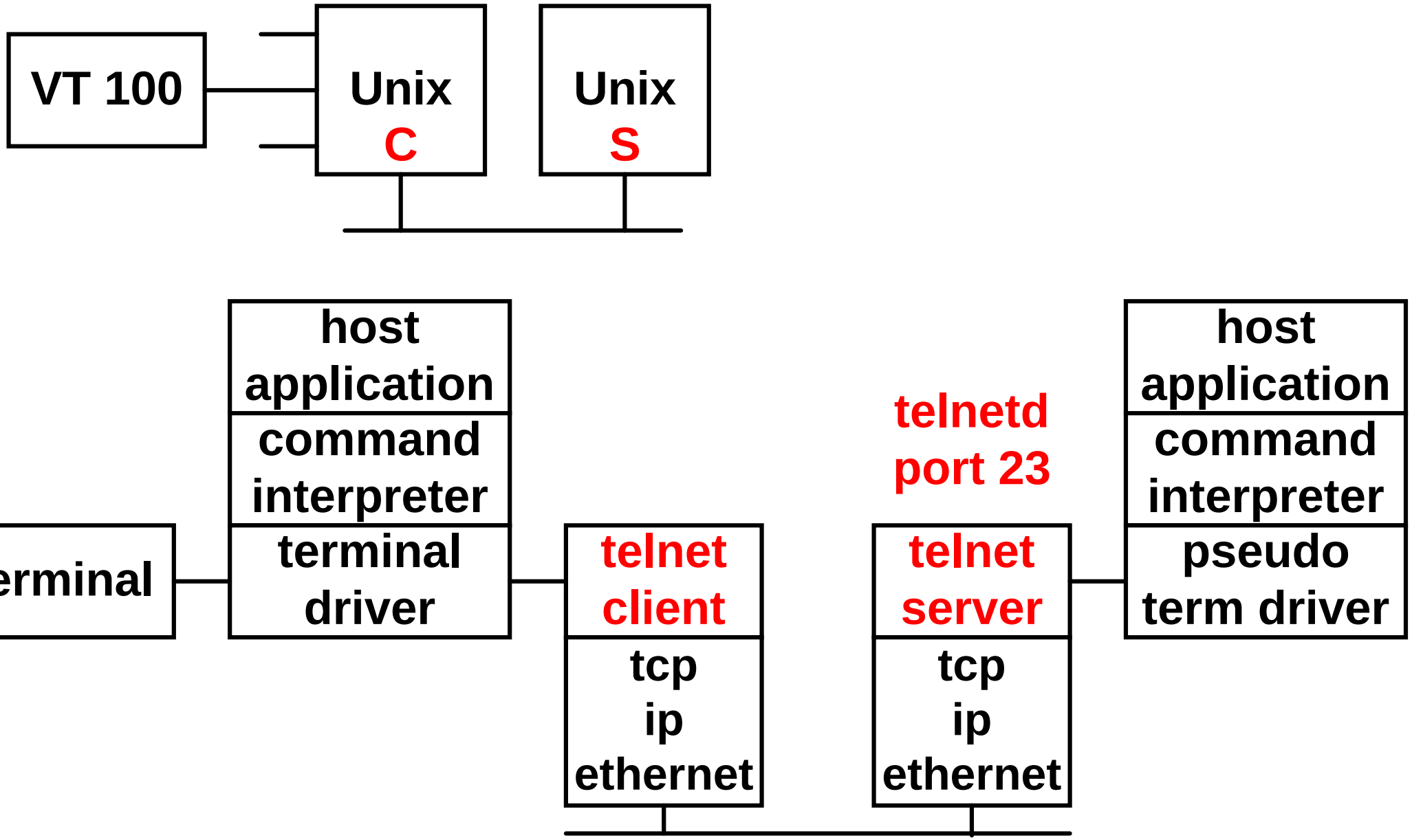



- L'utilisateur sur son terminal, disposant d'un compte utilisateur, peut accéder au serveur par le service *remote login*

- La liaison terminal - serveur, souvent asynchrone pour des raisons de coût, supporte des terminaux physiques comme VT-100

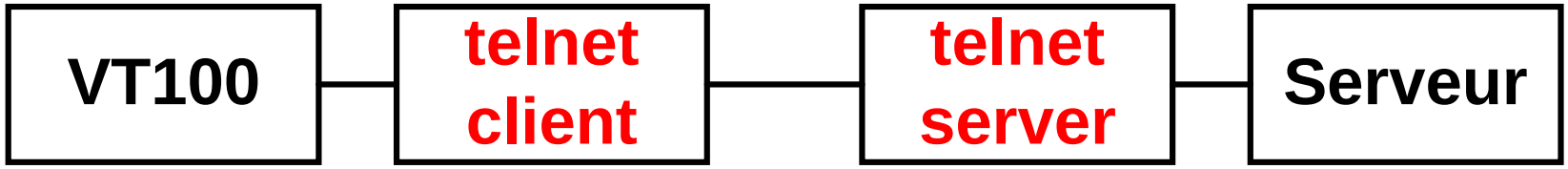
# *Telnet dans l'architecture Unix*



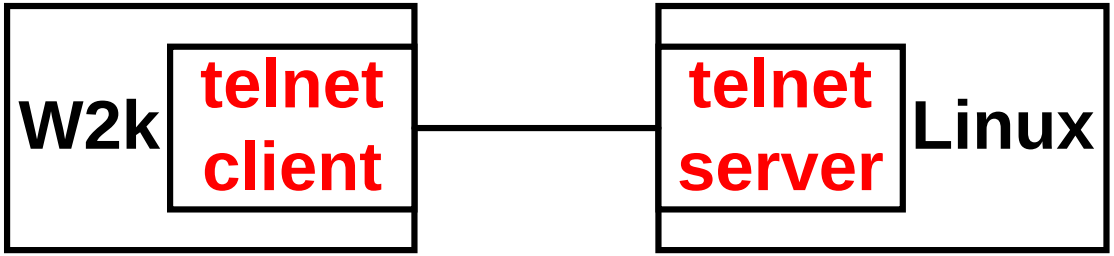


# Variantes





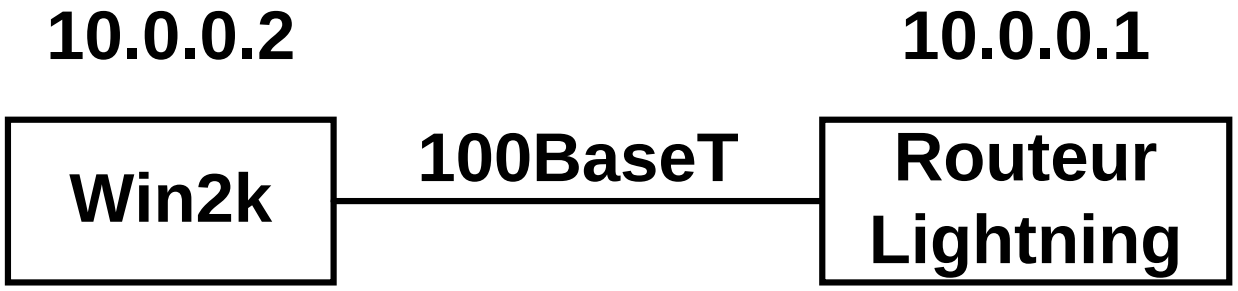

- Telnet a été conçu pour fonctionner entre n'importe quel serveur (c'est-à-dire tout système d'exploitation) et n'importe quel terminal (VT100, …)




- Connexion telnet entre client W2k et serveur Linux

# Configuration du routeur






```
Run:telnet 10.0.0.1
Linux 2.4.5 (MultiCom)
MultiCom login:abc
Password:123
/:
```

# Analyse de protocole



A ← → B

…
← Data Linux 2.4.5 (MultiCom)
← Data MultiCom login:
→ Data a
← Data a
…
← Data Password:
→ Data 1

# *Caractéristiques de telnet*



- **La connexion à distance (*remote login*) est une application pratique qui permet de se connecter à distance via le réseau sur un ordinateur pour autant que l'on dispose d'un compte utilisateur**

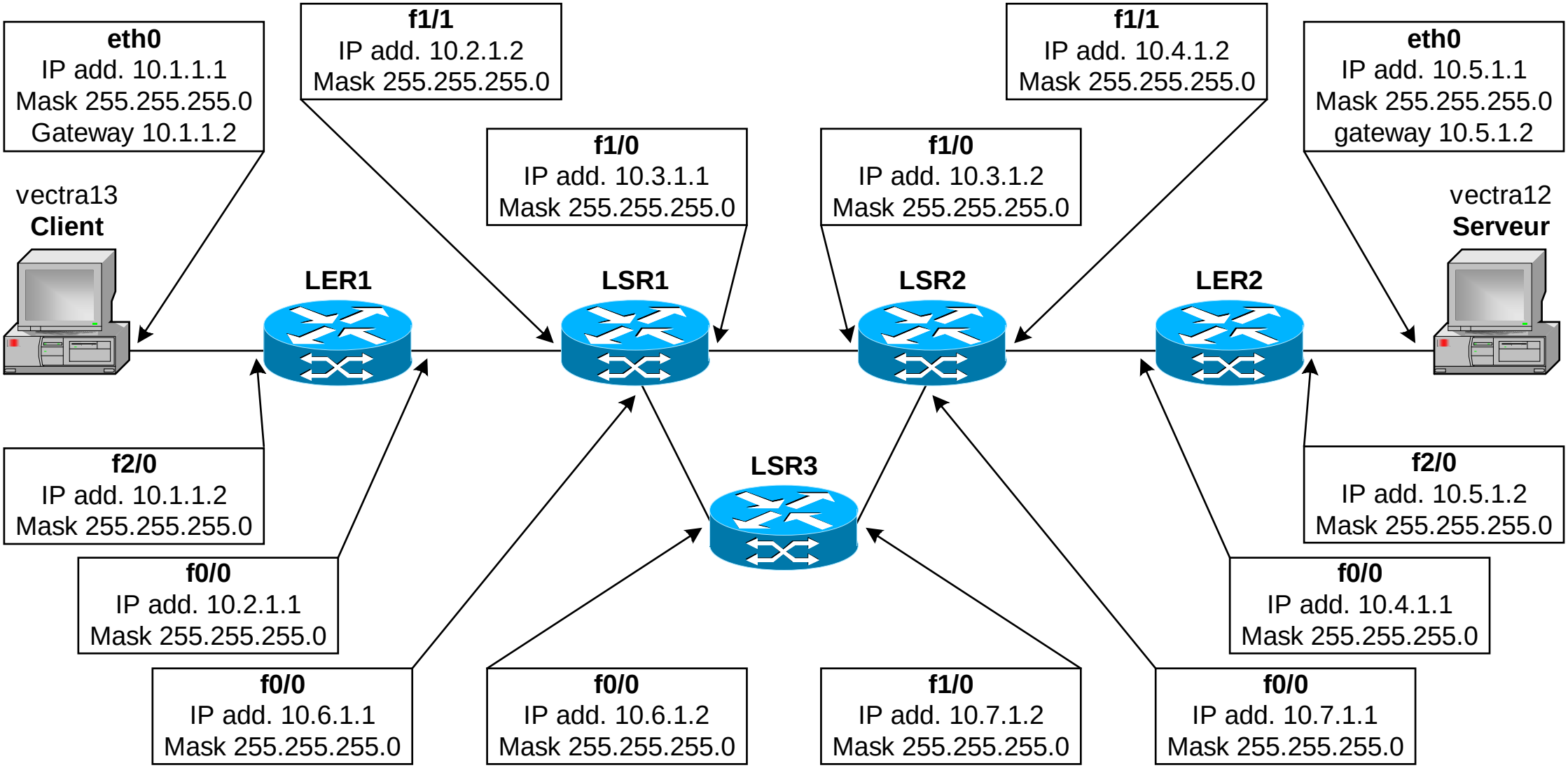
- **Elle se limite à des applications mode caractère**

- **Elle facilite notamment l'administration de composants réseau comme *hub, bridge, switch, router,* …**

- ***Username – Password* sont transmis en clair**

# Diplôme MPLS – Cisco 7200





# *File Transfer Protocol (FTP)*



- **Protocole de transfert de fichier**

- **2 connexions TCP sont nécessaires**

| | | |
|---|---|---|
| **Connexion de contrôle** | **21** | **E L** |
| **Connexion de donnée** | **20** | **T** |

- **Compte utilisateur**

- **Compte *anonymous***

# *FTP anonyme (1)*



- **Name anonymous**
- **Certains serveurs FTP anonymes exigent que le client possède un nom de domaine valide**
- **Le serveur effectue une requête DNS inverse sur la base de l'adresse IP reçue et compare le résultat avec l'adresse email entrée**

# FTP anonyme (2)



```
C:\>ftp ftp.luth.se
Connected to ftp.ludd.luth.se.
220 gort.ludd.luth.se FTP server ready.
User (ftp.ludd.luth.se:(none)): anonymous
331 Guest login ok, type your name as password.
Password:
230-    Welcome to the top ftp archive of Europe!

        The localtime is: Tue Nov 30 10:11:59 2004
        You are user #114 in your class (max 350)



230 Guest login ok, access restrictions apply.
ftp>
```

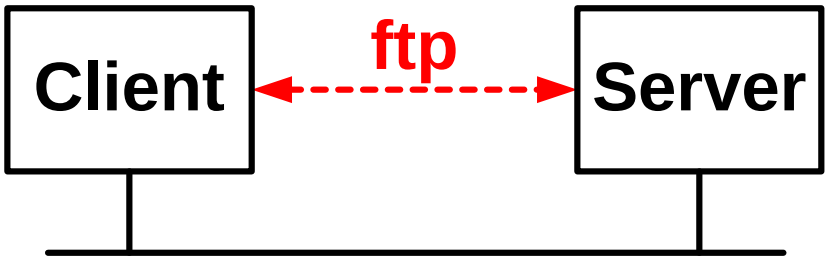
# *Labo B2 : Analyse du protocole FTP*

- **File Transfer Protocol**

**ftp.tdeig**

**10.1.1.2**

**SmartFTP (GUI)**
**ftp (DOS)**
**ftp depuis IE**




**Analyses de protocole**
**connexions TCP ?**
**ports utilisés ?**
**commandes ftp ?**
**réponses ftp ?**
**mode passif**

# FTP : commandes



- **USER** nom de l'utilisateur (identification)
- **PASS** mot de passe en clair (authentification)

- **LIST** liste des fichiers & répertoires
- **TYPE** A:ASCII I:Image
- **GET** C ← S RETR
- **PUT** C → S STOR

- **PORT**
- **PASV**
- **QUIT**

# FTP : réponses



- 220 *Service ready for new user*
- 331 *User name okay, need password*
- 230 *User logged in, proceed*

- 200 *Command okay*
- 150 *File status okay*
- 257 *"PATHNAME" created*

- 226 *Closing data connection*
- 227 *Entering Passive Mode (h1,h2,h3,h4,p1,p2)*
- 125 *Data connection already open; transfer starting*

# FTP : authentification



| Client | ← → | Server |
|---|---|---|
| ftp host | → TCP SYN 21 | |
| | ← 220 Bienvenue … | |
| ftp> username | → USER anonymous | |
| | ← 331 Anonymous access allowed, send … | |
| ftp> password | → PASS guest | |
| | ← 230 Anonymous user logged in | |

# FTP : dir (transfert normal)



| Client | ← → | Server |
|---|---|---|
| ftp> dir | → PORT 129,194,184,97,4,106 | |
| | ← 200 PORT command successful | |
| | → LIST | |
| | ← TCP SYN 1130=4x256+106 | 20 |
| | … | |
| | ← 150 Opening ASCII mode for /bin/ls | |
| Data received | | |
| | ← 226 Transfer complete | |

# FTP : put (transfert normal)



| Client | ← → | Server |
|---|---|---|
| ftp> put … | → PORT 129,194,184,97,4,107 | |
| | ← 200 PORT command successful | |
| | → STOR abc.txt | |
| | ← TCP SYN 1131=4x256+107 | 20 |
| | … | |
| | ← 150 Opening ASCII mode for abc.txt | |
| Data send | | |
| | ← 226 Transfer complete | |

# FTP : dir (mode passif)



| Client | ← → | Server |
|---|---|---|
| ftp> dir | → PASV | |
| | ← 227 Passive mode (129,194,184,212,5,0) | |
| | → LIST | |
| | → TCP SYN 1280=5x256+0 20 | |
| | … | |
| | ← 150 Opening ASCII mode for /bin/ls | |
| Data received | | |
| | ← 226 Transfer complete | |

# *FTP : put (mode passif)*

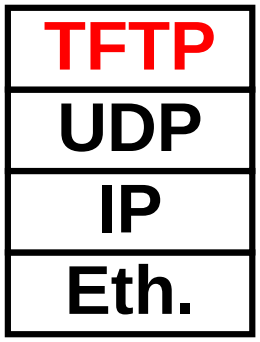


| Client | ← → | Server |
|---|---|---|
| ftp> get … | → PASV | |
| | ← 227 Passive mode (129,194,184,212,5,1) | |
| | → RETR readme.txt | |
| | → TCP SYN 1281=5x256+1 | 20 |
| | … | |
| | ← 150 Opening … for readme.txt | |
| Data received | | |
| | ← 226 Transfer complete | |

# *Trivial File Transfer Protocol (TFTP)*



- **Facilement mis en mémoire morte (ROM)**
- **Utilise UDP port 69**
- **Ne fournit pas de service d'identification**
- **Fournit sa propre gestion des erreurs**
- **Format des messages**
  - **Read request (RRQ)**
  - **Write request (WRQ)**
  - **Data (DATA)**
  - **Acknowledgment (ACK)**
  - **Error (ERROR)**



# *Messagerie*



- **Les systèmes de messagerie constituent un moyen pratique d'échange d'information**

- **Adresse de courrier électronique *(electronic mail, email, E-Mail)* sur carte de visite**

- **Format du message :**

  **from gerald.litzistorf@hesge.ch**

  **to ventura@eivd.ch**

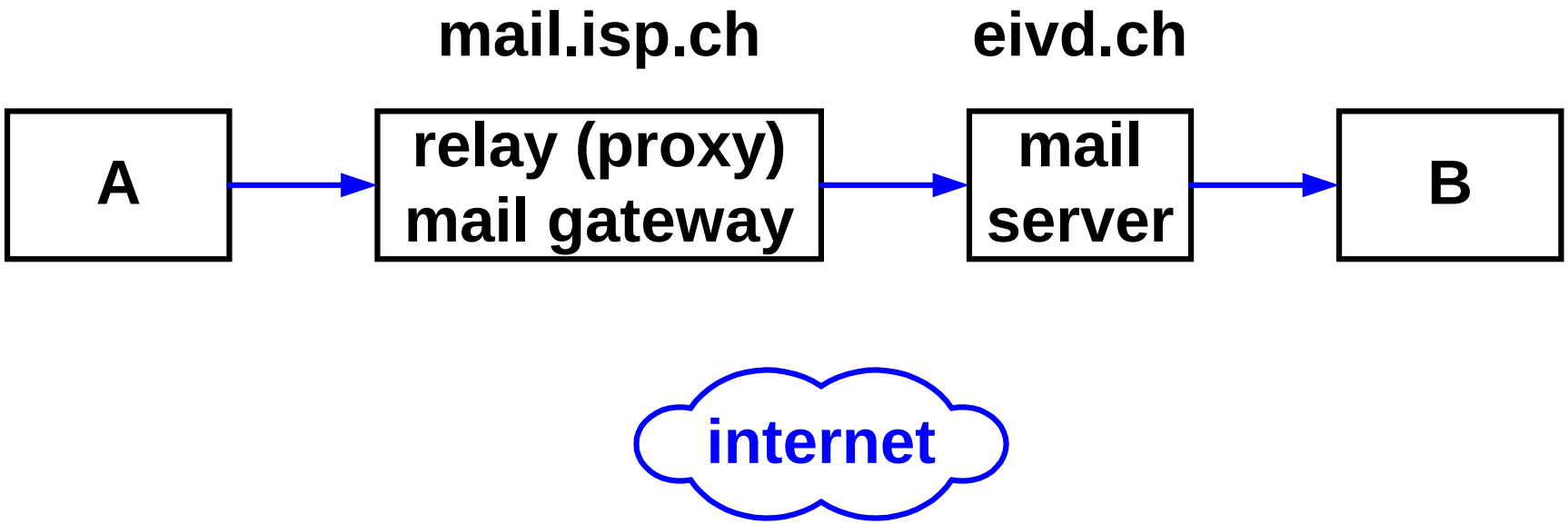
  **subject confirmation de la se'ance**

  **text je te confirme ...**

# *Architecture*



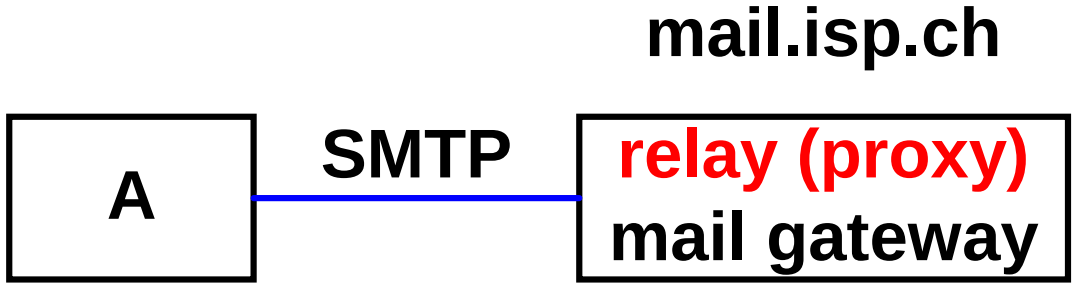
- A émet un message destiné à B




- A et B ne sont jamais connectés *(offline connexion)*
- A ou B peuvent être éteints
- DNS permet de retrouver l'adresse du serveur SMTP (MX)

# *Emission d'un message*

- **Fonction de relais *(relay)* pour l'émission d'un message**




- **Sécurité : vérifier que l'expéditeur est autorisé à émettre**
  **→ compte utilisateur**
  **sinon *spam* (message non solicité)**

- **Protocole SMTP *(Simple Mail Transfer Protocol)***

- **Transfert fiable grâce à TCP**

# *Echange smtp (1)*



A ← →

→ TCP SYN win:8760 src:1049 dst:25 (SMTP)

← TCP A,S win:8760 src:25 dst:1049

→ TCP ACK

← SMTP-Rsp 220 infomaniak.ch MailSite ...

→ SMTP-Cmd HELO HP1

← SMTP-Rsp 250 OK

# Echange smtp (2)



A ← →

→ SMTP-Cmd MAIL FROM:litzistorf@eig.unige.ch
← SMTP-Rsp 250 litzistorf@eig.unige.ch OK
→ SMTP-Cmd RCPT TO: ventura@eivd.ch
← SMTP-Rsp 250 ventura@eivd.ch OK
→ SMTP-Cmd DATA
← SMTP-Rsp 354 Ready for data

# *Echange smtp (3)*



A ← →

→ **SMTP-Data** From: <litzistorf@eig.unige.ch>
To: <ventura@eivd.ch>
Subject: essai
Date: ...7 Jan 2002 14:52:37
Message-ID:
MIME-Version: 1.0
Content-Type: text/plain;
charset="iso-8859-1"

# *Echange smtp (4)*



A ← →

→ SMTP-Data ...

```
Content-Transfer-Encoding: 7bit
X-Priority: 3 (Normal)
X-MSMail-Priority: Normal
X-Mailer: MS Outlook IMO, Build...
X-MIMEOLE: ...
Importance: Normal
Data = bonjour
```

# *Echange smtp (5)*



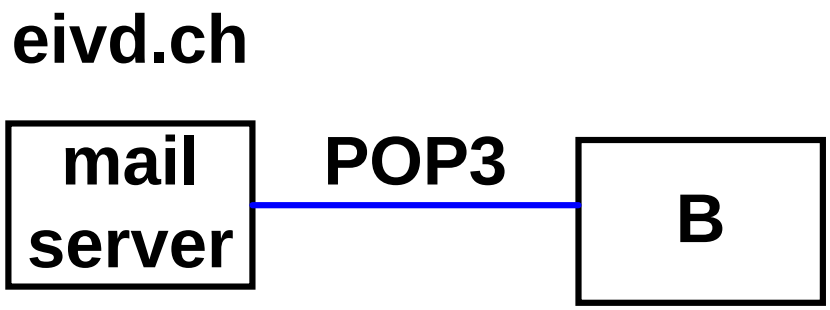
A ← →

→ SMTP-Data .
← SMTP-Rsp 250 Message received OK
→ SMTP-Cmd QUIT
← SMTP-Rsp 221 infomaniak.ch closing

# *Réception d'un message*



- **Serveur de messagerie pour la réception des messages**




- **Transfert entre le serveur et le destinataire avec les protocoles POP 2 et 3 *(Post Office Protocol)***

# Echange pop (1)



B ← →

→ TCP SYN win:8760 src:1051 dst:110 (POP)

← TCP A,S win:8704 src:110 dst:1051

→ TCP ACK

← POP-Rsp +OK POP3D ...

→ POP-Cmd USER ventura

← POP-Rsp +OK password please

→ POP-Cmd PASS en clair

# Echange pop (2)



B ← →

← POP-Rsp +OK Mailbox open, 1 message
→ POP-Cmd STAT
← POP-Rsp +OK 1 2560 msgid size
→ POP-Cmd UIDL
← POP-Rsp +OK 1 message
→ POP-Cmd RETR 1
← POP-Rsp +OK 1581 octets

# *Echange pop (3)*



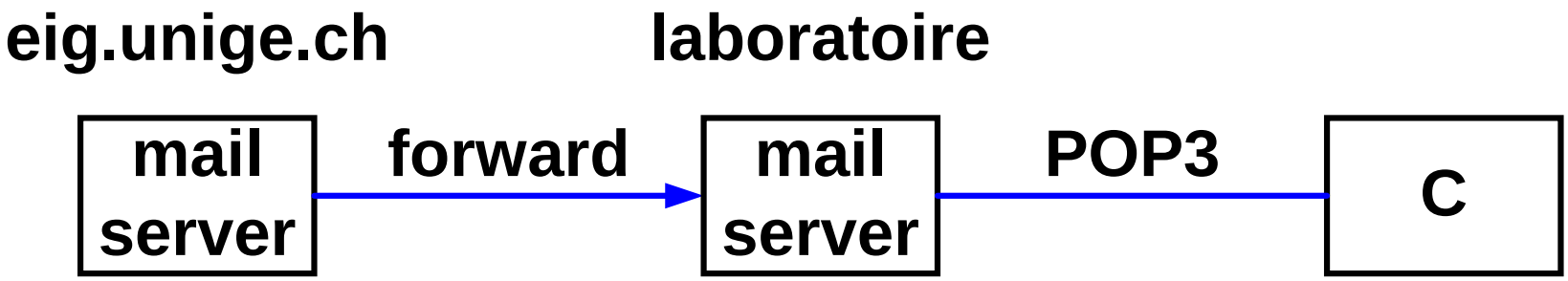
```
B ← →


← POP-Rsp
        Return-path:litzistorf@EIG.UNIGE.CH
             ...
→ POP-Cmd DELE 1
← POP-Rsp +OK Message deleted
→ POP-Cmd QUIT
← POP-Rsp +OK bye
```

# *Fonction de redirection – forward*



- **L'utilisateur conserve son adresse email**




- **mais redirige ces messages sur un serveur spécifique**

# *Fichier attaché*



- **Possibilité d'attacher un fichier au message avec le protocole MIME *(Multi-purpose Internet Mail Extensions)***

- **Exemples**
  - **application/pdf**
  - **audio/x-mpeg**
  - **image/gif**
  - **video/mpeg**
  - **text/plain**

# *Caractéristiques de la messagerie (1)*



- **Systèmes *offline* → commutation de messages**
  **Les messages sont stockés sur disque**

- **Temps de transmission du message de A → B ?**
  **Faible priorité pour les services de messagerie**
  **Sécurité → *firewall, virus scan, ...***

- **Rappel si destinataire pas atteignable (toutes les heures par ex.)**

- **Indication de message non délivré (après 3 jours par ex.)**

# *Caractéristiques de la messagerie (2)*



- **Différentes interfaces utilisateur (convivialité, simplicité,...)**

- **Plusieurs familles de protocoles de messagerie**
  **internet**
  **propriétaire (Microsoft, IBM, ...)**
  **public (X.400)**
  **sans fil (pager, SMS, …)**

  **→ Interconnexion des systèmes de messagerie**

- **Sécurité → signature, chiffrement**
  **→ Secure MIME (S/MIME), PGP, …**

# *Caractéristiques de la messagerie (3)*



- **Très simple d'envoyer un email forgé avec une fausse adresse**
- **Solution : envoyer des emails signés, *reverse MX lookup*, …**

- **De nombreux relais (> 100'000 ouverts en oct 2002) facilitent l'envoi des spams**
- **Les virus comme sobig, … installent des relais smtp sur leurs victimes**
- **Solution : filtrer le trafic entrant avec la liste noire maintenue par www.ordb.org**

# *HyperText Transfer Protocol – HTTP*



- **Protocole sans connexion entre navigateur *(browser)* et serveur (port 80 par défaut)**

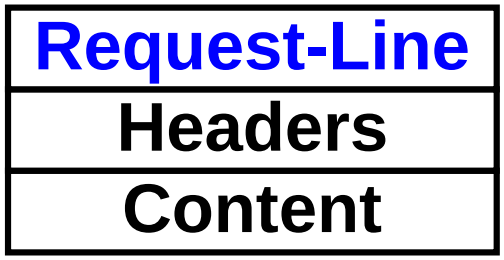
- **Structure d'une commande**

  **Request-Line** **obligatoire**

  **(( general-header | request-h. | entity-h. ) CRLF)**

  **CRLF**

  **[ message-body ]**



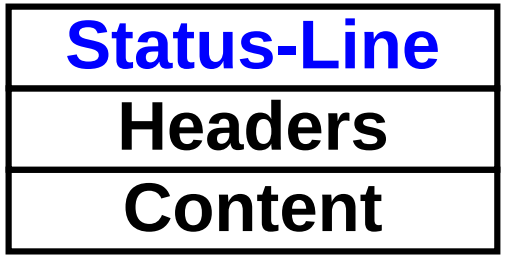

- **Structure d'une réponse**

  **Status-Line**

  **(( general-h. | response-h. | entity-h. ) CRLF)**

  **CRLF**

  **[ message-body ]**



# HTTP : commande



```
Request-Line =
Method SP Request-URI SP HTTP-Version CRLF
```

- **Example 1 with absolute URI**

```
GET http://www.google.ch/default.htm HTTP/1.1\r\n
Accept: */*\r\n
User-agent: ... (compatible; MSIE 6.0)\r\n
\r\n
```

```
Method          GET
Request-URI     http://www.google.ch/default.htm
HTTP-Version    HTTP/1.1
```

# HTTP : variantes



- Example 2 with host header

```
GET image1.gif HTTP/1.1\r\n
Host: www.td.unige.ch\r\n
Accept: */*\r\n
User-agent: ... (compatible; MSIE 6.0)\r\n
```

- Exemple 3

```
POST image1.gif HTTP/1.1\r\n
Host: www.td.unige.ch\r\n
Content-Length: XX
```

- Exemple 4

```
HEAD www.unige.ch HTTP/1.1\r\n
```

démo

# *HTTP : methods*



- **GET** — **Retrieve information identified by the URI**
  **Used to retrieve an HTML document**
- **HEAD** — **Retrieve meta-information about the URI**
  **Used to find out if a document has changed**
- **POST** — **Send information to a URI and retrieve result**
  **Used to submit a form**
- **PUT** — **Store information in location named by URI**
- **DELETE** — **Remove *entity* identified by URI**

# HTTP : réponse



```
Status-Line =
HTTP-Version SP Status-Code SP Reason-Phrase CRLF

HTTP/1.1 200 OK
Date: Wed, 12 Nov 2003 14:44:36 GMT
Server: Apache/1.3.23
Last-Modified: Fri, 07 Nov 2003 17:15:03 GMT
Content-Type: text/html
Content-Length: 1145

<html><head><title>…</html>
```

# HTTP : header



| | |
|---|---|
| `Accept:` | indique la liste des types de données supportées par le client |
| `User-agent:` | identifie le logiciel de navigation |
| `Host:` | indique le nom du serveur<br>+ numéro de port éventuel |
| `Server:` | identifie le logiciel serveur |
| `Date:` | date de génération de la réponse |
| `Last-Modified:` | date de la dernière modification |

# *Codes de réponse (1)*



- **1xx** messages d'information (pas utilisé)

- **2xx** Commande reçue et traitée par le serveur
  - **200** Requête s'est déroulée correctement
  - **201** Requête OK et création d'un nouveau document
  - **202** Requête acceptée, traitement en cours
  - **204** Requête OK, mais aucune information à envoyer

- **3xx** Redirection, nouvelle requête nécessaire pour accéder au document demandé
  - **301** Ressource demandée a été déplacée de façon permanente
  - **302** Ressource demandée a été déplacée de façon temporaire

# *Codes de réponse (2)*



- **4xx** **Erreur due au client**

  **400** **Erreur de syntaxe**

  **401** **Accès à la ressource exige une authentification**

  **403** **Accès à la ressource interdit**

  **404** **Ressource n’existe pas**

- **5xx** **Erreur due au serveur**

  **500** **Erreur interne**

  **503** **Serveur indisponible**

# Echange http (1)



```
→ TCP SYN win:8760  src:1059 dst:80 (HTTP)
← TCP A,S win:17520 src:80   dst:1059
→ TCP ACK

→ HTTP: GET Request (from client port 1059)
  HTTP: Request Method = GET
  HTTP: Uniform Resource Identifier = /
  HTTP: Protocol Version = HTTP/1.1
  HTTP: If-Modified-Since = 29 Nov 2001
                            09:42:59 GMT
  HTTP: User-Agent = ...
  HTTP: Host = www.td.unige.ch
  HTTP: Connection = Keep-Alive
```

# Echange http (2)



```
← HTTP: GET Response (to client port 1059)
  HTTP: Protocol Version = HTTP/1.1
  HTTP: Status Code = 0x0130
  HTTP: Reason = Not Modified
  HTTP: Server = Microsoft-IIS/5.0
  HTTP: Date = 20 Jan 2002 09:42:05 GMT
  HTTP: Content-Length = 0
```

# Echange http (3)



```
→ HTTP: GET Request (from client port 1059)
  HTTP: Request Method = GET
  HTTP: URI =/
  HTTP: Protocol Version = HTTP/1.1
  HTTP: Accept = */*
  HTTP: Host = www.google.ch
  HTTP: Referer = http://www.td.unige.ch/
```

**Referer …** pointe vers l'URI de la page à partir de laquelle le document est demandé

# *Sécurité des applications*



- **Authentification forte**
- **Confidentialité des données (chiffrement)**
- **Intégrité des données**

- ***Secure Shell (SSH)***

  **Solution de remplacement à telnet, ftp et les commandes R (Unix)**
- ***Secure Mime (S/Mime)***

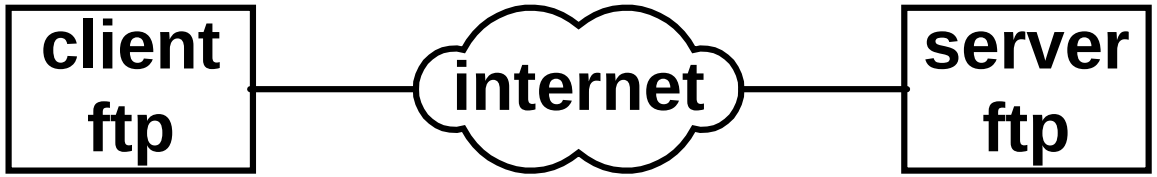
  **Sécurisation de la messagerie**
- ***Secure Socket Layer (SSL)***

  **Echange http sécurisé (https) pour application *e-commerce*, …**

# *Performances (1)*



- **L'utilisateur souhaite naturellement les meilleures performances au prix le plus bas !**
- **Améliorer sans cesse le débit utile *(throughput)* est une opération complexe !**
- **Considérons une application courante comme FTP :**



# *Performances (2)*



**Plusieurs paramètres vont influencer ce débit utile :**

- **Performances des ordinateurs (CPU, DMA, débit du disque dur, taille de la mémoire cache, optimisation des couches de protocole, carte réseau, ...)**
- **Dissymétrie (client rapide - serveur lent, ...)**
- **Caractéristique du réseau (nombres de routeurs traversés, temps aller-retour, débit binaire le plus lent,...)**

# *Performances (3)*



- **L’ingénieur en télécommunications, dans un travail d’optimisation (débit utile, coût, temps de réponse, ...) doit être capable de décomposer et de caractériser les différents composants de cette chaîne**

# *Longueur optimum (1)*



- **A tout bloc d'information correspond une longueur maximale, appelée MTU *(Maximum Transmission Unit)*, qui dépend du type de réseau (ethernet, ATM, ppp, ...)**

- **La spécification *ethernet*, par exemple, fixe cette taille à 1500 octets**

- **La couche IP, pour émettre un datagramme dont la longueur est supérieure au MTU, devra employer la fragmentation, qui consiste à casser ce datagramme en "morceaux" dont la taille ne dépasse pas ce MTU**

# *Longueur optimum (2)*



- **La RFC1191 précise quelques valeurs courantes de MTU de réseau :**

| réseau | MTU (octets) |
|---|---|
| ethernet | 1500 |
| IEEE 802.3 | 1492 |
| X.25 | 576 |
| PPP | 296 |
| Token Ring - 16 | 17814 |

# *MTU de chemin*



- **Quel MTU faut-il considérer lorsque client et serveur communiquent à travers plusieurs réseaux ?**

  **Logiquement le plus petit appelé qui définira le MTU de chemin**

- **Remarquons que ce MTU peut varier en fonction du chemin pris par les datagrammes IP à travers les routeurs d'*internet***

# *Taille maximum de segment TCP*



- **La taille maximum de segment (MSS : *Maximum Segment Size*) est le plus grand "morceau" de données que TCP enverra à l'autre extrémité**

- **Lors de l'établissement (SYN), chaque extrémité peut annoncer (champ option) son MSS qu'elle s'attend à recevoir; sinon une valeur par défaut de 536 est utilisée**

- **En général, un MSS le plus grand possible est souhaitable jusqu'à ce que la fragmentation apparaisse**

# *Débit utile maximum de TCP (1)*



- **Commençons par déterminer le débit utile maximum théorique sur un réseau *ethernet* de 10 Mbit/s :**

| Champ | Données | ACK |
|---|---|---|
| Préambule ethernet | 8 | 8 |
| En-tête ethernet | 14 | 14 |
| En-tête IP | 20 | 20 |
| En-tête TCP | 20 | 20 |
| Données utilisateur | 1460 | 0 |
| Bourrage | 0 | 6 |
| CRC ethernet | 4 | 4 |
| Interframe gap (9.6 µ s) | 12 | 12 |
| TOTAL | 1538 | 84 |

# *Débit utile maximum de TCP (2)*



- **Si la taille de fenêtre TCP exige un *ACK* par segment :**

  **débit utile = ( 1460 / (1538 + 84)) x 10 Mbit/s**

  **= 9,00 Mbit/s**

- **Par contre 44 segments de 1460 octets peuvent être envoyés si la fenêtre est ouverte à sa taille maximale de 65535 octets :**

  **débit utile = ( 44 x 1460 / ((44 x 1538) + 84 )) x 10**

  **= 9,48 Mbit/s**

- **Il s'agit d'une limite théorique qui fait l'hypothèse que les accusés de réception ne provoquent aucune collision**

# *Produit débit x délai (1)*



- **Ce produit définit le nombre d'octets émis dans l'intervalle nécessaire à son acquittement (contrôle de flux)**
- **Ainsi 3750 octets sont transmis sur un réseau ethernet de 10 Mbit/s pendant un temps aller - retour égal à 3 ms**
- **Une taille de fenêtre inférieure réduirait d'autant le débit utile entre producteur et consommateur**

# *Produit débit x délai (2)*



| Réseau | Débit binaire<br>bit/s | Temps Aller-retour<br>ms | Produit<br>octet |
|---|---|---|---|
| Ethernet | 10.000.000 | 3 | 3750 |
| T1 transcontinent | 1.544.000 | 60 | 11.580 |
| T1 satellite | 1.544.000 | 500 | 95.500 |
| T3 transcontinent | 45.000.000 | 60 | 337.500 |
| Gigabit transcontinent | 1.000.000.000 | 60 | 7.500.000 |