Problèmes et exercices

Exercice 1 Choix d'un service réseau

Énoncé

Un message de 40 octets doit être transmis entre deux équipements *A* et *B*. Supposons qu'on puisse connecter ces deux équipements à trois types de réseaux : (1) un réseau à commutation de circuits, (2) un réseau à commutation de paquets offrant un service orienté connexion, (3) un réseau à commutation de paquets offrant un service sans connexion. Quel type de réseau choisiriez-vous pour réaliser ce transfert de données ?

Solution

Sans autres précisions sur la nature du message et du degré de fiabilité souhaitée pour le transfert, il serait beaucoup plus simple d'utiliser un réseau à commutation de paquets, offrant un service sans connexion. Cela évite d'établir la connexion puis de la libérer après transfert (étapes obligatoires dans les réseaux en mode connecté) pour une si petite taille de message.

Exercice 2 Affectation des numéros de voie logique aux circuits virtuels

Énoncé

Deux entités *A* et *B* communiquent à travers un réseau de données utilisant des circuits virtuels. Pour communiquer, ils ont établi un circuit virtuel, identifié par le numéro de voie logique 152 du côté de *A*. Peut-on en déduire le numéro de voie logique utilisé par *B* ?

Solution

Non, il est impossible de déduire le numéro de voie logique utilisé du côté de *B* à partir du numéro employé par *A*. En effet, les numéros de voie logique sont définis localement, au niveau de l'interface entre l'équipement terminal et le nœud d'accès au réseau. Les deux numéros sont donc choisis indépendamment l'un de l'autre. Il n'existe aucune corrélation entre eux.

Remarque

Les numéros de voie logique ne sont pas choisis au hasard par les équipements. Si tel était le cas, un même numéro pourrait être attribué trop souvent à deux communications différentes, créant une *collision d'appels*. Pour éviter cela, l'équipement appelant (celui qui demande l'ouverture de connexion) choisit le plus grand numéro de voie logique disponible localement, alors que l'équipement appelé se verra affecté du plus petit numéro de voie logique disponible sur l'interface locale (en général à partir de 1, le numéro 0 étant réservé à l'administration du réseau). Le risque de collision d'appels est alors minimal.

Exercice 3 Ouverture de plusieurs circuits virtuels entre équipements terminaux

Énoncé

On considère un réseau à commutation de paquets offrant un service orienté connexion reliant plusieurs ETTD.

a Peut-on ouvrir simultanément plusieurs circuits virtuels entre deux ETTD ? Combien ? De quoi dépend ce nombre ?

b Si vous avez répondu par l'affirmative dans la précédente question, quel serait l'intérêt d'une telle solution ? Sinon, quel est l'inconvénient ?

Solution

a *A priori*, seul l'ETTD choisit avec qui il veut établir un circuit virtuel. Il peut naturellement en ouvrir plusieurs avec le même correspondant s'il le désire. Le nombre de circuits virtuels qu'il peut ouvrir simultanément dépend de deux facteurs : l'abonnement contracté et la taille du champ dévolu à l'identification du numéro de voie logique. Si celle-ci est, par exemple, 12 bits, on pourra identifier localement 4 096 circuits virtuels différents. Le nombre de circuits virtuels utilisables simultanément est un service facturé à l'utilisateur.

Remarque

La norme décrivant l'utilisation des circuits virtuels spécifie la taille du champ définissant le numéro de voie logique. Les réseaux publics utilisant les circuits virtuels s'appuient sur la recommandation *X.25* de l'ITU. Cette norme définit trois niveaux : le type d'interface série à utiliser, le protocole de liaisons de données à employer (généralement LAP-B, c'est-à-dire le mode équilibré de HDLC avec rejet simple des erreurs) et le mode d'acheminement des données (les paquets sont émis sur des circuits virtuels). Dans *X.25*, la taille du champ affecté aux numéros de voie logique est 12 bits.

b L'intérêt d'utiliser plusieurs circuits virtuels entre les deux mêmes ETTD est d'augmenter le flux des données entre eux, puisqu'à chaque circuit virtuel est associé un débit binaire maximal. Le débit binaire entre les deux ETTD est presque multiplié[2] par le nombre de circuits virtuels affectés à cette communication. Il faut toutefois disposer, aux deux extrémités, d'outils capables de réordonnancer les paquets provenant des différents circuits virtuels. Supposons qu'un ETTD utilise trois circuits virtuels différents pour communiquer avec le même ETTD distant. Il envoie ses données en utilisant un mécanisme d'« éclatement » des données sur les trois circuits virtuels (par exemple, en émettant le premier paquet sur le premier circuit virtuel, le deuxième paquet sur le deuxième circuit virtuel et ainsi de suite). L'ordre des paquets sera respecté au sein de chaque circuit virtuel mais ne correspondra pas forcément à l'ordre initial des paquets du message.

Remarque

Le débit binaire maximal disponible est lui aussi un paramètre négocié à l'abonnement. Il est complété par la notion de *classe de service*, qui garantit à l'utilisateur un débit binaire minimal pour la durée de la connexion. Par exemple, un abonné peut posséder un accès à 56 kbit/s et demander une classe de service de 9 600 bit/s. Cela signifie que, quel que soit l'encombrement du réseau, le débit binaire de ce circuit virtuel ne pourra être inférieur à cette valeur.

2. La transmission des paquets sur plusieurs circuits virtuels nécessite divers traitements au niveau des équipements terminaux. Le débit global est donc inférieur à la somme des débits des différents circuits virtuels utilisés pour le transfert des données.

EXERCICE 4 MULTIPLEXAGE DE CIRCUITS VIRTUELS SUR LA MÊME LIAISON DE DONNÉES

Énoncé

Un ETTD *A* établit 4 connexions avec 4 autres ETTD dénommés *B*, *C*, *D* et *E*. Il utilise pour cela un circuit virtuel par ETTD distant. Les numéros de voie logique affectés aux circuits virtuels valent respectivement 154, 153, 152 et 151. On suppose que *A* doit envoyer une information qui tient en 4 paquets à chacun des 4 destinataires.

a En supposant que les ETTD contactés n'aient aucune communication en cours, quel(s) numéro(s) de voie logique utiliseront-ils pour la communication avec l'ETTD *A* ?

b Si un paquet est contenu dans une seule trame I, combien de trames transportant les paquets à destination des ETTD distants seront émises par *A* ? Combien les autres ETTD recevront-ils de trames ?

c Comment s'effectue la répartition des paquets de données dans les différentes trames I au niveau de *A* ?

Solution

a Chaque ETTD distant n'ayant aucune communication en cours, il utilisera le numéro de voie logique 1 pour identifier le circuit virtuel avec l'ETTD *A*. Cela est dû au fait que l'attribution des numéros de voie logique est locale à l'interface ETTD – nœud d'accès au réseau.

b Il y a donc 16 trames I émises par l'ETTD *A*, chacune contenant un paquet à destination de l'un des ETTD distants. Par contre, chaque ETTD distant ne reçoit que 4 trames, contenant les 4 paquets qui lui sont destinés.

c Chaque ETTD distant reçoit ses paquets dans l'ordre où ils ont été émis par *A*. On ne peut pas en déduire pour autant l'ordre des trames I qui sont émises vers le nœud d'accès de *A*. En effet, chaque paquet à émettre est placé dans la file d'attente des paquets affectée au circuit virtuel concerné ; il y a autant de files d'attente que de circuits virtuels actifs.

Ensuite, l'entité qui gère la liaison de données prélève les données à émettre dans l'une des files d'attente, selon un ordre qui lui est propre (et qui dépend de la conception du logiciel). Ainsi, le gestionnaire de liaison peut décider d'entrelacer les données à destination des différents circuits virtuels (il envoie le premier paquet du premier circuit puis le premier paquet du deuxième circuit, et ainsi de suite), ou bien il peut envoyer tous les paquets d'un même circuit virtuel avant d'envoyer ceux du circuit virtuel suivant. Au moment d'émettre la trame I, le paquet remplit le champ de données de la trame.

Remarque

Cet exercice montre comment les entités du niveau immédiatement inférieur rendent les services à une entité de niveau supérieur. Il met aussi en évidence l'indépendance des entités des différents niveaux. En effet, le niveau gérant les paquets contient une ou plusieurs entités gérant les circuits virtuels, conformément au protocole de gestion des paquets. Chaque entité de ce niveau va soumettre des paquets à l'entité gérant la couche Liaison de données. Cette dernière est donc la seule à décider du mode d'envoi des données sur la liaison, en fonction de l'activité des différents circuits virtuels.

Exercice 5 Calcul du temps de transmission dans un réseau à commutation

Énoncé

Soit un réseau à commutation au sein duquel deux stations *A* et *B* ont établi une communication. *A* doit envoyer un fichier de taille *L* bits à *B*. Le transfert de données présente les caractéristiques suivantes :

- *S* est le nombre de commutateurs traversés pour la communication entre *A* et *B*.
- Toutes les liaisons de données utilisées ont un débit *D* bit/s.
- Le protocole de liaison est le même sur toutes les liaisons ; il ajoute un en-tête de *H* bits à chaque unité de données transférée.
- On néglige les temps de propagation et les temps de traitement dans les commutateurs du réseau. On néglige de même les temps de gestion des accusés de réception.

a Le réseau est un réseau à commutation de messages. Le fichier est transmis dans un seul message, d'une liaison à l'autre, jusqu'au destinataire. Donnez l'expression T_{fic1} du temps de transmission de ce fichier dans le réseau.

b Le réseau est un réseau à commutation de paquets. Le fichier est découpé en paquets contenant *P* bits de données (pour simplifier, on supposera que les paquets sont tous de taille identique). Montrez que l'expression T_{fic2} du temps de transmission du fichier est : $T_{fic2} = (S + L/P)(P + H)/D$.

c Calculez et comparez les temps obtenus dans les deux premières questions en prenant : $L = 64\,000$ octets ; $H = 9$ octets ; $S = 2$; $D = 64$ kbit/s. On prendra trois valeurs possibles pour la taille des paquets : $P = 128$ octets ; $P = 16$ octets ; $P = 48$ octets (dans ce dernier cas, il s'agit d'une cellule ATM dont l'en-tête *H* utilise 5 octets).

d Quels sont les avantages et les inconvénients de la commutation de paquets par rapport à la commutation de messages ?

e Les liaisons sont affectées d'un taux d'erreurs noté τ. Montrez que la probabilité *p* pour qu'une trame de longueur *L* soit reçue correctement vaut $p = (1 - \tau)^L$. En déduire que le nombre moyen *N* de transmissions d'une trame vaut : $N = 1/p$. Pour obtenir ce résultat, on supposera que le protocole de liaison répète indéfiniment la même trame sans anticipation, jusqu'à ce que la trame soit correctement reçue.

f Refaire l'application numérique de la question c en prenant un taux d'erreurs $\tau = 10^{-4}$. Pour ces calculs, on considère qu'une seule trame est émise dans le réseau à commutation de messages. Dans la commutation de paquets, chaque paquet est transmis dans une trame.

g Comparez les résultats et concluez. Ces techniques sont-elles adaptées aux hauts débits ?

Solution

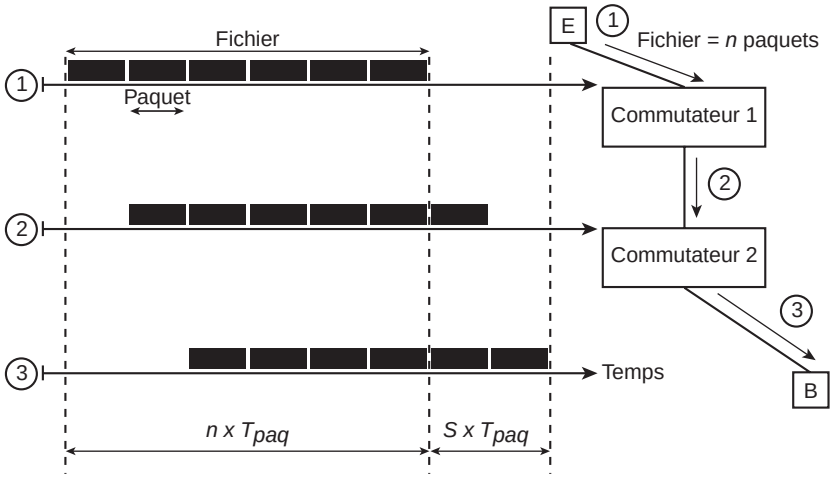
a La durée de transmission du fichier sur une liaison est égale à : $T_{fic} = (L + H)/D$. La durée de transmission du fichier est égale au temps de transmission sur toutes les liaisons traversées, c'est-à-dire : $T_{fic1} = T_{fic}*(S + 1)$.

b La durée de transmission d'un paquet sur une liaison de données vaut : $T_{paq} = (P + H)/D$. La durée de transmission du fichier est égale à la durée de transmission des paquets jusqu'au premier commutateur, plus le délai nécessaire au dernier paquet pour parvenir jusqu'à *B*. Le nombre de paquets nécessaires pour transmettre le fichier vaut $n = L/P$. On en déduit : $T_{fic2} = (S + n)*T_{paq} = (S + n)*(P + H)/D$.

La figure 3.18 montre comment calculer les différents temps de transmission.

Figure 3.18

Calcul des différents temps de transmission.



c Applications numériques :

a. Cas de la commutation de messages : $P = L = 64\,000*8 = 512\,000$ bits

$$T_{fic1} = (2 + 1)*(64\,000 + 9)*8/64\,000 = 24\ s.$$

b. Cas de la commutation de paquets avec $P = 128$ octets = 128*8 = 1 024 bits ; $n = L/P = 500$ paquets

$$T_{fic2} = (2 + 500)*(128 + 9)*8/64\,000 = 8{,}6\ s.$$

c. Cas de la commutation de paquets avec $P = 16$ octets = 16*8 = 128 bits ; $n = L/P = 4\,000$ paquets

$$T_{fic2} = (2 + 4\,000)*(16 + 9)*8/64\,000 = 12{,}5\ s.$$

d. Cas de la commutation de cellules ATM avec $P = 48$ octets = 48*8 = 384 bits ; $H = 5$ octets ; $n = L/P = 1\,334$ paquets (par excès)

$$T_{fic2} = (2 + 1\,334)*(48 + 5)*8/64\,000 = 8{,}85\ s.$$

Remarque

Pour la commutation de messages, le temps de transmission du fichier ne dépend que du nombre de liaisons traversées. En revanche, pour la commutation de paquets, il faut tenir compte du recouvrement des temps de transmission des différents paquets sur l'ensemble des liaisons : en effet, pendant que *A* transmet son deuxième paquet au premier commutateur, celui-ci envoie le premier paquet au commutateur suivant et ainsi de suite. C'est la raison pour laquelle les performances des réseaux à commutation de paquets sont supérieures à celles des réseaux à commutation de messages. L'écart des performances sera encore plus notable si certaines liaisons transmettent le message avec des erreurs, comme nous le verrons aux questions suivantes.

d Nous voyons bien que le découpage en paquets permet de réduire les délais d'acheminement à travers le réseau. Cependant, il faut respecter une juste proportion entre la taille de l'en-tête par rapport au corps du message : une taille de paquet trop petite provoque un allongement de délai d'acheminement.

e Pour qu'une trame de longueur L soit reçue sans erreur, il faut que tous ses bits soient reçus sans erreur. La probabilité de recevoir un bit sans erreur vaut $1 - \tau$. La probabilité de recevoir L bits sans erreur vaut : $(1 - \tau)^L$. La probabilité de recevoir une trame erronée est de $p_t = 1 - (1 - \tau)^L$.

Puisque la longueur d'une trame vaut $L = P + H$, le nombre moyen d'émissions est donc :

$$1*(1 - p_t) + 2*(1 - p_t)\, p_t + 3*(1 - p_t)\, p_t^2 + \ldots = 1/(1 - p_t).$$

En appliquant la formule précédente en tenant compte des répétitions, on obtient :

$$T'_{fic} = T_{fic}/(1 - p_t) = T_{fic}/1 - (1 - \tau)^L.$$

f Les applications numériques donnent :

a. Cas de la commutation de messages : $P = L = 64\,000*8 = 512\,000$ bits

$T'_{fic} = 16\,848$ s soit plus de 4 heures !

b. Cas de la commutation de paquets avec $P = 128$ octets = 128*8 = 1 024 bits

$T'_{fic} = 9{,}6$ *s,* soit une dégradation de 11,6 % par rapport au cas parfait.

c. Cas de la commutation de paquets avec $P = 16$ octets = 16*8 = 128 bits ; $n = L/P = 4\,000$ paquets

$T'_{fic} = 12{,}75$ *s*, soit une dégradation de 2 % par rapport au cas parfait.

$T'_{fic} = 9{,}22$ *s*, soit une dégradation de 4,2 % par rapport au cas parfait.

g La prise en compte du taux d'erreurs dans les liaisons montre tout l'intérêt du découpage des messages en paquets. Il est visiblement hors de question d'utiliser la commutation de messages pour les applications nécessitant les hauts débits, tout particulièrement lorsque les liaisons sont peu fiables. Nous voyons également qu'une taille de paquet trop petite est un choix peu judicieux. Les cellules ATM et les paquets de 128 octets sont donc des compromis intéressants entre les différentes contraintes pour les hauts débits.

EXERCICE 6 TRANSFERT DE DONNÉES SUR CIRCUIT VIRTUEL

Énoncé

Après avoir fait vos courses dans l'hypermarché proche de votre domicile, vous vous présentez à une caisse pour régler vos achats. Vous décidez de payer avec la carte du magasin. Examinons le déroulement des opérations entre votre caisse et le centre de traitement informatique de la chaîne de magasins *via* Transpac, le réseau public de données français utilisant le protocole *X.25*.

La caisse *C* de l'hypermarché possède un accès avec les caractéristiques suivantes : le débit binaire est de 48 000 bit/s ; le niveau Liaison de données utilise le mode équilibré du protocole HDLC (protocole LAP-B) avec une fenêtre d'anticipation de 4. La gestion des paquets de données est faite sur un CVC (circuit virtuel commuté), ouvert par la caisse dès sa mise en service (le CVC reste ouvert jusqu'à la fermeture du magasin : inutile de le fermer après chaque opération de la caisse). La taille de la fenêtre d'anticipation des paquets est 1.

Le serveur *S* de la chaîne d'hypermarchés possède un accès configuré comme suit : le débit binaire est de 2,048 Mbit/s ; le niveau Liaison de données utilise le même protocole LAP-B avec une fenêtre d'anticipation de 7 ; le CVC possède une fenêtre d'anticipation des paquets de 1.

☞

Énoncé *(suite)*

Après avoir saisi le prix de tous les articles que vous avez achetés, la caissière appuie sur une touche qui provoque l'envoi d'une requête *Demande de paiement par carte magasin*. Cette requête parvient au centre de traitement de la chaîne de magasins et active l'application de facturation. Celle-ci traite la transaction et renvoie une *Autorisation de paiement par carte magasin*. La caisse affiche alors sur l'écran un message vous invitant à taper le chiffre qui précise la nature de votre paiement (à comptant ou à crédit), puis les chiffres de votre code secret. Vous pianotez votre choix de paiement puis votre code sur le pavé numérique. La caissière vérifie que la saisie est correcte et valide les données fournies en appuyant sur une touche. La caisse transmet les données que vous avez tapées dans un paquet *Données du paiement EFT*[3], puis elle reçoit un paquet *Fin de transfert EFT*, qui provoque l'affichage du message d'accueil du client suivant sur l'écran de la caisse.

a En supposant que le plus grand numéro de voie logique disponible soit 9 et le plus petit soit égal à 1 pour les deux ETTD, décrivez l'échange des paquets nécessaires à l'ouverture du CVC par la caisse, en respectant les conventions d'attribution des numéros de voie logique. Pour ouvrir le CVC et transférer les données, vous disposez des types de paquets donnés au tableau 3.1.

b En supposant que chaque message ou requête tienne dans un seul paquet de données, décrivez le transfert des paquets de données entre la caisse et son ETCD (le nœud d'accès au réseau) d'une part, et entre le centre de traitement et son ETCD d'autre part. Vous veillerez à respecter la chronologie des événements (par exemple, un paquet émis par le serveur en réponse à un paquet de la caisse doit être émis *après* que le paquet de la caisse est parvenu au serveur…).

c Pourquoi le serveur *S* utiliserait-il le numéro de voie logique 5 pour identifier la connexion avec la caisse *C* ?

d Indiquez comment s'effectue la fermeture du CVC par la caisse à la fin de la journée.

e Si des parasites sur les liaisons de données avaient produit des erreurs de transmission, quel niveau les détecterait ?

Tableau 3.1 : Les différents types de paquets utilisés sur un circuit virtuel

Type du paquet	Fonction	Paramètres	Mnémonique utilisé
Ouverture de connexion	Ouvre le circuit virtuel	O, D, n° VL	APPEL (O, D, n° VL) (a)
Indication d'ouverture	Alerte l'ETTD distant	O, D, n° VL	AP_ENT (O, D, n° VL) (b)
Acceptation d'ouverture	Accepte la communication	O, D, n° VL	COM_ACC (O, D, n° VL) (c)
Indication d'acceptation	Indique à l'ETTD appelant que l'appelé accepte la connexion	O, D, n° VL	COM_ETA (O, D, n° VL) (d)
Demande de libération	Un ETTD demande la fin du transfert de données	O, D, n° VL	DEM_LIB (O, D, n° VL) (e)

☞

3. EFT signifie *Electronic Fund Transfer* (transfert de fonds électronique).

Énoncé *(suite)*

Tableau 3.1 : Les différents types de paquets utilisés sur un circuit virtuel (suite)

Type du paquet	Fonction	Paramètres	Mnémonique utilisé
Indication de libération	Indique une demande de fin de transfert de l'autre ETTD	O, D, n° VL	IND_LIB (O, D, n° VL) (f)
Confirmation de libération	Confirme la fin du transfert de données	O, D, n° VL	LIB_CONF (O, D, n° VL)
Acquittement des données	Acquitte les paquets de données jusqu'à *P(R) – 1*	X	PRR(X) (g)
Paquet de données	Numérote le paquet émis et acquitte les paquets reçus par piggy-backing	X,Y, Z	DXY(Z) (h)

Au tableau 3.1, nous utilisons les notations suivantes :

(a) *O* est l'adresse complète de l'abonné origine ; *D* l'adresse complète de l'abonné distant ; *n° VL* est le numéro de voie logique localement utilisé pour identifier le circuit virtuel à ouvrir.

(b) *AP_ENT* est la notation abrégée utilisée pour un appel entrant. Ce paquet est émis par le nœud de l'ETTD appelé pour l'avertir d'une demande de connexion par un ETTD distant.

(c) *COM_ACC* signifie que l'ETTD distant accepte la demande de connexion. Ce paquet est émis par l'ETTD appelé vers son ETCD.

(d) *COM_ETA* sert à indiquer à l'ETTD appelant que l'ETTD distant a accepté la communication. Ce paquet est émis par le nœud de l'ETTD origine.

(e) *DEM_LIB* signifie que l'ETTD qui l'émet demande la fin du transfert de données.

(f) *LIB_CONF* est la confirmation de la libération demandée soit par le réseau (en cas de panne ou d'impossibilité de traiter l'appel), soit par l'ETTD distant.

(g) *PRR(X)* est le paquet d'acquittement des paquets reçus en séquence jusqu'au paquet de numéro *P(R)* – 1.

(h) *DXY(Z)* est un paquet de données dans lequel *X* représente le numéro d'ordre *P(S)*, *Y* est l'acquittement des paquets jusqu'à *P(R) – 1* et *Z* le numéro de voie logique utilisé.

Solution **a** La figure 3.19 montre l'échange des paquets entre les ETTD et les ETCD pour ouvrir le CVC. *C* utilise le plus grand numéro de voie logique et *S* le plus petit disponible sur l'interface locale.

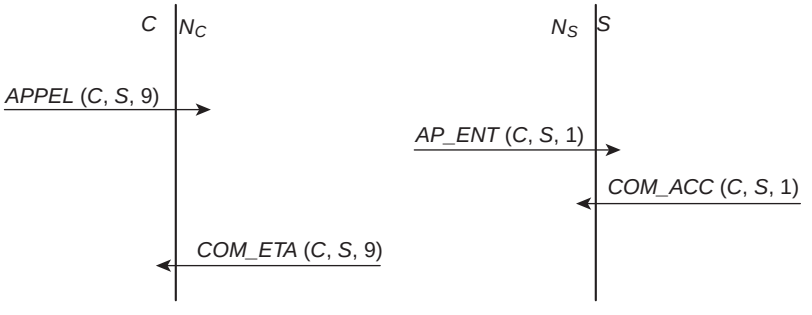
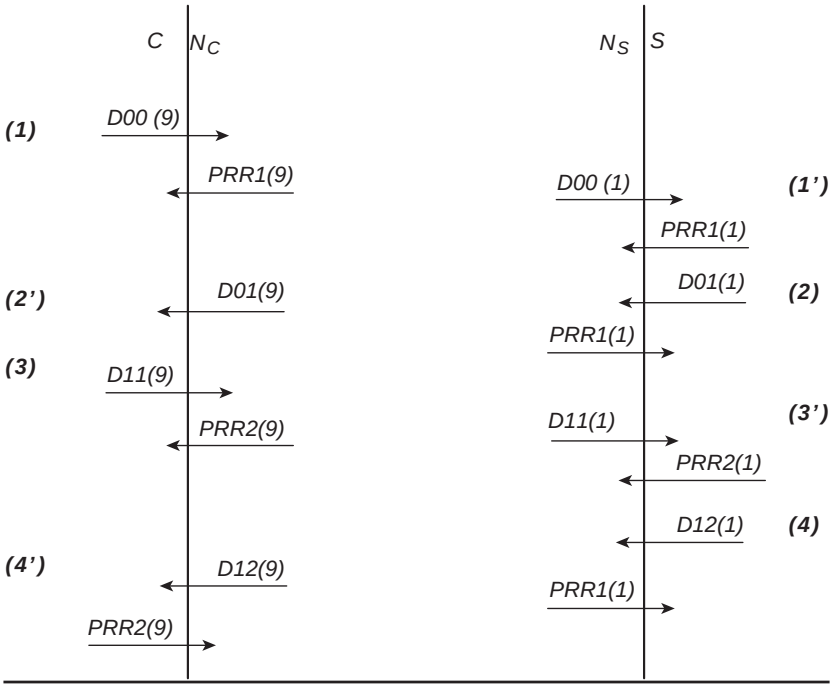


Figure 3.19
Ouverture du CVC par la caisse.

b La figure 3.20 nous montre les échanges de paquets entre ETCD et ETTD pour assurer le transfert de données demandé.

Figure 3.20

Transfert de données entre la caisse et le serveur.



(1) : Paquet contenant « Demande de paiement par carte magasin ».

(1') : Le paquet contenant « Demande de paiement par carte magasin » est arrivé dans le nœud du serveur. Il est envoyé au serveur et va servir à lancer la transaction de paiement.

(2) : Paquet contenant « Autorisation de paiement par carte magasin », généré par l'application de facturation, après vérification des données du client. Ce paquet est envoyé au nœud du serveur.

(2') : Le paquet contenant « Autorisation de paiement par carte magasin » est arrivé au nœud de la caisse, qui le transmet à la caisse. Il va servir à afficher le message vous invitant à taper votre code secret.

(3) : Paquet contenant « Données du paiement EFT », contenant vos données client.

(3') : Le paquet contenant « Données du paiement EFT » est arrivé au nœud du serveur, qui le retransmet au serveur. Il va permettre de mettre à jour votre compte client dans le serveur.

(4) : Paquet contenant « Fin de transfert EFT » pour demander la fin de la transaction.

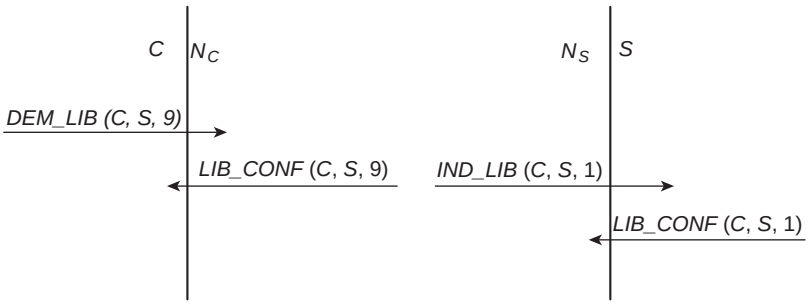
(4') : Le paquet contenant « Fin de transfert EFT » arrive au nœud de la caisse. Ce dernier le retransmet à la caisse. Il va permettre d'afficher le message d'accueil du client suivant.

c Dans la réponse à la question précédente, nous supposons implicitement que le serveur n'a pas d'autres circuits virtuels ouverts quand il reçoit la demande de connexion de la caisse *C*. S'il utilise maintenant le numéro de voie logique 5, c'est tout simplement parce qu'il a déjà utilisé les quatre premiers numéros de voie logique pour d'autres connexions.

d La figure 3.21 montre la fermeture du CVC par la caisse.

Figure 3.21

Fermeture du CVC par la caisse.



Remarque

Nous avons vu à la question a que l'ouverture d'un circuit virtuel s'effectue *de bout en bout* : seul l'ETTD appelé peut répondre à la demande d'ouverture de connexion envoyé par l'ETTD appelant. À la question d, nous voyons que la libération du circuit virtuel s'effectue *localement* : c'est le nœud auquel est raccordé l'ETTD qui confirme la libération de connexion et non l'ETTD distant. La libération d'un circuit virtuel s'effectue donc beaucoup plus rapidement que son ouverture.

e La détection des erreurs ne peut se faire au niveau de l'entité qui gère le circuit virtuel puisque le protocole de gestion du circuit virtuel ne contient aucun mécanisme de détection des erreurs. Elle est assurée au niveau du protocole gérant la liaison de données, grâce au FCS placé à la fin de la trame. Si la transmission altère une trame (et donc le paquet qui est dans son champ de données), celle-ci est rejetée car le récepteur détecte l'erreur dans la trame et demande sa retransmission. Tout paquet se trouvant dans une trame erronée est ignoré du récepteur.

EXERCICE 7 TRANSFERT DE DONNÉES SUR PLUSIEURS CIRCUITS VIRTUELS

Énoncé

Soit un réseau utilisant le protocole *X.25*, dans lequel un serveur *C* initialise les deux circuits virtuels qui le relient à *A* et *N*. Le premier, ouvert entre *C* et *A*, utilise une fenêtre de 1 alors que celui entre *N* et *C*, ouvert en second, utilise une fenêtre de 2. Le transfert des données se déroule comme suit : *A* et *N* envoient au même instant deux paquets vers le serveur *C*. Lorsqu'il a reçu tous les paquets des ETTD, *C* leur répond en envoyant à chacun deux paquets, *en entrelaçant les paquets des 2 circuits virtuels* (c'est-à-dire *C* envoie le premier paquet vers *A* puis le premier paquet vers *N* puis le second paquet vers *A* et enfin le second paquet vers *C*). On s'intéresse au transfert des paquets de données entre les stations *A*, *N* et le serveur *C*.

Protocole réseau : le transfert des données s'effectue sur des CVP (circuits virtuels permanents[4]), ouverts par le centre serveur *C*, au moment de la mise en service du réseau.

Les fenêtres de chaque CVP sont indiquées ultérieurement. La figure 3.22 montre deux stations clientes (*A* et *N*) communiquant avec un serveur *C*. Les points N_A, N_C, N_N constituent les points d'accès au réseau *X.25* (ce sont les ETCD locaux respectivement raccordés aux ETTD à *A*, *C* et *N*).

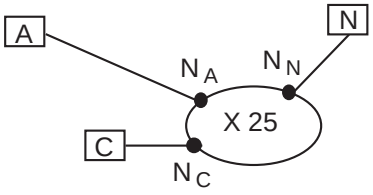
☞

4. Un CVP est un circuit virtuel établi une fois pour toutes, entre deux ETTD, jusqu'à la fin de l'abonnement contracté. On n'a pas besoin de l'établir, ni de le libérer.

Énoncé *(suite)*

Figure 3.22

Connexions entre ETTD et ETCD du réseau.



a Quels sont les numéros de voie logique utilisés par les différents ETTD, sachant que le plus grand numéro de voie logique disponible sur tous les ETTD est égal à 10 et qu'aucun ETTD n'a de circuit virtuel déjà ouvert ?

b Représentez, sur le même diagramme, les échanges de paquets correspondant au transfert de données entre les trois stations et répondant aux hypothèses décrites précédemment. Vous supposerez pour cela que les circuits virtuels sont déjà ouverts avant le transfert de données. Le circuit virtuel entre *A* et *C* utilise une fenêtre de 1, celui entre *N* et *C* une fenêtre de 2.

c Sur un circuit virtuel, les données peuvent aussi être acquittées de bout en bout, c'est-à-dire que l'ETTD destinataire est le seul habilité à acquitter les données reçues. Déterminez en quoi cela peut affecter la façon de transférer les données et les performances attendues.

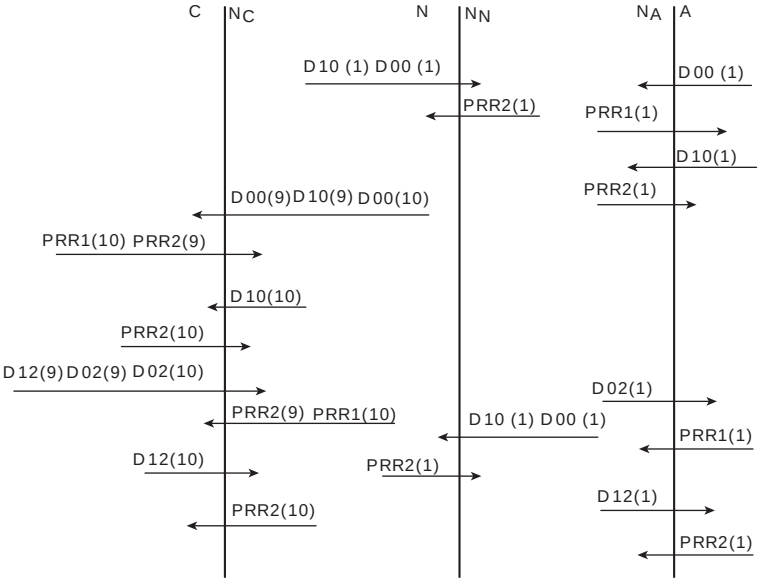
Solution

a D'après les conventions d'affectation des numéros de voie logique, le circuit virtuel entre *A* et *C* utilise le numéro de voie logique 10 dans l'ETTD *C*. Le numéro 9 identifie le circuit virtuel entre *C* et *N*. Les ETTD *A* et *N* utilisent tous les deux, pour identifier le circuit virtuel avec *C*, le numéro de voie logique 1.

b La figure 3.23 décrit les échanges entre les trois ETTD.

Figure 3.23

Transfert de données entre les trois ETTD.



c Avec des *acquittements locaux*, les paquets de données émis par l'ETTD sont acquittés par le nœud d'accès. Cette technique permet d'émettre les données le plus vite possible mais, en cas de panne, il y a un risque d'en perdre : dès qu'un paquet est acquitté, l'ETTD émetteur efface les données qui viennent d'être envoyées pour réutiliser la zone mémoire alors que le paquet n'est pas encore parvenu au destinataire ! (Comme pour l'envoi d'une

lettre par la Poste : ce n'est pas parce qu'on vient de la mettre dans la boîte qu'elle se trouve déjà dans les mains du destinataire. Elle peut se perdre ou être détruite avant d'arriver, même si cette éventualité est rare.)

Avec des *acquittements de bout en bout*, les nœuds intermédiaires se contentent de transporter le paquet puis de renvoyer l'acquittement qui est émis par le récepteur distant. En le recevant, l'émetteur est sûr que le paquet est bien parvenu à destination.

L'acquittement de bout en bout est la méthode d'acquittement la plus fiable mais aussi la plus lente. Après l'envoi du paquet dans le réseau, il faut attendre que l'acquittement du destinataire ait parcouru tout le réseau en sens inverse jusqu'à l'expéditeur. Au mieux, le transfert des données avec acquittement de bout en bout pour chaque paquet est deux fois plus lent qu'un transfert n'utilisant que des acquittements locaux. Si nous conservons notre exemple de courrier postal, cette méthode d'acquittement correspond à l'envoi d'une lettre recommandée avec accusé de réception.

Remarque

Pour éviter l'engorgement du réseau et des cadences de transfert trop lentes, il est recommandé de ne pas utiliser uniquement l'acquittement de bout en bout mais de le réserver à des points de reprise critiques du fichier. L'exercice 8 montre comment panacher acquittements locaux et acquittements de bout en bout.

Pour illustrer ce principe, supposons qu'on veuille transférer le contenu de ce livre. Sachant que chaque page génère plusieurs paquets de données, plusieurs manières de transférer le fichier peuvent s'envisager.

Chaque paquet est acquitté de bout en bout : cette méthode, longue et fastidieuse, n'est pas à conseiller, même si elle est très fiable !

Chaque page est acquittée de bout en bout : seul le dernier paquet de la page est acquitté de bout en bout. En recevant l'acquittement correspondant, l'émetteur sait que toute la page a été bien reçue, puisqu'un acquittement valide également les paquets qui le précèdent. Cette méthode est intéressante si le réseau présente une fiabilité insuffisante en regard des besoins de l'application de transfert de fichiers.

Chaque chapitre est acquitté de bout en bout : la méthode est la même pour le dernier paquet du chapitre au lieu du dernier paquet de la page. Cette méthode est certainement la plus sûre et la plus rapide si le réseau est suffisamment fiable

Exercice 8 Transfert de données sur des circuits virtuels avec acquittements locaux et acquittements de bout en bout

Énoncé

Un serveur *S* communique avec une station *A*. Il utilise pour cela un circuit virtuel *déjà ouvert*, identifié par les numéros de voie logique 7 du côté de *S*, 1 du côté de *A* ; la fenêtre du circuit virtuel vaut 2 dans les deux sens.

A envoie 4 paquets de données vers le serveur, qui répond par 2 paquets de données. L'acquittement des paquets de données est *local* pour les premiers paquets de données, *de bout en bout* pour le dernier paquet transmis par chaque ETTD.

a Représentez, à l'aide d'un diagramme, les échanges de paquets correspondant à la transaction effectuée entre les stations *A* et *S*. Les paquets transmis de bout en bout seront distingués des paquets acquittés localement par un astérisque sur le diagramme.

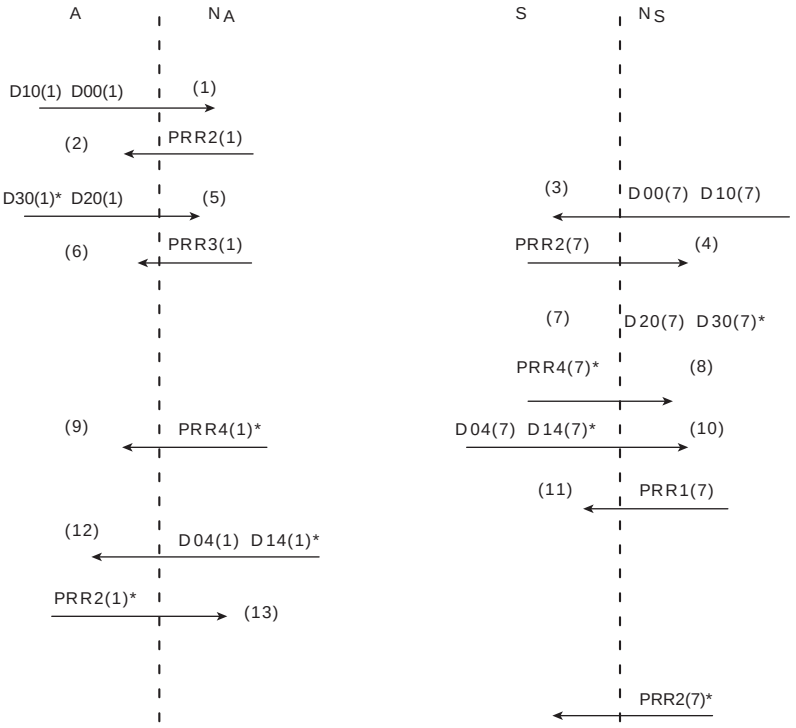
b Combien *A* peut-il encore envoyer de paquets, une fois qu'il a envoyé son dernier paquet de données ?

Solution

a La figure 3.24 montre le transfert des données entre *A* et *S*.

Figure 3.24

Transfert de données avec acquittements locaux et de bout en bout.



(1) : A envoie les paquets correspondant à la fenêtre d'anticipation.

(2) : L'ETCD les acquitte localement.

(3) : Après avoir traversé le réseau, les deux premiers paquets de données sont transmis à S.

(4) : S acquitte les paquets reçus.

(5) : A envoie ses 2 derniers paquets et demande l'acquittement de bout en bout du dernier paquet.

(6) : L'ETCD n'acquitte que le troisième paquet.

(7) : Les derniers paquets de A sont transmis à S.

(8) : S acquitte tous les paquets et demande la propagation de bout en bout de son acquittement.

(9) : L'acquittement des paquets a traversé tout le réseau et parvient à A.

(10) : S envoie 2 paquets et demande l'acquittement de bout en bout du dernier paquet, tout en validant les paquets reçus précédemment.

(11) : Seul le premier paquet de S peut être acquitté par l'ETCD.

(12) : Les paquets sont transmis à A ; le dernier doit être acquitté de bout en bout.

(13) : A acquitte les paquets et demande la propagation de bout en bout de son acquittement.

(14) : L'acquittement de bout en bout parvient à S.

b Après avoir envoyé son dernier paquet, *A* ne peut pas envoyer plus d'un paquet, car la réception de l'acquittement local *PRR3(1)* lui indique que seuls 3 paquets sont acquittés. La fenêtre n'est complètement ouverte qu'après réception de l'acquittement de bout en bout provenant de *S*. La situation est identique pour le serveur *S* qui n'a reçu que l'acquittement (local) du premier paquet.

Remarque

La demande d'acquittement de bout en bout revient au fait que l'ETTD distant commande l'ouverture de la fenêtre de l'ETTD local.

Chapitre 4

Les architectures de communication

Ce chapitre décrit comment se formalisent les notions d'empilement de couches de protocoles et de services, que nous avons rapidement évoquées au cours du chapitre précédent. Nous abordons également les modifications qu'il a fallu apporter au modèle initial pour décrire la structure des réseaux locaux (LAN), aujourd'hui les plus utilisés dans les entreprises ou même chez les particuliers. Nous introduisons ensuite l'architecture de communication utilisée dans Internet, connue sous le nom d'*architecture TCP/IP* ou encore de *pile TCP*. Enfin, nous évoquons comment normes et standards sont élaborés par les organismes internationaux et le comité technique chargé de l'évolution d'Internet.

1 Concept d'architecture en couches

L'empilement des couches et les services qu'elles offrent constituent l'*architecture de communication*. Une architecture de communication est donc une représentation abstraite (indépendante de toute référence à des logiciels ou des matériels particuliers) de la circulation des informations et des concepts utilisés au sein d'un réseau quelconque. Pour cela, nous nous appuyons sur le modèle abstrait d'architecture de communication défini dans le milieu des années 1970 pour les réseaux grande distance (WAN). Cette architecture, communément appelée *modèle de référence OSI* ou *modèle OSI*, a apporté un vocabulaire et des concepts encore utilisés de nos jours. En effet, même des architectures élaborées en dehors de ce modèle se réfèrent à la terminologie qui y est définie.

1.1 Pourquoi utiliser une architecture en couches ?

La structuration en couches considère un système comme logiquement composé d'un ensemble de *n* sous-systèmes ordonnés. Les sous-systèmes adjacents communiquent à travers leur interface commune. Un sous-système de rang *i* peut être constitué d'une ou plusieurs *entités* ; il communique avec les autres sous-systèmes de même rang : on parle alors de la *couche de rang i* ou, plus simplement, de la *couche i*. Les avantages d'une telle structure sont multiples :

- Une architecture de communication se définit entièrement en décrivant les services offerts par chaque couche, les interfaces entre les couches adjacentes et la manière dont ces couches coopèrent avec les entités du même niveau (les entités *homologues*) dans les autres systèmes.
- On peut développer séparément et simultanément toutes les couches d'une architecture de communication, une fois définies les interfaces entre les différents sous-systèmes.
- Le nombre d'interfaces à définir est minimal : il suffit de décrire, pour chaque niveau, les interfaces avec la couche supérieure (sauf pour la couche la plus élevée de l'architecture) et avec la couche inférieure (sauf pour la couche la plus basse). Les coopérations entre entités homologues sont régies par un ou plusieurs *protocoles*.

Ainsi, chaque couche fournit des services aux entités des couches supérieures et s'appuie sur les services offerts par les entités des couches inférieures. La couche la plus élevée offre à l'utilisateur tous les services utilisables dans l'architecture ; la couche la plus basse communique directement avec le support de transmission.

1.2 Terminologie employée

Dans la suite, nous utilisons la notation (*i*) pour signifier « de niveau *i* », afin de ne pas alourdir inutilement les définitions et les notations employées.

- *Service* (*i*). Capacité que possède la couche (*i*) et les couches inférieures à celle-ci, fournie aux entités ($i + 1$), à la frontière entre la couche (*i*) et la couche ($i + 1$). Les services sont invoqués par des *primitives*, spécifiques du service.
- *Primitive.* Demande de service par une entité de niveau supérieur à une entité de niveau inférieur.
- *Protocole* (*i*). Ensemble de règles et de formats déterminant les caractéristiques de communication des entités (*i*) lorsqu'elles effectuent les fonctions nécessaires à l'exécution du service (*i*). Le protocole utilise des unités de données appelées *PDU(i)* [*Protocol Data Unit*(*i*)].

- *Point d'accès à des services (i)* [*SAP(i)*, *Service Access Point(i)*]. Point où les services (*i*) sont fournis par une entité (*i*) à une entité (*i* + 1). Les unités de données du service *SDU*(*i* + 1) [SDU, *Service Data Unit*] traversent les *SAP(i)*.
- *SDU*(*i* + 1). Unités de données du service échangées localement entre entités (*i* + 1) et entités (*i*) pour l'exécution d'un service (*i*).
- *PDU(i)*. Unité de données du protocole, échangée entre entités (*i*) homologues.

1.3 Notion d'encapsulation

Une entité (*i* + 1) qui fait appel aux services assurés par une entité (*i*) lui transmet les données concernées et utilise la primitive appropriée à la demande d'exécution du service. Pour cela, l'entité (*i*) construit une (ou plusieurs) unité(s) de données, pour transporter les données, conformément au protocole (*i*). Celles-ci sont insérées dans le champ de données de la PDU(*i*) : on dit que les données (*i* + 1) sont *encapsulées* dans une unité de données (*i*).

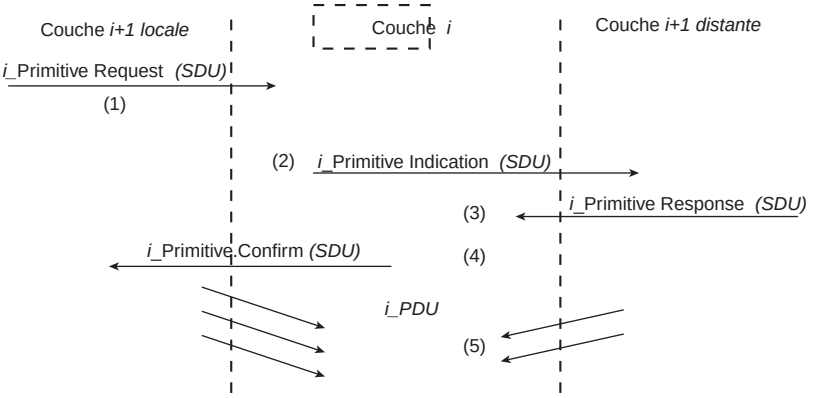
Une entité (*i*) reçoit donc les données émanant de l'entité (*i* + 1) sous la forme d'une SDU(*i*). À chaque SDU(*i*), l'entité (*i*) ajoute – en tête et/ou en queue – des informations de contrôle permettant d'exécuter le service demandé pour constituer une PDU(*i*) qui est soumise à la couche inférieure. Ainsi, une PDU(*i*) constitue une SDU(*i* – 1) à l'interface entre les couches (*i*) et (*i* – 1). Inversement, une SDU(*i*) contient une PDU(*i* + 1).

Dans certaines couches, les protocoles employés pour rendre le service (*i*) peuvent utiliser plusieurs PDU(*i*) pour traiter une SDU(*i*). En outre, les services peuvent nécessiter des PDU spécifiques comme, par exemple, des PDU d'acquittement qui ne contiennent pas de champ de données.

Une couche fournit un ensemble de services au niveau supérieur, invoqués par des primitives. On utilise quatre primitives, selon le sens et la nature de l'interaction : la requête, l'indication, la réponse et la confirmation (voir figure 4.1).

Figure 4.1

Les différents types de données échangées entre couches adjacentes ou entre couches homologues.



(1) L'entité (*i+1*) locale fabrique la primitive demandant un service à une entité de la couche (*i*) locale et lui fournit les données (*SDU*)
(2) L'entité (*i*) distante prévient l'entité (*i+1*) destinataire par une primitive de type Indication
(3) L'entité (*i+1*) distante donne sa réponse dans une primitive de type Response
(4) L'entité (i) locale transmet la réponse de l'entite (i+1) distante dans une primitive de type Confirm
(5) Les entités (*i*) fabriquent les *PDU* (*i*) qui transportent les données des entités (*i+1*)

Comme on le voit à la figure 4.1, la partie données de la primitive est la SDU : son contenu est totalement transparent pour le fournisseur du service. Pour signaler un événement à la

couche supérieure lorsque le protocole le prévoit, le fournisseur utilise une primitive de type *Indication*. Le destinataire de la primitive renvoie une primitive de type *Response* (si le protocole le requiert). Enfin, une primitive de type *Confirm*, émise par son prestataire de service, permet à l'auteur de la requête d'être informé de la fin – correcte ou non – de sa requête. Un paramètre précise alors les conditions de l'exécution de la requête.

2 Modèle OSI (*Open System Interconnection*)

La compatibilité (l'interopérabilité) entre équipements hétérogènes (constructeurs, fonctions ou générations de matériels différents…) implique des normes d'interconnexion définissant le comportement de chaque équipement vis-à-vis des autres. Tout équipement (ou ensemble d'équipements) à interconnecter est un *système ouvert* (un ordinateur, un terminal, un réseau…), s'il respecte des normes d'interconnexion. Le modèle OSI est une architecture abstraite de communication, décrit dans la norme *X.200* de l'ITU. Il est composé de sept couches, chacune remplissant une partie bien définie des fonctions permettant l'interconnexion.

Remarque

Open System Interconnection se traduit en français par « interconnexion des systèmes ouverts », ce qui donne l'acronyme ISO. Pour éviter toute confusion entre le modèle et l'organisme de normalisation, nous parlerons du modèle OSI.

2.1 Différentes couches du modèle OSI

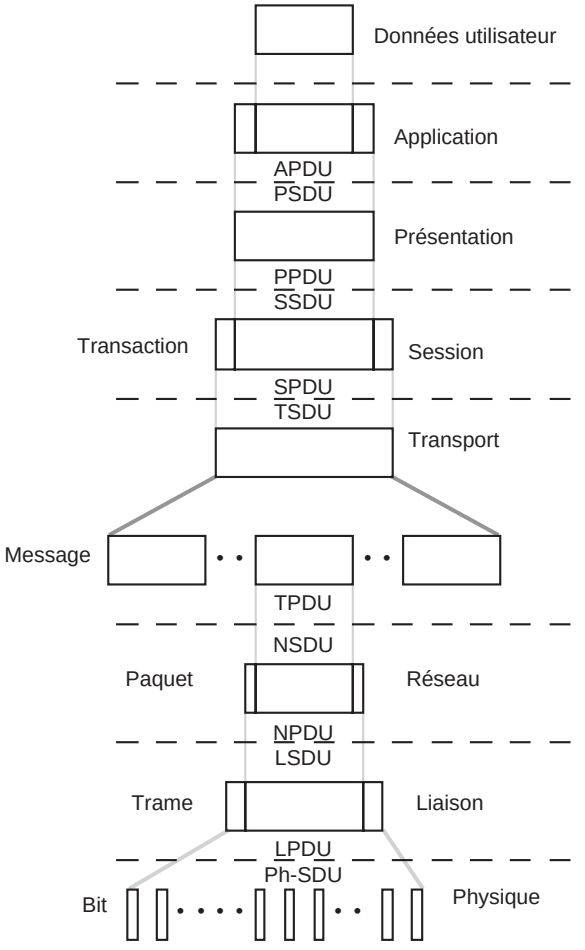
La figure 4.2 montre les SDU et PDU de toutes les couches du modèle OSI. La couche de plus bas niveau est la couche *Physique.* Elle se caractérise par son taux d'erreurs, la vitesse de transmission et le délai de transit. L'unité de données manipulée à ce niveau est le *bit*. Au-dessus, la couche *Liaison de données* fournit les moyens d'établir, de maintenir et de gérer les connexions de liaison de données entre entités de réseau. Elle détecte et corrige, dans la mesure du possible, les erreurs de la couche physique. La *trame* est l'unité de données manipulée par la liaison de données. La couche *Réseau* fournit aux entités de transport les moyens d'établir, de maintenir et de gérer les connexions de réseau ; elle manipule des *paquets* et les achemine à travers le réseau. Ces trois premières couches ont été définies dans la norme *X.25*, que nous avons évoquée au chapitre 3. Tous les équipements du réseau, dans les systèmes intermédiaires comme dans les systèmes d'extrémité, contiennent des entités de réseau. Seuls les systèmes d'extrémité implémentent les couches supérieures. La quatrième couche, *Transport*, assure un transfert de données fiable et optimise les coûts d'utilisation des services réseau disponibles, compte tenu des exigences de service des entités supérieures. Le *message* est l'unité de données qu'elle manipule. Cette couche charnière masque, pour les couches hautes, les disparités entre réseaux. La cinquième couche, *Session*, organise et synchronise le dialogue entre les systèmes d'extrémité. La sixième couche, *Présentation*, s'occupe de la représentation des informations, quels que soient les modes de représentation interne des machines et dans le réseau. Elle peut se charger aussi de la compression de données et de la sécurité des informations échangées (chiffrement/déchiffrement, qu'on appelle parfois cryptage/décryptage). La dernière couche est la couche *Application*. Elle contient les entités d'application, c'est-à-dire les processus des utilisateurs

qui génèrent les informations à échanger. Au sens du modèle OSI, une entité d'application peut être une entité de messagerie ou de transfert de fichiers par exemple. Il n'y a pas de mot dans la langue française pour qualifier l'unité de données des trois dernières couches. On parle en général de *messages*.

Dans les échanges à l'intérieur du modèle OSI, nous trouvons des PDU et des SDU : *Ph_PDU* désigne les PDU du niveau Physique, *L_PDU* celle du niveau Liaison, les *N_PDU* (*N* pour network) sont les *PDU* de niveau Réseau… La lettre préfixe est *T* pour Transport, *S* pour Session, *P* pour Présentation et *A* pour Application. De même, nous rencontrons des SDU de niveau Liaison, de niveau Réseau, etc. Les premières sont des *L_SDU* et les suivantes des *N_SDU*, etc. À titre d'exemple, les *N_SDU* sont des unités de données provenant d'entités de la couche Transport, comme les paramètres de connexion fournis par l'entité de transport pour l'ouverture du circuit virtuel au niveau Réseau.

Figure 4.2

Empilement des sept couches du modèle OSI, avec les PDU et SDU utilisées.



A priori, chaque couche (sauf la couche Physique) peut utiliser les quatre types de primitives. Dans le modèle de référence, le nom complet d'une primitive est couramment obtenu en faisant précéder son type par l'initiale de la couche, suivie de la nature de l'opération : *Request, Indication, Response, Confirm*.

Exemple

La primitive *T_CONNECT.Request* est la requête (*Request*) de demande d'ouverture de connexion (*CONNECT*) émise par l'entité de couche Session vers son entité de Transport (*T*). Cette primitive demande à la couche Transport d'établir une connexion.

2.2 Modèle d'architecture à 5 couches

Le découpage proposé par le modèle initial a connu une évolution, du fait de la complexité des normes. En effet, les couches 5 et 6 (Session et Présentation) ont été progressivement vidées de leur substance au fil du temps : les couches adjacentes ont intégré leurs fonctionnalités. La couche Transport assume le plus souvent celles de la couche Session, alors que les applications incorporent la description des structures de données, même si la compatibilité totale entre systèmes ou versions de systèmes est encore loin d'être parfaite… Les architectures existantes ne respectent plus vraiment le découpage tel que le décrit le modèle initial, mais le principe même de la structuration en couches reste la base de toutes les architectures. La terminologie et le découpage des services réseau ont été adoptés par tous : on parle couramment de couche 2 pour la couche Liaison, de couche 3 pour désigner la couche Réseau, même si l'architecture considérée ne les place à cet endroit !

Remarque

Dans l'architecture ATM (que nous avons citée au chapitre 3), il n'y a que deux couches basses : la couche 1 ou Physique et… la couche 3 ou couche ATM. La couche supérieure, baptisée *Adaptation à l'ATM* (ou AAL, *ATM Adaption Layer*) est considérée comme une couche Transport donc de niveau 4.

2.3 Modèle d'architecture adapté aux réseaux locaux

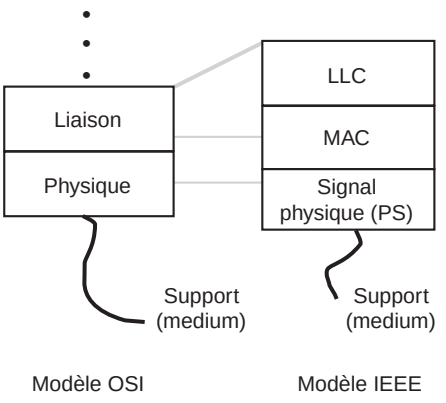
L'IEEE (*Institute for Electricity and Electronics Engineers*), une société américaine, a constitué un comité d'études au début des années 1980 (le groupe 802, essentiellement des constructeurs américains). L'objectif est alors de développer des standards pour la transmission de messages à haut débit entre systèmes informatiques, à travers un support partagé par ces systèmes et indépendant de leur architecture. Ce comité a publié une série de standards nommés 802.*n*. Nous les décrivons plus en détail au chapitre 5. L'ISO a ensuite repris les travaux du groupe 802 et les a référencés sous le numéro 8802.*n* (le *n* des références ISO est identique au *n* utilisé dans les références de l'IEEE).

La modélisation de l'IEEE redéfinit les niveaux 1 et 2 du modèle OSI pour les réseaux locaux. Cette modélisation spécifie les services rendus à la couche supérieure et la façon d'implanter les niveaux 1 et 2. La figure 4.3 montre la correspondance entre les couches 1 et 2 du modèle OSI et les couches du modèle IEEE. Nous remarquons que, par rapport au modèle OSI, l'architecture normalisée dans les réseaux locaux découpe la couche Liaison en deux sous-couches : MAC (*Medium Access Control*) et LLC (*Logical Link Control*).

Le niveau MAC, comme son nom l'indique, définit la *méthode d'accès*, c'est-à-dire la manière dont il faut envoyer et recevoir les données sur le support partagé par toutes les stations du réseau local. Il existe différentes méthodes d'accès, incompatibles entre elles. CSMA/CD, la méthode d'accès des réseaux Ethernet, est la plus connue et la plus utilisée. Elle est décrite dans le standard 802.3. La sous-couche LLC masque les disparités des méthodes d'accès. Le chapitre 5 consacré aux réseaux locaux décrit le fonctionnement détaillé de ces deux sous-couches.

Figure 4.3

Comparaison des modèles OSI et IEEE : des positionnements différents.



3 Architecture TCP/IP (*Transmission Control Protocol/Internet Protocol*)

L'architecture TCP/IP porte le nom des protocoles principaux qui la constituent, à savoir TCP et IP ; on l'a définie dans les années 1960 pour le réseau ARPAnet. Elle s'est considérablement développée avec le succès d'Internet. Les chapitres 6 et 7 consacrés à Internet et à chacun de ses deux protocoles phares TCP et IP en détaillent le fonctionnement.

3.1 Standardisation de l'architecture TCP/IP

La conception de l'architecture TCP/IP a été très différente de la méthode utilisée dans le modèle OSI. Les organismes de normalisation internationaux ont en effet défini des concepts universels, répondant à tous les besoins possibles, et des fonctionnalités en dehors de tout souci de réalisation. Les normes de l'ISO sont de ce fait très complexes, car elles contiennent de nombreuses options, destinées à couvrir l'ensemble des fonctionnalités proposées, quel que soit l'environnement d'application. Les concepteurs de l'architecture TCP/IP se sont attachés à décrire en premier les protocoles, avant de proposer un modèle de référence décrivant leur empilement. De plus, ils ont privilégié une approche pragmatique : trouver une solution rapide et opérationnelle, même si elle ne résout pas la totalité du problème.

3.2 Présentation de l'architecture TCP/IP

L'architecture TCP/IP compte quatre couches, comme le montre la figure 4.4 : *Réseau physique*, *Réseau*, *Transport* et *Application.*

Aucune caractéristique particulière n'est requise pour l'infrastructure du ou des réseaux physiques traversés. La couche Réseau physique est donc quelconque. La couche Réseau (qu'il serait d'ailleurs plus juste de dénommer « interréseaux ») assure la communication entre les réseaux grâce au protocole IP (*Internet*[1] *Protocol*). On utilise la commutation de

1. Internet est l'apocope d'*internetworking* (interréseaux).

paquets de type datagramme pour acheminer des données entre les systèmes d'extrémité, quelle que soit la technologie réseau qu'ils emploient. Le protocole IP gère les datagrammes : il les achemine jusqu'à leur destinataire, se charge du routage et de l'adaptation de la taille des données au réseau sous-jacent. IP définit en fait un service minimal, l'acheminement des datagrammes à travers l'interconnexion de réseaux. Ce service est sans connexion et sans garantie. Il ne fait aucune hypothèse quant à la fiabilité des réseaux traversés.

Au-dessus de la couche IP, l'architecture définit deux protocoles de transport, en fonction des besoins des utilisateurs : un protocole en mode connecté, TCP (*Transmission Control Protocol*), et un protocole en mode non connecté, UDP (*User Datagram Protocol*). TCP est destiné à fiabiliser les échanges : il permet aux utilisateurs situés aux extrémités de la connexion d'échanger des données, avec contrôle de flux, contrôle d'erreur, contrôle de séquence entre les deux extrémités. Il garantit en particulier la livraison séquentielle des données, leur non-duplication et la récupération des données manquantes. UDP est un protocole non fiable. Il sert aux applications qui ne souhaitent pas ralentir les transferts de données par la lourdeur de la mise en œuvre des processus de gestion du mode connecté, ou à celles qui n'ont pas besoin de la fiabilité de TCP : inutile de mettre en œuvre des mécanismes complexes pour garantir le séquencement des messages par exemple, quand il n'y a qu'un seul message à émettre.

Enfin, la couche Application se greffe directement au-dessus de la couche Transport. Elle contient tous les protocoles de haut niveau qu'un utilisateur souhaite avoir à sa disposition : Telnet, FTP, SMTP, HTTP… Nous verrons plusieurs exemples d'applications au chapitre 9.

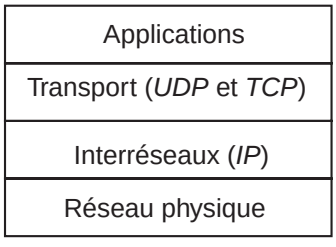


Figure 4.4

Architecture TCP/IP : un ensemble charnière TCP et IP.

4 Normalisation dans les télécommunications et les réseaux

Dans des domaines techniques comme les réseaux et les télécommunications, la normalisation répond aussi bien aux attentes des consommateurs qu'aux besoins des fabricants. D'un côté, elle offre aux utilisateurs la garantie que deux produits aux fonctions identiques mais de fabricants différents puissent fonctionner correctement ensemble. De l'autre, les industriels peuvent espérer toucher un plus grand nombre de consommateurs grâce à la normalisation de leurs produits. En effet, toute solution propriétaire provoque la réticence des utilisateurs et des entreprises à dépendre d'un seul fournisseur pour leur approvisionnement : dans la mesure où certains équipements sont vitaux pour la survie même de l'entreprise, la continuité du service passe par la possibilité de disposer de plusieurs sources indépendantes d'approvisionnement.

Différents organismes de normalisation édictent des avis qui couvrent tous les aspects d'un équipement : aspects électriques, mécaniques, interconnexion… Les principaux organismes internationaux de normalisation regroupent des représentants des industriels, des administrations, des utilisateurs : l'ISO (*International Standardization Organization*), l'ITU (*International Telecommunications Union*)… On trouve également divers groupements de constructeurs comme : l'ECMA (*European Computer Manufacturer*), l'EIA (*Electronic Industries Association*)… Dans Internet, l'IAB (*Internet Architecture Board*) définit la politique du réseau à long terme alors que l'IETF (*Internet Engineering Task Force*) s'occupe de l'homogénéité des solutions et publie les RFC (*Request For Comments*).

Une norme passe par plusieurs étapes : le résultat des compromis entre les différentes parties s'exprime dans un document brouillon (*draft*). La forme stable du document est publiée sous forme de *draft proposable*. Le *Draft International Standard* est la forme quasiment définitive. Il constitue une base de travail pour les industriels. Enfin, l'IS (*International Standard*) est la forme définitive du document. L'organisme considéré publie la forme finale, en utilisant une référence qui dépend de son domaine d'application.

Résumé

Une architecture de communication formalise l'empilement de couches de protocoles et des services offerts pour assurer l'interconnexion des systèmes. Nous avons présenté le modèle abstrait, défini dans le milieu des années 1970, communément appelé *modèle de référence* ou *modèle OSI*. Si ce modèle ne précise pas le fonctionnement détaillé des systèmes réels, il a apporté un vocabulaire et des concepts de structuration encore utilisés de nos jours. La preuve en est donnée par les adaptations apportées au modèle OSI initial pour tenir compte de l'avènement des réseaux locaux, aujourd'hui les plus utilisés dans les entreprises ou chez les particuliers. Nous avons présenté succinctement l'architecture de communication TCP/IP, la plus largement répandue puisqu'elle est utilisée dans Internet. Les prochains chapitres détailleront les protocoles un par un. Enfin, nous évoquons brièvement le processus de normalisation.

Problèmes et exercices

Exercice 1 Choix des primitives d'un niveau donné

Énoncé

Une interface entre deux couches adjacentes peut se décrire par l'ensemble des primitives qu'elle offre aux entités de la couche supérieure. Quels sont les critères à retenir dans le choix des primitives d'un niveau donné ?

Solution

Trois considérations principales doivent guider le concepteur d'un niveau donné :

- définir les services offerts par la couche (i) aux entités ($i + 1$), en tenant compte de la position de ce niveau dans l'architecture de communication ;
- proposer un nombre minimal de primitives différentes, pour ne pas compliquer inutilement l'interface ;
- minimiser le nombre de paramètres à prendre en compte dans chaque primitive définie.

Pour satisfaire la première condition, il faut avoir une idée très précise du fonctionnement de la couche (i). Par ailleurs, la complexité de l'interface dépend de l'endroit où se situe le niveau considéré : une couche de bas niveau offre forcément des services plus limités qu'une couche haute de l'architecture. Il faut donc définir les services qui seront disponibles et décider du nombre d'entités nécessaires à leur gestion. Il faut également déterminer les interactions entre les différentes entités, afin de définir la circulation des informations au sein de la couche.

La deuxième condition doit être remplie dans un souci d'efficacité. Si l'interface compte un grand nombre de primitives différentes, l'entité ($i + 1$) risque au mieux de ne pas en utiliser toutes les subtilités. Au pire, elle peut entraîner une baisse des performances préjudiciable à toute l'architecture. En effet, le nombre élevé de primitives risque fort de mener à des doublons. À l'opposé, les entités ($i + 1$) pourraient laisser certaines primitives inusitées.

La troisième condition fournit une meilleure lisibilité du service demandé et accélère son traitement. Supposons que chaque paramètre de la primitive soit géré par une entité. Si la primitive contient 10 paramètres, il faudra concevoir 10 entités pour les manipuler, définir leur mode de coopération et prévoir toutes les situations possibles entre chaque paire d'entités concernées par le traitement du service… Cela risque de provoquer, à l'intérieur du niveau, des boucles dans le parcours de données entre les entités chargées de traiter le service demandé. Dans tous les cas, le temps de traitement de la primitive s'en trouve notablement allongé par rapport à une primitive plus simple. Il faut donc assurer un compromis entre des conditions contradictoires : une primitive simple se traite plus efficacement, mais elle n'exécute qu'un service limité par rapport à une primitive plus sophistiquée. Il faut alors augmenter le nombre de primitives disponibles. On risque de tomber dans le travers de définir un excès de primitives, forcément très mal utilisé…

Exercice 2 Procédures en cas de panne du réseau

Énoncé

Pendant un transfert de données sur un circuit virtuel, une panne se produit à l'intérieur d'un réseau respectant la norme *X.25*.

a Quelles sont les conséquences de la panne dans le réseau ?

b Quels types de primitives le réseau utilise-t-il pour signaler l'événement aux deux extrémités du circuit virtuel ?

c Quelles en sont les conséquences pour les entités de niveau Transport ?

d Quelles sont les stratégies de reprise possibles ?

Solution

a La panne dans le réseau provoque la rupture du circuit virtuel déjà établi. Les nœuds voisins du nœud défaillant (ou les nœuds situés aux deux extrémités de la liaison de données défaillante) détectent cet événement et le propagent jusqu'aux extrémités du circuit virtuel.

b Les entités de Réseau des systèmes d'extrémité le signalent à leurs entités de Transport en envoyant une primitive de type Indication, indiquant la cause de la rupture du circuit virtuel, lorsque celle-ci est connue.

c En recevant la primitive de type Indication, les entités de Transport réagissent en fonction des choix qu'elles ont faits au moment de l'établissement de la connexion de Transport. Par exemple, si elles ont ou non négocié la possibilité d'un fonctionnement en mode dégradé.

d Deux stratégies sont possibles : soit il y a abandon de la communication interrompue, soit il y a tentative de reprise, en particulier si la connexion de Transport a prévu un fonctionnement en mode dégradé.

Dans le premier cas, la connexion est rejetée ; il faut rétablir ultérieurement une connexion de Transport, en utilisant la même connexion Réseau (si on utilise un CVP, circuit virtuel permanent, réinitialisé) ou en ouvrant une nouvelle connexion Réseau avant d'établir la nouvelle connexion de Transport.

Dans le second cas, les entités de Transport ont prévu d'utiliser une procédure de récupération d'erreurs, ce qui leur permet de redémarrer le transfert des données par la réémission de toutes les données non encore acquittées.

Exercice 3 Établissement de connexions Liaison et Réseau

Énoncé

Une entité de Réseau souhaite envoyer des données provenant des couches supérieures.

a Décrivez les échanges de primitives entre une entité de Réseau quelconque et l'entité de Liaison, en supposant que la liaison de données soit exploitée avec le protocole LAP-B, qu'elle ne soit pas encore initialisée et qu'il n'y ait aucune erreur de transmission des données.

b Que se passe-t-il s'il se produit une erreur de transmission ?

c En supposant que la liaison de données soit opérationnelle, décrivez les échanges entre l'entité de Réseau locale et l'entité de Liaison locale pour ouvrir un circuit virtuel.

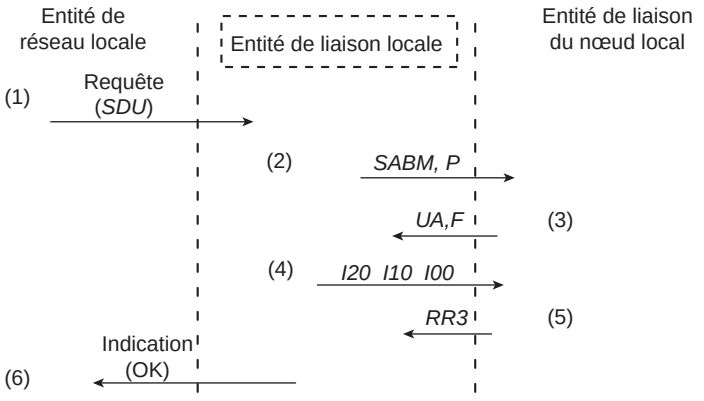
Solution

a Supposons que nous soyons du côté du système d'extrémité qui prend l'initiative. Nous appellerons *entité de Réseau locale* l'entité de Réseau située dans notre système et *entité de Réseau distante* l'entité de Réseau du système d'extrémité qui doit recevoir les données. De même, nous appellerons *nœud local* le point d'accès au réseau du système d'extrémité local et *nœud distant* le point d'accès au réseau du système d'extrémité distant. Les échanges entre les entités locales de Liaison et de Réseau concernent le système d'extrémité local qui désire envoyer des données.

Pour envoyer ses données à l'entité de Réseau distante, l'entité de Réseau locale émet une primitive de type Requête vers son entité locale de Liaison. Cette requête contient les données à transmettre. À la réception de cette primitive, l'entité de Liaison locale constate qu'elle ne peut pas assurer le service demandé puisque la liaison de données n'est pas opérationnelle. Elle établit donc la liaison à l'aide d'une trame d'initialisation (une trame U, par exemple SABM,P et reçoit en retour la trame U d'acceptation UA,F, [voir chapitre 2]). Puis l'entité de Liaison locale envoie à l'entité de Liaison du nœud local autant de trames I que de paquets à envoyer. Elle reçoit les acquittements des trames I émises par l'entité de Liaison du nœud local, conformément au protocole de liaison. Une fois le transfert de données correctement terminé, l'entité de Liaison locale envoie à l'entité de Réseau locale une primitive de type Indication pour la prévenir de la (bonne) fin du transfert. La figure 4.5 montre les échanges entre les deux entités locales et l'entité de Liaison du nœud local.

Figure 4.5

Échanges entre les différentes entités pour le traitement de la requête : une requête réseau simple peut donner lieu à de multiples échanges du protocole de liaison.



(1) L'entité de réseau locale du système d'extrémité émet une requête pour envoyer les données (SDU).

(2) L'entité de liaison locale demande l'initialisation de la liaison de données en envoyant la trame SABM,P

(3) l'entité de liaison du nœud d'accès accepte l'initialisation de la liaison en envoyant UA,F.

(4) L'entité de liaison d'accés émet autant de trames I que nécessaire (ici 3 trames).

(5) L'entité de liaison du nœud local acquitte les trames I.

(6) L'entité de liaison locale prévient l'entité de réseau locale de la bonne fin du transfert de données.

b En cas d'erreur de transmission, l'entité de Liaison située dans le nœud d'accès émet une trame REJ si la trame erronée est suivie d'une autre trame I. Si la trame erronée est la dernière, elle est retransmise par l'entité de Liaison locale après expiration du temporisateur associé.

Remarque

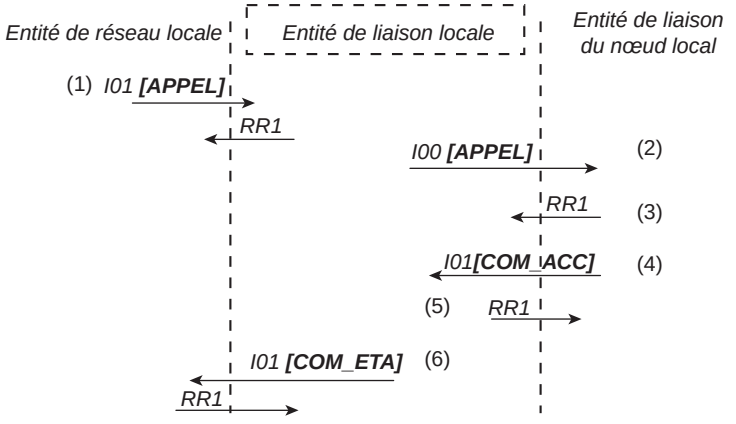
L'échange des trames et des primitives montre bien que les entités de Réseau (locale et distante) ne peuvent pas être averties d'une erreur de transmission, puisque cet événement est totalement pris en charge par leurs entités de Liaison respectives : les entités de Réseau ne manipulent que des unités de données exemptes d'erreurs (en cas d'erreurs résiduelles, elles gèrent des unités de données dont les erreurs n'ont pas été détectées par les entités de Liaison).

C Pour ouvrir le circuit virtuel, l'entité de Réseau locale envoie une primitive de type Requête avec les paramètres de la future connexion contenus dans le paquet *APPEL* (voir chapitre 3). L'entité de Liaison locale insère ce paquet dans une trame I et l'envoie dans le réseau. L'entité de Liaison du nœud local acquitte la trame I contenant le paquet *APPEL*.

Au bout d'un certain temps, l'entité de Liaison locale reçoit de l'entité de Liaison du nœud local la trame I contenant le paquet *COMMUNICATION_ETABLIE* (provenant de l'entité de Réseau distante). Elle extrait ce paquet du champ de données et le soumet à l'entité de Réseau locale. Celle-ci analyse le contenu du paquet et apprend ainsi que l'entité de Réseau distante a accepté la connexion. Elle peut alors commencer le transfert de données. La figure 4.6 décrit les échanges entre les entités concernées. Dans cette figure, les unités de données manipulées par l'entité de Réseau sont en italique gras ; le contenu du champ de données d'une trame est entre crochets. Les trames sont les unités de données manipulées par les entités de Liaison.

Figure 4.6

Échange de données entre les entités de Réseau et de Liaison.



(1) L'entité de réseau locale émet une demande d'ouverture de connexion (*APPEL*).

(2) L'entité de liaison locale insère le paquet *APPEL* dans une trame *I* référencée *I00.*

(3) L'entité de liaison du nœud local acquitte la trame contenant le paquet *APPEL*.

(4) L'entité de liaison du nœud local a reçu la réponse de l'entité de réseau distante. Elle l'insère dans le champ de données de la trame *I01.*

(5) L'entité de liaison locale acquitte la trame contenant la réponse de l'entité de réseau distante.

(6) L'entité de liaison locale soumet le résultat de la requête à l'entité de réseau locale par une primitive de type *Indication* (*COM_ETA*).

EXERCICE 4 NOTIONS DE SDU ET PDU DES NIVEAUX RÉSEAU ET LIAISON

Énoncé

Utilisez les notations de ce chapitre pour décrire les PDU et SDU échangées entre entités locales de couches adjacentes : reprenez l'exercice précédent demandant les échanges de données nécessaires à l'ouverture du circuit virtuel en indiquant quelles sont les PDU et les SDU des niveaux Réseau et Liaison de données.

Solution

Dans ces échanges, nous trouvons deux types de PDU et deux types de SDU : les PDU de niveau Liaison (*L_PDU*) et de niveau Réseau (*N_PDU*). Puisque nous nous plaçons au niveau de l'interface Liaison et Réseau, nous ne pouvons pas trouver de *N_SDU* dans notre échange.

Les *N_PDU* sont les unités de données manipulées par les entités de Réseau : il s'agit du paquet *APPEL* (généré par l'entité de Réseau locale à partir des données fournies par l'entité de Transport locale) et du paquet *COMMUNICATION_ETABLIE* (généré par l'entité de Réseau distante et transporté à travers le réseau de communication jusqu'à l'entité de Réseau locale).

Les *L_SDU* sont les unités de données du service fourni par la couche Liaison. Ces unités de données sont insérées dans le champ de données d'une trame d'informations. Elles contiennent les paquets *APPEL* et *COMMUNICATION_ETABLIE*.

Les *L_PDU* sont les unités de données que gère le protocole de liaison. Il s'agit donc des trames de type I, S et U. Dans notre exemple, nous trouvons les trames I transportant les paquets et les trames S qui acquittent les trames précédentes : d'une part les trames I00 et I01 (*L_PDU* avec champ de données) et, d'autre part, les trames RR1 (*L_PDU* d'acquittement, sans données).

EXERCICE 5 CONTENU DES PDU D'UN NIVEAU QUELCONQUE

Énoncé

a Que peut contenir une PDU(*N*), c'est-à-dire une PDU de niveau *N*, pour $N \leq 7$?

b Donnez des exemples pour $N = 2$ et $N = 3$ dans le cas de *X.25* (gestion des circuits virtuels avec LAP-B comme protocole de liaison).

Solution

a En nous appuyant sur l'exercice précédent, nous voyons qu'il y a deux types de PDU(*N*) : celles qui contiennent des informations de supervision (ou de commande) et celles qui contiennent des données et/ou des informations de commande de l'utilisateur. Les entités homologues s'échangent le premier type de PDU pour la supervision du protocole de niveau *N*. Le second type de PDU correspond à celles qui transportent des données et des informations de commande du niveau $N + 1$: ce sont les PDU qui transportent tout ou partie d'une SDU de niveau *N*, c'est-à-dire une PDU de niveau $N + 1$.

b Lorsque $N = 3$, il s'agit de *N_PDU*. Ce sont soit des PDU de commande, comme *APPEL*, *RR*, *LIB_CONF*, *COMMUNICATION_ETABLIE*, soit des paquets contenant les données de l'utilisateur ou des informations de commande provenant de la couche Transport.

Quand $N = 2$, les *L_PDU* de commande sont, par exemple, *SABM*, *UA*, *RR*… Les trames I sont les seules *L_PDU* contenant des données d'un utilisateur de la couche Réseau, quel que soit leur contenu.

Remarque

Puisque les trames I sont les seules *L_PDU* contenant des unités de données du protocole réseau, nous voyons bien maintenant la différence entre une *trame RR* et un *paquet RR :* une trame RR est une trame S, c'est-à-dire une *L_PDU* de commande. Un paquet RR est une *N_PDU* de commande transportée dans une trame I.

Exercice 6 Quelques primitives du modèle OSI

Énoncé

Un système d'extrémité utilise une architecture de communication conforme au modèle OSI. Considérons par exemple la primitive : *N_CONNECT.Indication*.

a Dans quelle couche du modèle OSI se situe l'entité qui émet cette primitive ?

b À quelle entité est destinée cette primitive ?

c Considérons maintenant la primitive : *L_DATA.Request*. À quel niveau se situe l'entité qui émet cette primitive ? Quelle est son utilité ?

Solution

a La primitive provient d'une entité de Réseau. Elle signale qu'une demande d'ouverture de circuit virtuel lui est parvenue.

b La primitive s'adresse à une entité de Transport, qui décide si elle accepte ou non la connexion. La réponse de cette entité se trouve dans la primitive *N_CONNECT.Response* (elle contient un paramètre codant l'acceptation ou le refus de la connexion).

c La primitive *L_DATA.Request* émane d'une entité de Réseau, destinée à l'entité de Liaison. Elle fournit à cette dernière les paquets de données à émettre. En la recevant, l'entité de Liaison construit une trame d'informations avec les informations de commande appropriées (en-tête et queue de la trame), puis elle place le paquet dans le champ de données de la trame.

Exercice 7 Rôle des primitives du modèle OSI

Énoncé

Un système d'extrémité utilise une architecture de communication conforme au modèle OSI. Considérons la primitive de service : *N_CONNECT.Request*.

a Dans quelle couche du modèle OSI se situe l'entité émettrice de la primitive ?

b À qui est destinée cette primitive ?

c À quoi sert cette primitive, si le système d'extrémité accède à un réseau public de données utilisant le protocole *X.25* ?

d Comment l'entité émettrice de la primitive *N_CONNECT.Request* saura-t-elle que sa primitive a été traitée ?

Solution

a L'entité émettrice se situe *au-dessus* de la couche Réseau puisqu'elle utilise ses services. Elle se trouve donc dans la couche Transport.

b Elle s'adresse au prestataire de services, donc à une entité de niveau Réseau.

c Elle sert à déclencher le processus d'établissement d'un circuit virtuel.

d Elle sait que sa primitive est prise en compte en recevant une primitive *N_CONNECT.Confirm* de la part de l'entité de niveau Réseau. Cette primitive lui indique comment s'est déroulé l'établissement de la connexion (succès ou échec).

Exercice 8 Relations entre PDU et SDU des différents niveaux

Énoncé

Un utilisateur souhaite envoyer une photo de 448 000 octets dans un réseau. L'architecture de communication respecte le modèle à 5 couches évoqué à la section 2.2, c'est-à-dire un modèle de référence OSI adapté aux réseaux locaux. Sachant que la couche Transport gère des messages de 1 Kilo-octet (1 Kilo-octet vaut 1 024 octets) et utilise pour cela un en-tête de 24 octets pour gérer 1 000 octets de données, que la couche Réseau utilise des paquets de 128 octets dont les 3 premiers constituent l'en-tête du paquet, on demande :

a Quelle est, en octets ou en kilo-octets, la taille d'une *T_SDU ?* Quelle est celle d'une *T_PDU* de données ? D'une *N_SDU* ? D'une *N_PDU* de données ? D'une *L_SDU* ? D'une *L_PDU* de données ?

b Combien de paquets de données faut-il envoyer pour transmettre la photo si on ne tient pas compte des paquets à envoyer pour la gestion du protocole de transport ? Même question si on en tient compte.

c L'utilisateur bénéficie maintenant de l'architecture TCP/IP dans son ordinateur. Quel protocole de transport l'application utilise-t-elle ? Pourquoi ?

d TCP gère des segments de 64 Kilo-octets. Combien de segments différents TCP fabrique-t-il pour transporter la photo sur Internet ?

Solution

a La *T_SDU* est la photo à envoyer. Sa taille est de 448 000 octets. L'entité de Transport gère des unités de données de 1 Kilo-octet ; une *T_PDU* de données contient donc 1 000 octets de données, une *N_SDU* en contient 1 024.

L'entité de Réseau utilise des paquets de 128 octets, dont les 3 premiers représentent l'en-tête du paquet. Une *N_PDU* de données contient 125 octets, alors qu'une *L_SDU* contient 128 octets dans son champ de données.

b Nous avons vu que dans chaque paquet se trouvent 125 octets de données. Il faut : 3 584 paquets de données pour transporter la photo. En fait, il faut envoyer : 448 000/ 1 024 = 437,5 soit 437 messages de 1 024 octets et un message à moitié plein. Cela fait en tout 3 589 paquets, sans compter les paquets nécessaires à la gestion de la connexion de Transport.

c Il utilise TCP pour que le destinataire puisse recevoir les différents paquets de la photo dans l'ordre et sans erreur…

d TCP fabrique 6 segments de 64 Kilo-octets de données plus un dernier segment de données partiellement rempli.