Les Protocoles de Transport

UVCI

Équipe Pédagogique Réseau Informatique @
UVCI 2018

Mai 2018
Version 1.0

Table des matières

Objectifs

A la fin de cette leçon, vous serez capable de :

- Connaître les notions utilisées par les protocoles de transports
- Connaître le principe de fonctionnement des protocoles de transport

Introduction

Une application utilise les services de la couche transport avec qui elle échange des données à travers une interface de programmation livrée avec la pile TCP/IP. Pour les échanges de données exigeant une grande fiabilité (une certaine qualité de service), le protocole de transport TCP (Transmission Control Protocol) est utilisé. Pour ceux qui ne nécessitent pas une telle fiabilité, un protocole de transport plus simple, appelé UDP (User Datagram Protocol) fournit un service de bout en bout en mode sans connexion.

I. Généralités sur les protocoles de transport

Objectifs

- Connaître les notions utilisées par les protocoles de transports

Un protocole de transport offre aux applications situées sur la machine de l'utilisateur une interface leur permettant d'utiliser les services offerts par le ou les réseaux physiques sous-jacents. Comme plusieurs applications sont susceptibles d'accéder au réseau, il faut les identifier sans ambiguïté ; on utilise pour cela la notion de port. Par ailleurs, les extrémités qui échangent des données sont repérées grâce à la notion de socket.

Un protocole de transport souhaite en outre s'assurer que l'en-tête contenant les informations nécessaires à la gestion du protocole n'a pas été altéré au cours de la transmission. Il emploie à cette fin des techniques de redondance appelées total de contrôle (ou checksum).

1. Notions utilisées dans les protocoles de transport

1.1. Notion de Port

Les identifiants d'application sont indispensables, car plusieurs applications différentes peuvent s'exécuter simultanément sur la même machine. Un protocole de transport doit donc savoir pour qui il travaille ! L'identifiant d'application est appelé *numéro de port* – ou *port* – (ce qui n'a rien à voir avec les ports d'un commutateur). Selon les systèmes d'exploitation, le numéro de port peut correspondre au PID (Process Identifier), l'identifiant du processus qui s'exécute sur la machine.

Il existe des milliers de ports utilisables, puisque le numéro de port tient sur 16 bits (soit 2^16 ports). Une affectation officielle des ports a été définie par l'IANA (Internet Assigned Numbers Authority), afin d'aider à la configuration des réseaux. Parmi ceux-ci, les ports 0 à 1023 sont les ports bien connus ou réservés (well known ports), affectés aux processus système ou aux programmes exécutés par les serveurs.

Le tableau ci-dessous nous donne une liste de numéro de port bien connu :

N° de port	20, 21	22	23	53	80	110	143	161
Service	FTP	SSH	Telnet	DNS	HTTP	POP3	IMAP	SNMP

Figure 1 : Numéros de ports bien connus

1.2. Interface entre le protocole de transport et l'application

Les applications ayant le choix du protocole de transport (TCP ou UDP), le système d'exploitation doit utiliser un système de nommage qui affecte un identifiant unique, appelé socket, à tout processus

local utilisant les services d'un protocole de transport. Le socket est constitué de la paire :

< *adresse IP locale, numéro de port local* >.

Pour identifier de manière unique l'échange de données avec le processus applicatif distant, le protocole de transport utilise un ensemble de cinq paramètres formés par le nom du protocole utilisé, le socket local et le socket distant :

< *protocole, adresse IP locale, numéro de port local ; adresse IP distante, numéro de port distant* >.

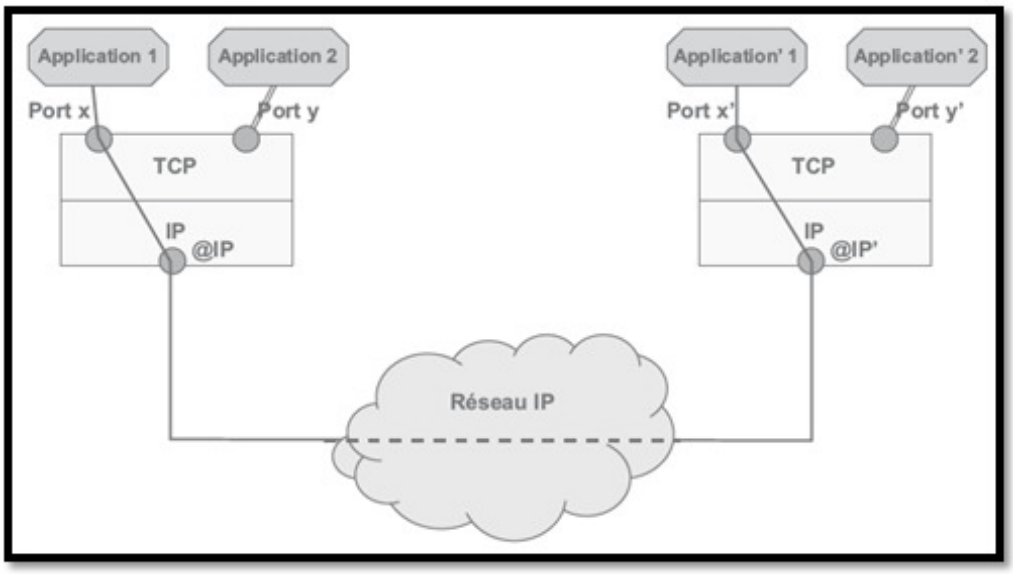

Figure 2 : Connexion entre un protocole de transport et une application

1.3. Total de Contrôle ou Checksum

Un total de contrôle est une information de redondance permettant de contrôler l'intégrité d'un bloc d'informations défini. Dans TCP ou dans UDP, le total de contrôle est un champ de 16 bits incorporé dans l'en-tête. Sa position dépend du protocole de transport utilisé.

2. Exercice : Exercices

[Solution n°1 p 15]

Exercice : Exercice 1

L'interface de programmation entre un protocole de transport et une application est le . Il est constitué d'une et d'un de

Exercice : Exercice 2

Lors des échanges de données, les protocoles de transport utilisent les paramètres formés par :

- ☐ le protocole de transport
- ☐ l'adresse IP locale
- ☐ le numéro de port local
- ☐ l'adresse IP distante
- ☐ le numéro de port distant

Exercice : Exercice 3

Affecter à chaque numéro de port, l'application correspondante.

Numéro de Port	Application
21	
23	
80	
53	

II. Les protocoles TCP et UDP

Objectifs

- Connaître le principe de fonctionnement des protocoles de transport

1. Le Protocole UDP (User Datagram Protocol)

1.1. Principe

UDP est un protocole de transport qui permet l'émission de messages sans l'établissement préalable d'une connexion (*mode non connecté*). C'est un *protocole non fiable*, beaucoup plus simple que TCP, car il n'ajoute aucune valeur ajoutée par rapport aux services offerts par IP. L'utilisateur n'est donc pas assuré de l'arrivée des données dans l'ordre où il les a émises, pas plus qu'il ne peut être sûr que certaines données ne seront ni perdues, ni dupliquées, puisqu'UDP ne dispose pas des mécanismes de contrôle pour vérifier tout cela. De ce fait, il n'introduit pas de délais supplémentaires dans la transmission des données entre l'émetteur et le récepteur. C'est la raison pour laquelle il est utilisé par les applications qui ne recherchent pas une grande fiabilité des données ou qui ne veulent pas assumer la lourdeur de gestion des mécanismes mis en jeu dans le mode connecté.

1.2. Format du datagramme UDP

Les messages UDP contiennent deux parties, un en-tête et des données encapsulées dans les datagrammes IP, comme les segments TCP. Le format est illustré par la figure 2.

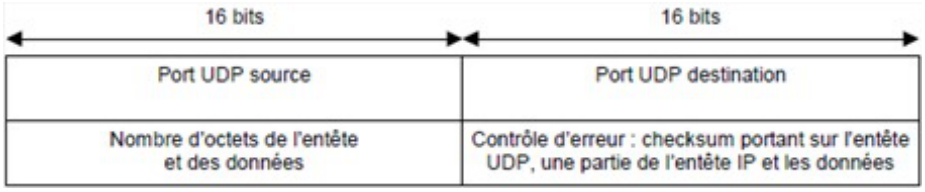

Figure 3 : format datagramme UDP

L'en-tête très simple compte quatre champs :

- Port source (16 bits). Il s'agit du numéro de port correspondant à l'application émettrice du paquet. Ce champ représente une adresse de réponse pour le destinataire.
- Port destination (16 bits). Contient le port correspondant à l'application de la machine à laquelle on s'adresse. Les ports source et destination ont évidemment la même signification que pour TCP.
- Longueur en-tête et données (16 bits). Précise la longueur totale du datagramme UDP, exprimée en octets. La longueur maximale des données transportées dans le datagramme UDP est de : 2^16 – 4*16, soit 65 472 octets.
- Contrôle d'erreur ou checksum (16 bits). Bloc de contrôle d'erreur destiné à contrôler l'intégrité de l'en-tête du datagramme UDP, comme dans TCP.

1.3. Interface entre UDP et les applications

Les processus applicatifs utilisent des sockets UDP. Leur manipulation est très simple puisque le protocole n'est pas en mode connecté : il n'y a pas de procédure de connexion et donc pas de fermeture non plus.

2. Le Protocole TCP (Transmission Control Protocol)

TCP comble les carences d'IP lorsque les applications requièrent une grande fiabilité. Ce protocole de transport, lourd et complexe, met en œuvre *la détection et la correction d'erreurs, gère le contrôle de flux et négocie les conditions du transfert des données* entre les deux extrémités de la connexion. L'entité gérée par le protocole TCP s'appelle le *segment*.

2.1. Dialogue de bout en bout

À la demande des applications qui ont besoin d'échanger des données de manière fiable, TCP ouvre une connexion et gère un dialogue. TCP n'est implanté que sur les machines des utilisateurs. Il est défini dans la RFC 793 et s'occupe de gérer et de fiabiliser les échanges, alors qu'IP est responsable de la traversée des différents réseaux.

2.2. Fonctionnalité de TCP

TCP est capable de détecter les datagrammes perdus ou dupliqués et de les remettre dans l'ordre où ils ont été émis. Ce service repose sur la numérotation et l'acquittement des données et utilise une fenêtre d'anticipation. ses principales caractéristiques sont :

Etablissement et fermeture de la connexion virtuelle ;

Segmentation et ré-assemblage des données ;

séquencement des segments (ré-séquencement si la couche réseau ne les délivre pas dans l'ordre) ;

contrôle de flux

2.3. Format d'un segment TCP

Le segment contient un en-tête de 20 octets (sans options) et un champ de données. L'en-tête du segment TCP est présenté par la figure 4 :

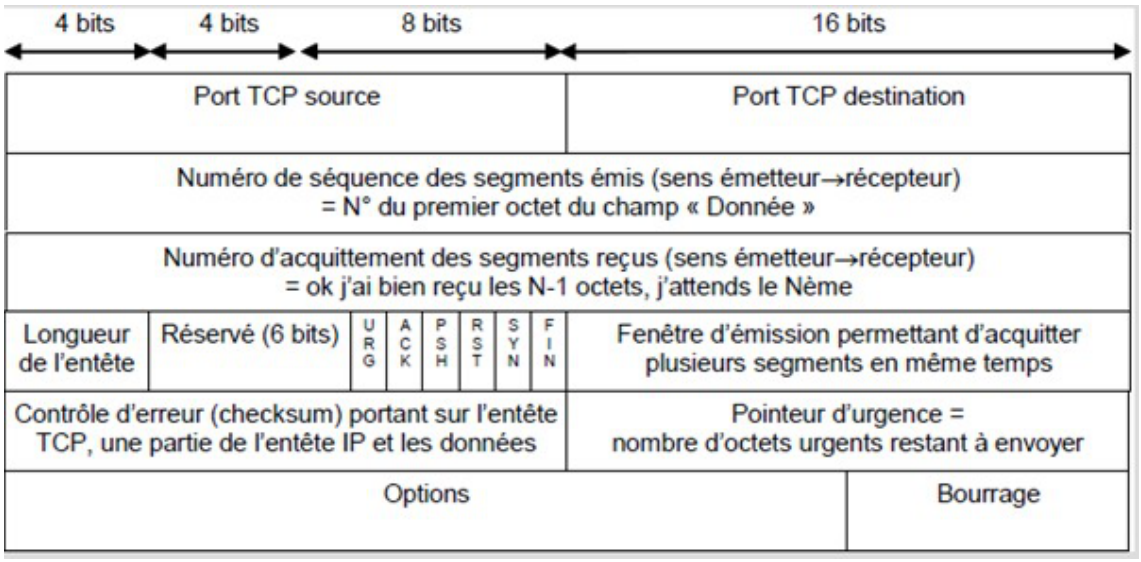

Figure 4 : format du segment TCP

Des options peuvent être négociées entre entités TCP (champ ''options''), par exemple la taille maximale des segments transportés.

Les significations de quelques champs du segment TCP sont présentées comme suit :

- Port Source (16 bits). Numéro du port utilisé par l'application en cours sur la machine source.
- Port Destination (16 bits). Numéro du port relatif à l'application en cours sur la machine de destination.
- Numéro de séquence des segments émis (32 bits). La signification de ce numéro est à interpréter selon la valeur du drapeau SYN (Synchronize). Lorsque le bit SYN est à 0, le numéro d'ordre est celui du premier octet de données du segment en cours (par rapport à tous les octets du flot de données transportées). Lorsqu'il est à 1, le numéro d'ordre est le numéro initial, celui du premier octet du flux de données qui sera transmis (Initial Sequence Number). Celui-ci est tiré au sort, plutôt que de commencer systématiquement à 0.
- Numéro d'acquittement des segments reçus (32 bits). Numéro d'ordre du dernier octet reçu par le récepteur (par rapport à tous les octets du flot de données reçues).
- Longueur en-tête (4 bits). Il permet de repérer le début des données dans le segment. Ce décalage est essentiel, car il est possible que l'en-tête contienne un champ d'options de taille variable. Un en-tête sans option contient 20 octets, donc le champ longueur contient la valeur 5, l'unité étant le mot de 32 bits (soit 4 octets).

3. Connexion TCP

3.1. Ouverture d'une connexion

TCP est un protocole qui fonctionne en mode connecté. Le périphérique A ouvre une demande de connexion qui est transmise à la couche transport en envoyant un premier segment, parfois appelé séquence de synchronisation. Ce segment contient en particulier le numéro de séquence initial (le numéro du premier octet émis) et le drapeau SYN est mis à 1. Le périphérique B répond par un acquittement comprenant le numéro du premier octet attendu (celui qu'il a reçu + 1) et son propre numéro de séquence initial (pour référencer les octets de données du serveur vers le client). Dans ce segment de réponse, le périphérique B a positionné les drapeaux SYN et ACK à 1. Enfin, le périphérique A acquitte la réponse du serveur en envoyant un numéro d'acquittement égal au numéro de séquence envoyé par le périphérique B + 1. Dans ce troisième message, seul le drapeau ACK est mis à 1. L'ensemble des trois segments correspond à l'ouverture de la connexion, illustrée à la figure 5.

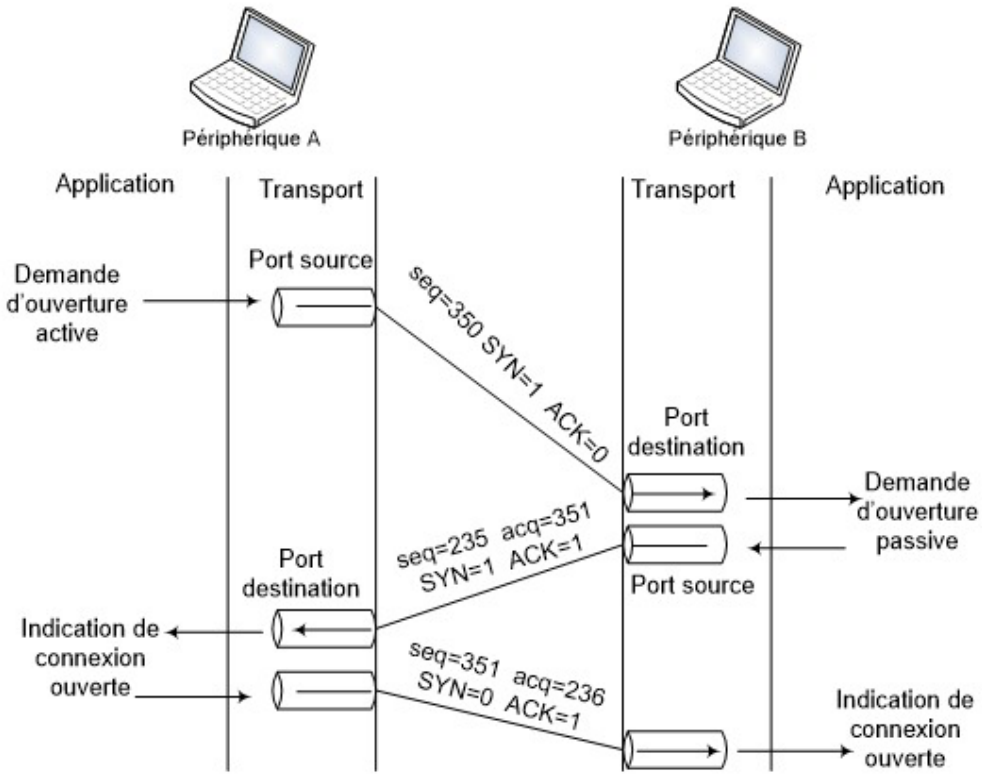

Figure 5 : ouverture connexion TCP

Après la dernière phase, le transfert des données peut commencer.

Remarque

Le numéro de séquence initial est généré aléatoirement. Après chaque émission de segment, il est incrémenté du nombre d'octets transportés dans ce segment.

3.2. Transfert des données

Le transfert de données commence avec avec les séquences en cours. Le contrôle de flux est réalisé dans les deux sens par les numéros d'acquittement (le bit ACK est à 1). Chaque accusé de réception indique le nombre d'octets correctement reçus.

Dans l'exemple de la figure 6, le numéro d'acquittement 362 renvoyé par le périphérique B indique à l'émetteur que les 10 octets de 352 à 361 ont été reçus et que les prochains octets, à partir du numéro de séquence 362, peuvent être transmis.

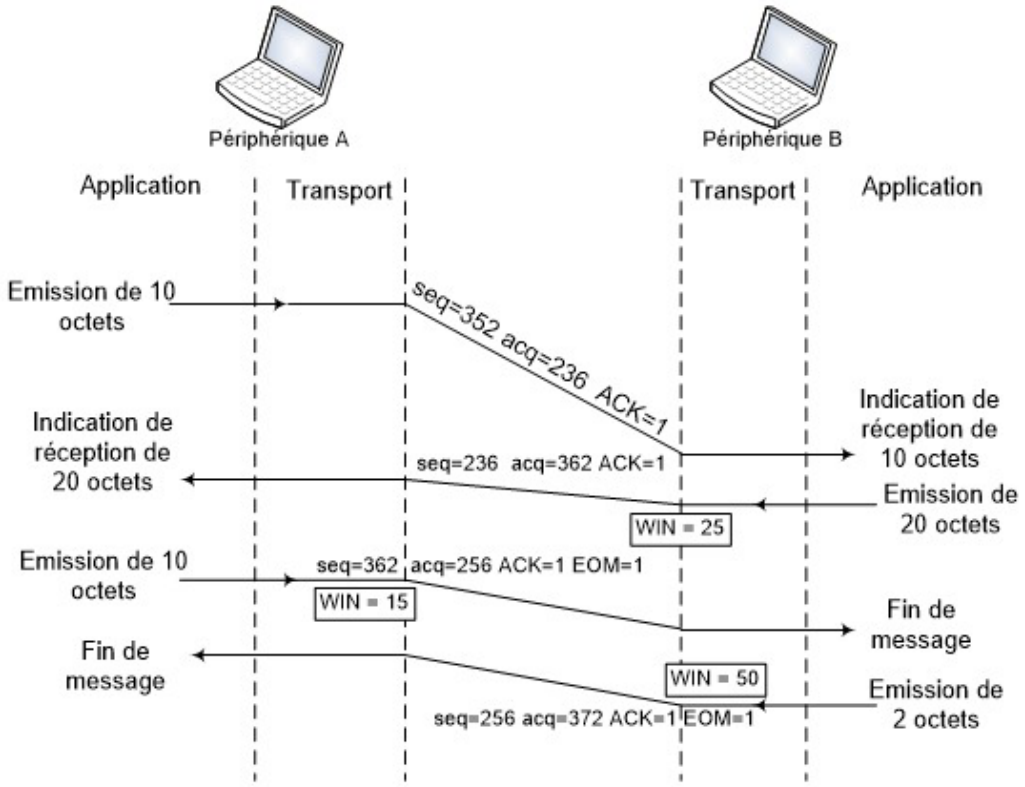

Figure 6 : Transfert de fichier

3.3. Fermeture de connexion

La demande de fermeture d'une connexion est initiée lorsqu'un des périphériques reçoit un en-tête TCP dont le bit FIN est à 1. dans l'exemple de la figure 7, après acquittement par le périphérique B de la demande de fermeture, la connexion est à demi fermée. cet état intermédiaire permet à des données en transit d'arriver jusqu'au périphérique A avant que B n'effectue à sont tour une demande de fermeture suivie d'une confirmation.

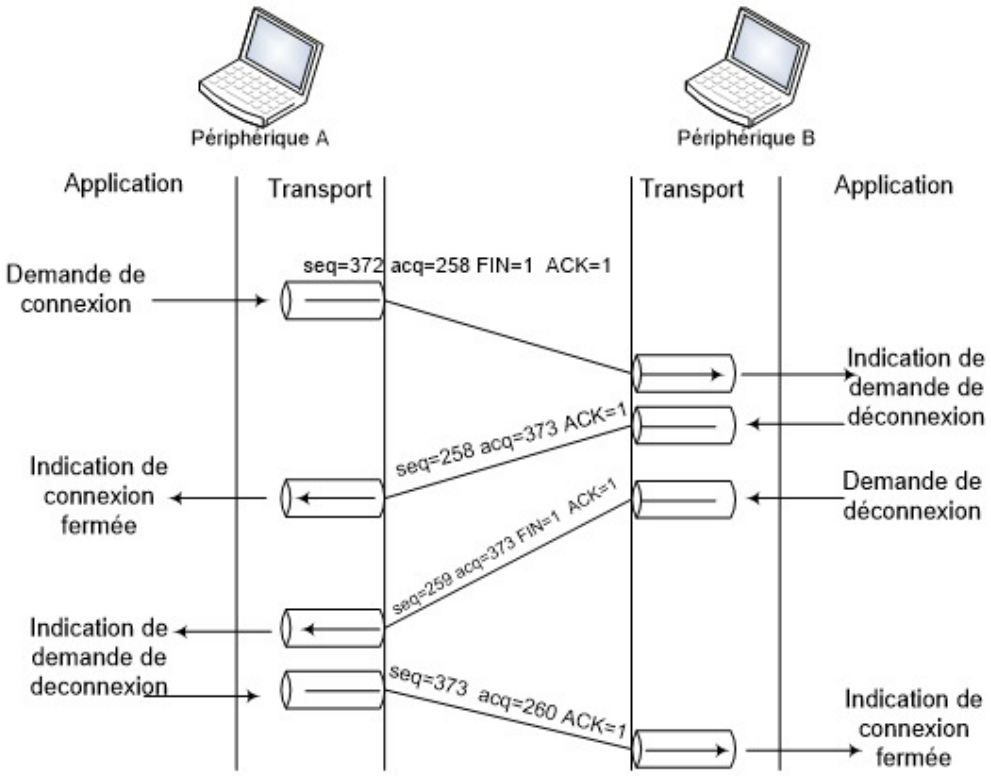

Figure 7 : Fermeture de connexion

4. Contrôle de flux et de congestion

4.1. Le contrôle de flux

Le contrôle de flux est réalisé par rapport au récepteur, elle permet d'adapter le rythme de l'émetteur aux capacités du récepteur et aux performances du réseau. L'émetteur adapte le nombre de segments envoyés à la taille du tampon de réception. ce contrôle de bout en bout est réalisé grâce :

- aux numéros d'acquittement ;
- à la fenêtre window transmise par le récepteur.

Quand le récepteur veut ralentir la source, il peut ralentir l'envoi des acquittements et/ou envoyer une valeur de fenêtre faible ou nulle.

4.2. Le contrôle de congestion

Le contrôle de congestion est réalisé par rapport au réseau. L'émetteur adapte le débit des données envoyées à la bande passante instantanée du réseau. La bande passante peut tendre vers zéro (0) lorsqu'un routeur intermédiaire est saturé.

Ce contrôle est géré côté émetteur grâce au dimensionnement dynamique d'une fenêtre de congestion *cwnd* (vongestion windows) dont la valeur augmente ou diminue suivant le nombre et la valeur des acquittements reçus.

Exemple

Si on a envoyé des segments de 1 024, 2 048, 4 096 octets correctement mais qu'un segment de 8 192 octets provoque une expiration de temporisation, on évitera la congestion en dimensionnant la fenêtre à 4 096 octets et on n'enverra pas de segments de taille supérieure à cette valeur, quelle que soit la taille de la fenêtre de réception.

5. Exercice : Exercices

[Solution n°2 p 15]

Exercice : Exercice 1

Le protocole TCP dans son fonctionnement, met en oeuvre :

- [ ] la détection et la correction d'erreurs
- [ ] le contrôle de flux
- [ ] les conditions de transfert

Exercice : Exercice 2

Le protocole TCP est un protocole qui gère la _____ des échanges entre deux applications contrairement au protocole UDP.

Exercice : Exercice 3

L'en-tête d'un segment TCP est constitué de :

- ◯ 20 octets
- ◯ 24 octets

Exercice : Exercice 4

Parmi les champs ci-dessous, spécifier les champs communs aux protocoles UDP et TCP.

- [ ] Port source
- [ ] Port destination
- [ ] Longueur en-tête
- [ ] Numéro de séquence
- [ ] Numéro d'acquittement
- [ ] Contrôle d'erreur

Exercice : Exercice 5

Une application A initie une demande de connexion auprès d'une application B. Spécifier le segment du périphérique qui fait la demande de connexion.

- ◯ seq = 250, SYN - 1, ACK = 0
- ◯ seq = 251, SYN = 1, ACK = 1

Conclusion

Deux protocoles de transport sont utilisés dans l'architecture TCP/IP. Le premier, TCP, est un protocole complet, destiné à pallier toutes les défaillances de l'interconnexion de réseaux. C'est un protocole en mode connecté qui met en œuvre une détection et une correction d'erreurs, un contrôle de séquence, de flux et de congestion. Il est de ce fait complexe et lourd à gérer. TCP est indispensable pour toutes les applications qui transfèrent de grandes quantités d'informations et qui ont besoin de fiabilité dans les échanges de données.

UDP, lui, utilise le protocole IP pour acheminer un message d'un ordinateur à un autre, sans aucune valeur ajoutée (pas de connexion, pas de contrôle d'erreur, de contrôle de flux ni de contrôle de séquence) par rapport aux services rendus par IP.

Il convient aux applications de type requête/réponse simples ou ayant des contraintes temporelles fortes.

Solutions des exercices

> Solution n°1

Exercice p. 7

Exercice 1

L'interface de programmation entre un protocole de transport et une application est le socket. Il est constitué d'une adresse IP et d'un numéro de port

Exercice 2

☑ le protocole de transport

☑ l'adresse IP locale

☑ le numéro de port local

☑ l'adresse IP distante

☑ le numéro de port distant

Exercice 3

Affecter à chaque numéro de port, l'application correspondante.

Numéro de Port	Application
21	FTP
23	Telnet
80	HTTP
53	DNS

> Solution n°2

Exercice p. 13

Exercice 1

☑ la détection et la correction d'erreurs

☑ le contrôle de flux

☑ les conditions de transfert

Exercice 2

Le protocole TCP est un protocole qui gère la fiabilité des échanges entre deux applications contrairement au protocole UDP.

Exercice 3

- (•) 20 octets
- ( ) 24 octets

Exercice 4

- [x] Port source
- [x] Port destination
- [x] Longueur en-tête
- [ ] Numéro de séquence
- [ ] Numéro d'acquittement
- [x] Contrôle d'erreur

Exercice 5

- (•) seq = 250, SYN - 1, ACK = 0
- ( ) seq = 251, SYN = 1, ACK = 1

Bibliographie

Guy Pujolle, Initiation aux Réseaux cours et exercices, Éditions Eyrolles 2001

Guy Pujolle, Initiation aux réseaux cours et exercices, Editions Eyrolles 2001

Andrew Tanenbaum, Réseaux, 4è édition, Nouveaux Horizons, ISBN 978-2-915236-75-0

Jean-Luc Montagnier, Réseaux s'entreprise par la pratique, 2è Editions, Editions Eyrolles, ISBN 2-212-11258-0

Danièle DROMARD, Dominique SERET, Architecture des réseaux Synthèses de cours et exercices corrigés, collection Synthex, ISBN 978-2-7440-7385-4, 2009 Pearson Education France

Claude Servin, RESEAUX & TELECOMS cours et exercices corrigés, 3è édition DUNOD, ISBN 978 2 10 052626 0