PROTOCOLE : DESCRIPTION ET EXEMPLES

La notion de protocole se retrouve à tous les niveaux du modèle en couche. Le sendmail par exemple s'appuie sur le protocole SMTP, les serveurs WWW sur le protocole HTTP. Pour obtenir leur description précise, il faut consulter les RFC (ftp://ftp.inria.fr/rfc).

1 - NOTION DE PROTOCOLE

Pour une couche donnée, on appelle *protocole*, l'ensemble des règles et des formats (sémantiques et syntaxiques) prédéfinis déterminant les caractéristiques de communication des processus de la couche. La mise en oeuvre d'un protocole est effectuée à partir d'un *PDU* (Protocol Data Unit).

Un *sous-système* dans une couche est un élément n'ayant des interactions qu'avec les éléments des biveaux immédiatement supérieur et inférieur. Il est composé d'une ou plusieurs *entités*. On peut donc considérer qu'une entité est un processus qui met en oeuvre un protocole particulier. On dit que le programme correspondant *implémente* le protocole.

Les *services* d'une couche sont les fonctionnalités offertes par cette couche à la couche supérieure. On y accède par des *primitives* spécifiques du service. Le *point d'accès à des services (SAP)* est le point où les services sont fournis par une entité de la couche aux sous-systèmes des couches supérieures ou inférieures. Les blocs de données utilisateurs (*SDU, Service Data Unit*) sont transmis par l'intermédiaire des SAP. Une couche offre en général plusieurs points d'accès.

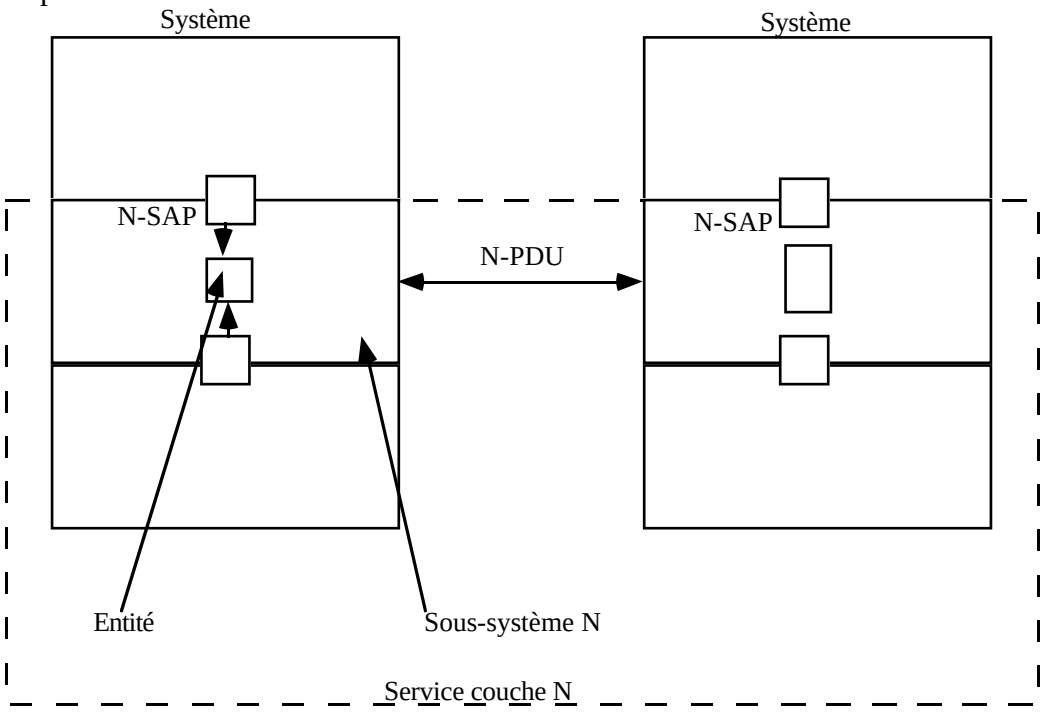

Figure 1. Détail des composants d'une couche.

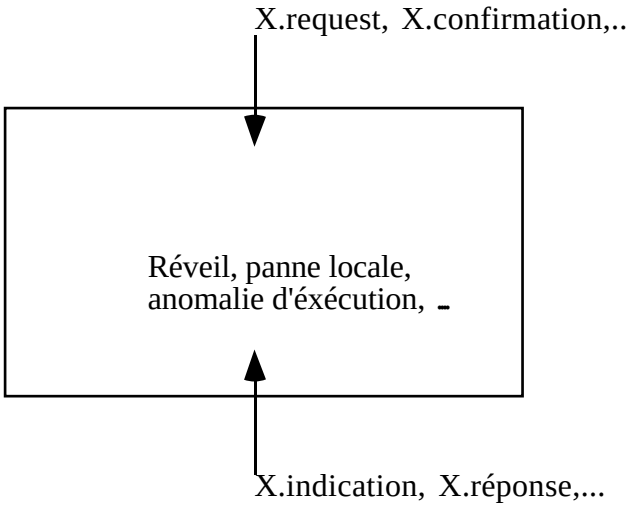

Figure 2. Evènements liés à un protocole.

Exemple. Pour fixer les idées, on peut tout à fait assimiler le fonctionnement à un service postal. Lorsque quelqu'un écrit à un chercheur dans un laboratoire, il envoie une ou plusieurs enveloppes (ce sont les PDU). Si il s'agit de plusieurs enveloppes à réordonner, il va les numéroter et rajouter donc une information inutile pour la poste mais utile pour le chercheur. Le message ainsi constitué est un SDU. Lorsque le facteur livre le courrier quelle que soit la personne du laboratoire, il la met dans la boite à lettre (N-1SAP) globale. Puis une personne (entité) va les distribuer dans les casiers (N-SAP).

Pour mettre en oeuvre correctement un protocole, il faut formaliser les actions et les objets utilisés ainsi que les interactions entre prestataires de services et utilisateurs. Une entité protocolaire (sous-système ou couche) est sollicitée par des évènements qui sont :

- externes : primitives d'interactions entre couches,
- internes, résultat de processus ou traitement.

La figure 2 donne une idée schématique des évènements liés à un protocole.

On décrit souvent un protocole par un automate d'état fini. Chaque transition est de la forme :

<état_départ, évènement, état_arrivée,
action_vers_couche_supérieure, action_vers_couche_inférieure, action_interne>.

Les trois derniers champs peuvent être vides.

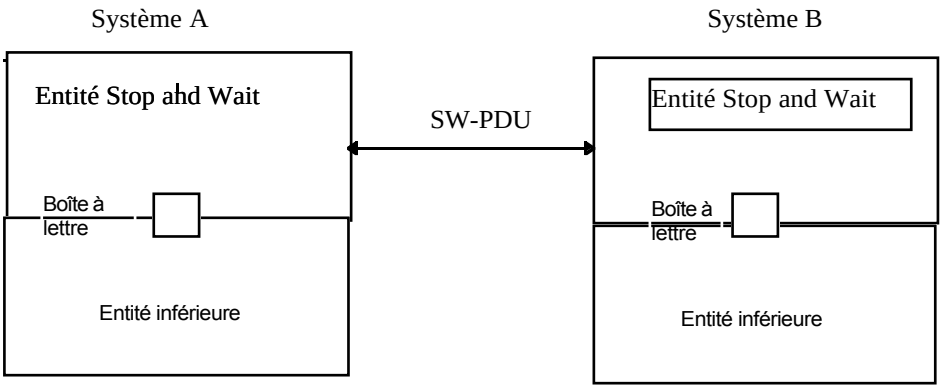

Figure 3. Architecture Send and Wait.

2 - EXEMPLE PROTOCOLE « SEND AND WAIT »

Le protocole « envoi et attente » est un des plus simple. Chaque entité peut émettre et recevoir des PDU par l'intermédiaire de sa boîte à lettres locale à son sous-système. Après l'envoi d'un PDU, l'entité se met en attente de réception d'un accusé de réception. La figure 3 montre l'architecture du système. Les primitives sont les suivantes :

- en provenance de la couche supérieure data.request,
- à destination de la couche supérieure data.indication,
- en provenance de la couche inférieure envoi.indication,
- à destination de la couche inférieure envoi.request.

L'automate correspondant est décrit figure 4.

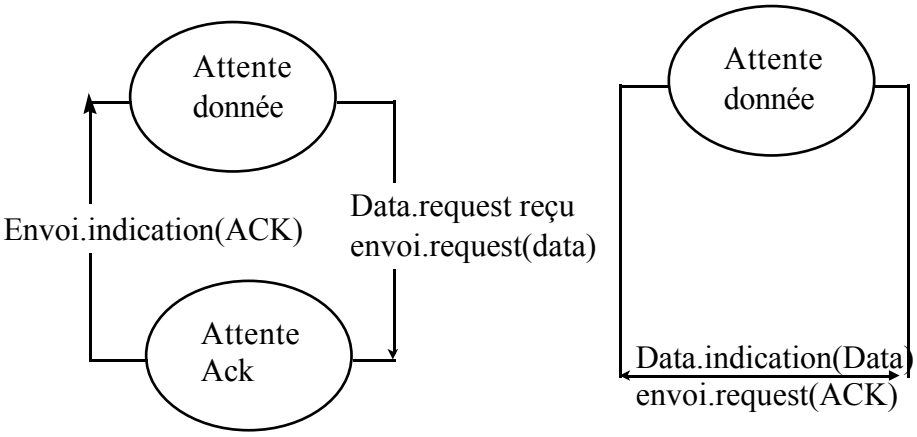

Figure 4. Automate send and wait.

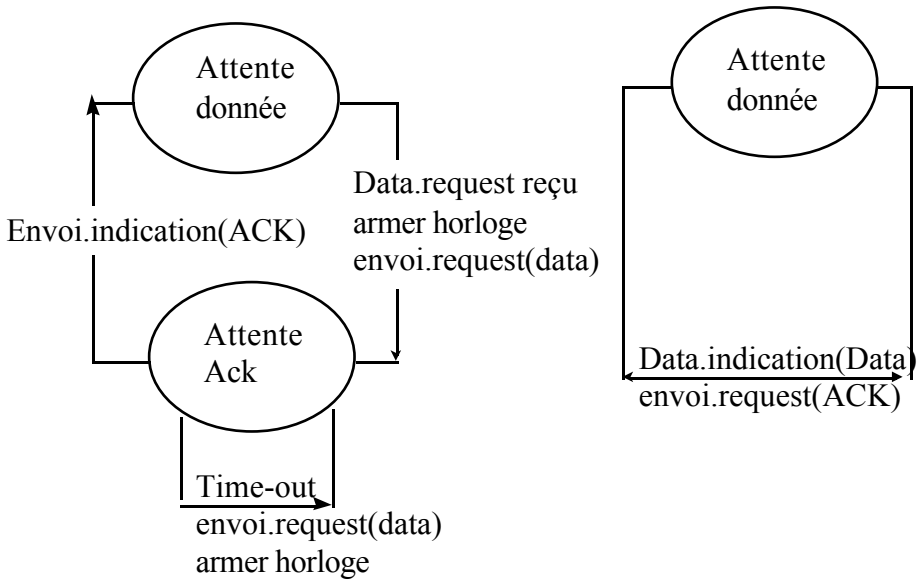

Figure 5. Send and Wait avec horloge.

Ce protocole simpliste ne prend pas en compte les événements extérieurs qui pourraient amener à un échec. Si l'on souhaite prévoir la possibilité de perte d'un PDU data ou ack, il faut ajouter au mécanisme, une horloge que l'émetteur d_clenchera chaque fois qu'il envoie un SDU data. L'envoyeur est alors en état d'attente de deux _v_nements :

- arrivée d'un ack,
- alarme de l'horloge.

Dans les deux cas il pourra par exemple réémettre le SDU.

Dans cet exemple, nous avons discuté des états et des échanges en fonction des événements définis par le protocole, il faut également prendre en compte le format des données à transmettre. Elles vont intervenir sur le comportement du récepteur.

3 - PROTOCOLE ICMP

Le protocole ICMP (Internet Control Message Protocol) permet l'échange des diagnostics d'erreurs entre éléments actifs du réseau. Bien que tournant au dessus de IP (couche 3), il est requis par toute machine fonctionnant sous IP. La plupart des paquets contiennent des informations sur les causes de destructions des paquets par exemple TTL expiré, destination inaccessible. Le format des données est extrêmement élaboré par contre il n'y a pas de dialogue entre deux entités ICMP. Tout paquet ICMP commence par un en-tête :

- type : 8 bits,
- code : 8 bits,
- checksum : 16 bits.

Les types de message sont les suivants :

0 Réponse d'écho
3 Destination inaccessible
4 Demande de ralentissement
5 Redirection
8 Echo
9 Annonce de routeur
10 Sollicitation de routeur
11 TTL expiré
12 Problème de paramètre
13 Horodatage
14 Réponse d'horodatage
15 Demande d'information
16 Réponse d'information

Les messages les plus fréquents sont ceux liés aux types Destination inaccessible, TTL expiré, et Demande de ralentissement. L'en-tête est alors suivie de 32 bits de remplissage ainsi que les premiers octets du paquet qui a déclenché le problème. Ces octets permettront au récepteur d'identifier l'application qui a émis le paquet. Par exemple dans le cas de Destination inaccessible le champ code précisera si c'est une erreur de type réseau, protocole, hôte, ...

Les types « écho » permettent de réaliser des fonctions de test du réseau par un utilisateur final. Ces tests sont implémentés par les commandes traceroute et ping.

Commande ping

Elle utilise les types 8 en émission et 0 en réception pour vérifier la qualité de la liaison avec un hôte. La réponse est fabriquée en permutant les adresses sources et destination dans le paquet IP. Le paquet ICMP de réponse dans son champ de données recopie le champ de données du paquet reçu. Ainsi on peut par exemple vérifier le séquencement des messages. De plus l'application ping mesure le temps d'aller-retour du message dans le réseau.

Exemple

```
[maylis] ~ > /usr/etc/ping -s www.cwi.nl
PING info4u.cwi.nl: 56 data bytes
64 bytes from info4u.cwi.nl (192.16.196.148): icmp_seq=0. time=116. ms
64 bytes from info4u.cwi.nl (192.16.196.148): icmp_seq=1. time=1237. ms
64 bytes from info4u.cwi.nl (192.16.196.148): icmp_seq=3. time=83. ms
64 bytes from info4u.cwi.nl (192.16.196.148): icmp_seq=4. time=100. ms
64 bytes from info4u.cwi.nl (192.16.196.148): icmp_seq=5. time=144. ms
64 bytes from info4u.cwi.nl (192.16.196.148): icmp_seq=7. time=82. ms
```

Commande traceroute

Elle utilise le type 11. Cette commande permet de vérifier les relais sur la route vers un hôte. Elle procède à l'envoi de paquets successifs en incrémentant à chaque fois le TTL du paquet. Le premier paquet est envoyé avec un TTL égal à 1. le premier équipement le recevra le décrémentera de 1 et comme il sera devenu nul, détruira le paquet puis émettra un paquet ICMP de type 11 en inversant les champs expéditeur et émetteur dans le paquet IP. Le second paquet initialisé à 2 franchira le premier équipement mais lors de la traversée du second équipement le TTL deviendra nul et provoquera de même l'émission d'un paquet ICMP de type 11,... ainsi de suite.

Exemple

```
[maylis] ~ > traceroute www.cwi.nl
traceroute to info4u.cwi.nl (192.16.196.148), 30 hops max, 40 byte packets
 1  iceman (147.210.8.154)  2 ms  2 ms  2 ms
 2  b3a1 (147.210.8.254)  2 ms  2 ms  3 ms
 3  b9a1.u-bordeaux.fr (147.210.254.253)  2 ms  2 ms  2 ms
……..
18  cwi-gw.cwi.nl (192.16.183.32)  94 ms  92 ms  85 ms
19  laxeer.cwi.nl (192.16.191.31)  85 ms  89 ms  87 ms
20  info4u.cwi.nl (192.16.196.148)  91 ms  88 ms  106 ms
```

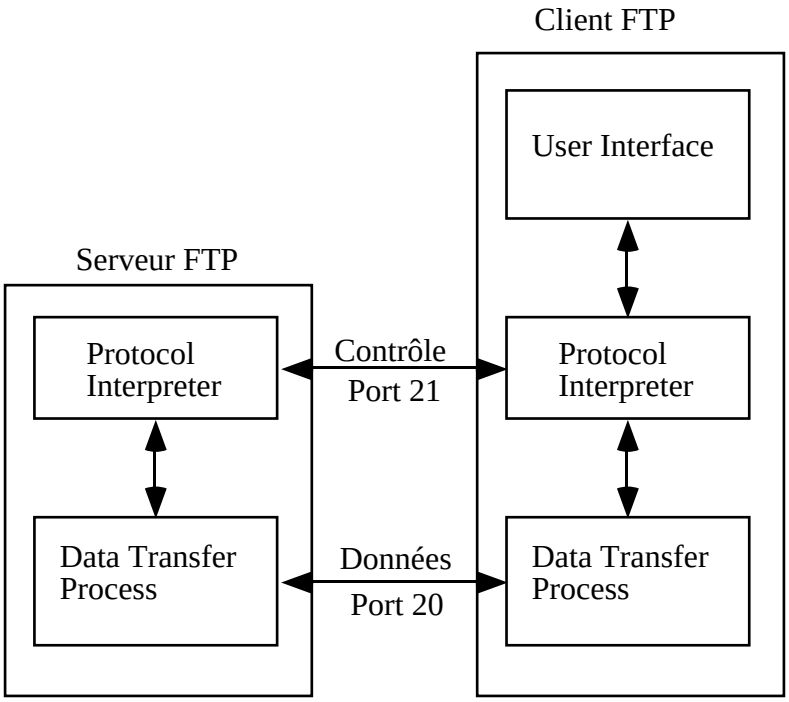

Figure 6. Relation serveur-client FTP.

4 - PROTOCOLE FTP

Le protocole FTP est un exemple de protocole alliant le dialogue à un codage très précis des données. Il est décrit précisément dans le RFC 959. Nous n'aborderons ici que les grandes lignes. Ce protocole utilise deux ports de connexions

- pour les commandes de contrôle port 21, interpréteur de protocole (PI),
- pour les données port 20, protocole de transfert de données (DTP).

Le PI client recoit les demandes en provenance de l'interface utilisateur, les transforme en commande et les transmet au PI serveur. Lors d'une commande de transfert de fichier le PI concerné transmet au DTP (dont il a la charge) pour ouvrir la connexion avec le DTP de l'autre partie. Les commandes transmises sont des chaînes de caractères de la forme

<mnémonique> <liste_argument> <CR><LF>.

La réponse est une chaîne de trois chiffres indiquant le statut après traitement (exemple 200 pour commande réussie). Il y a trois groupes de commandes :

- les commandes d'accès
  - USER : nom de login
  - PASS : mot de passe
  - ACCT : compte
  - CWD : répertoire de positionnement
  - CDUP : équivalent à cd ..
  - QUIT : fin de session

- les commandes de paramétrage

TYPE : ASCII (A), EBCDIC (E), Image (I), Local (L), ...

STRU : pour décrire la structure du fichier

MODE : flot continu de données, blocs ou compression

PORT : indique le port ou le DTP serveur doit se connecter (numero IP suivi de deux octets pour le port)

- les commandes de services

RETR : le DTP serveur envoit le fichier, il est copié sur la machine locale

STOR : le DTP serveur recoit un fichier, il est copié sur la machine distante

SYST :retourne le nom du système d'exploitation

LIST : équivalent à ls distant

HELP : liste des commandes implémentées dans l'UI

SITE : éxécution de commande sur le site distant

DEL : suppression de fichier distant

MKD : création de répertoire distant

RMD : suppression de réperoire distant

NOOP : demande d'accusé de réception

Chaque groupe de commande est décrit par un automate d'état fini. On trouvera figure 7, l'automate décrivant le protocole associé à la séquence login et figure 8, la correspondance avec les codes de signalisation.

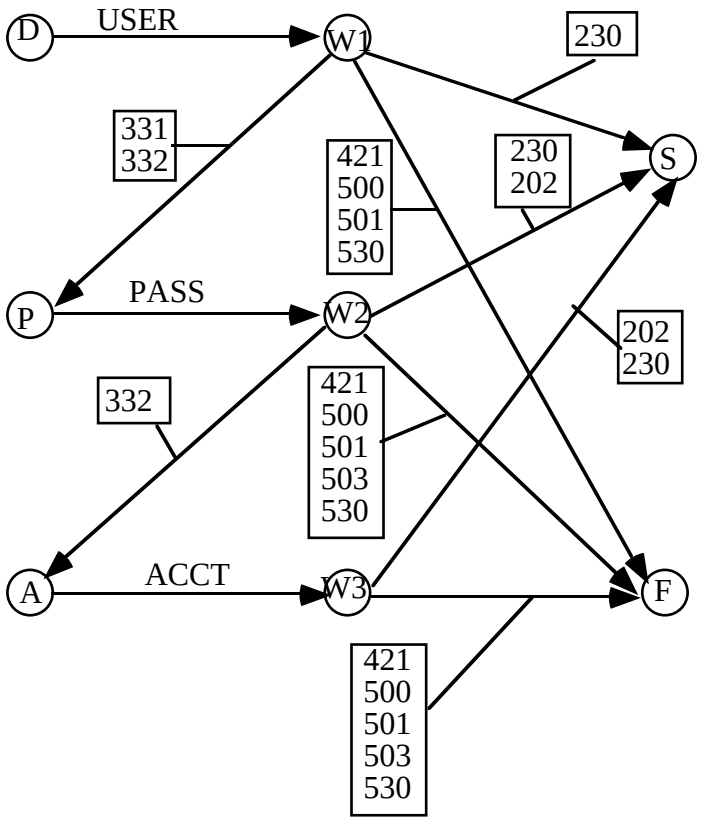

Figure 7. Protocole associé au login.

202	Command not implemented, superfluous at this site
230	User logged in, proceed
331	User name okay, need password
332	Need account for login
421	Service not available, closing control connection
500	Syntax error, command unrecognized
501	Syntax error in parameters or arguments
503	Bad sequence of commands
530	Not logged in

Figure 8. Codes de signalisation.

exemple

[maylis] ~ > ftp -dv firmin
Connected to firmin.
220 firmin FTP server (SunOS 4.1) ready.
Name (firmin:maylis):
---> USER maylis
331 Password required for maylis.
Password:
---> PASS <mon_mot_de_passe>
230 User maylis logged in.
ftp> get toto
---> PORT 147,210,8,138,7,0
200 PORT command successful.
---> RETR toto
150 ASCII data connection for toto (147.210.8.138,1792) (392 bytes).
226 ASCII Transfer complete.
local: toto remote: toto
405 bytes received in 0.036 seconds (11 Kbytes/s)
ftp> put toto
---> PORT 147,210,8,138,7,1
200 PORT command successful.
---> STOR toto
150 ASCII data connection for toto (147.210.8.138,1793).
226 ASCII Transfer complete.
local: toto remote: toto
405 bytes sent in 0.0018 seconds (2.2e+02 Kbytes/s)
ftp> quit
---> QUIT
221 Goodbye.