**M2-CCI Enseignement de Réseaux – Compte-Rendu du TP n.3**

*Groupe BAJA Sara – BOLLOLI Marco*

**Premesse:**

Contrairement aux deux premiers comptes-rendus, celui-ci présentera moins de figures. La raison est que la récupération des données est devenue plus difficile sur les ordinateurs, car la reconnaissance d’une clé USB ne semble plus fonctionner, et l’établissement d’une connexion internet a été aléatoire, car le browser refuse souvent de nous laisser connecter à internet car la connexion n’est pas sécurisée sur le PC. Cela ne dépend en rien de notre volonté.

**Introduction :**

Ce compte-rendu a pour objet les manipulations effectuées dans le TP n.2, et les observations et les calculs qui y étaient associés. De suite, les étapes et les questions seront présentées avec la numérotation adoptée dans l’énoncé du TP pour des raisons de clarté.

La structuration du réseau a été faite de la manière usuelle (Cf. Figure 1), mais seulement deux machines ont été utilisées (PC2 et PC3 généralement).

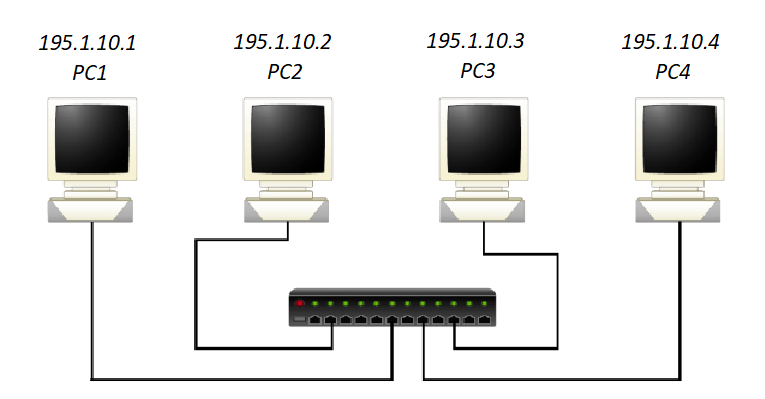


Figure 1

**2.1 Le Protocole UDP**

Sur chacune des deux machines connectées au réseau, nous avons lancé l’interface Socklab en choisissant UDP comme protocole pour créer une socket UDP afin de permettre l’échange des messages.

|  |  |
| --- | --- |
| Liste commandes exécutées sur la machine 1 | Liste commandes exécutées sur la machine 2 |
| # socklab udp  socklab-udp> sock  socklab-udp> sendto <Id de Socket>< nom de machine><numero de port> | # socklab udp  socklab-udp> sock  socklab-udp> recvfrom <Id de Socket> <nb d’octects> |

Le protocole UDP est un protocole non fiable et sans connexion, il permet à une application d’envoyer des messages à une autre application avec un minimum de fonctionnalités.

Les éléments pertinents de la capture ci-dessus sont :

1. Le port source et le port destination : ils permettent de référencer les processus qui s'exécutent sur la machine source (locale) et la machine destination (distante). Ils sont codé chacun sur 16 bits.

* La longueur totale du message : elle représente les “données + entête”. Elle est codée sur 16 bits. Son unité est l’octet.

**Question 2 : Le numéro de port doit etre le meme sur les deux machines ? Peut-on utiliser deux fois le meme port sur une machine ?**

Le port ne doit pas etre forcement le meme sur les deux machines. Coté émetteur, on ne peut pas créer de socket avec le meme port ; coté récepteur, on ne peut pas faire un bind de plusieurs socket sur le meme port.

**Question 3**

Une socket est un point d’accès au réseau pour couches transport. Elle peut être réalisée par différentes familles de protocole et créée dynamiquement par un processus. Lorsque l'on crée une socket, on doit préciser la famille de protocoles de communication que l'on désire utiliser.

L'identificateur de socket permet de distinguer une socket des autres sur une machine. Une machine ne connait pas le numéro de socket de la machine distante. Un numéro de socket est local à la machine et ne peut donc pas être connu par une machine distante.

**Question 4**

L’identificateur de la socket est un indice locale à la table des sockets locale à la machine.

**Question 5**

L’en-tête du paquet UDP se présente comme suit:

|  |  |
| --- | --- |
| Port source (16 bits) | port destination (16 bits) |
| Longueur (16 bits) | checksum (16 bits) |

Voici la signification des différents champs:

* Port Source: il s'agit du numéro de [port](http://www.commentcamarche.net/contents/528-port-ports-tcp-ip) correspondant à l'application émettrice du paquet UDP.
* Port Destination: Ce champ contient le port correspondant à l'application de la machine destinataire à laquelle on s'adresse.
* Longueur: Ce champ précise la longueur totale du paquet avec l’en-tête comprise.
* Checksum: Il s'agit d'une [somme de contrôle](http://www.commentcamarche.net/contents/97-le-controle-des-erreurs) réalisée de telle façon à pouvoir contrôler l'intégrité du segment.

A l'émission d’un paquet, les informations que UDP passe à IP sont l’en-tete du paquet UDP et les données utiles.

A l’émission d’un paquet, l’UDP transmet à l’IP l'adresse internet, et le numéro de port du processus concernée par l’échange sur la machine distante.

**Question 6: fonctionnement du protocole UDP**

Le protocole UDP est un protocole de la couche transport. Ce protocole n’est pas orienté connexion puisqu’il ne propose pas de contrôle d’erreurs. Un simple checksum permet de s’assurer de l’intégrité du paquet reçu. Ce protocole est plutôt destiné à des applications temps-réel plutôt qu’à des applications pouvant être impactées par des pertes de paquets.

L’envoi des données selon le protocole UDP ne nécessite pas l'établissement d’une connexion entre les machines qui échangent. Voici quelques observations faites lors des manipulations :

* On ne peut pas recevoir les données, avant de les avoir envoyées.
* Quand on envoie plusieurs paquets avant de demander la réception, lorsqu’on demande la réception, les paquets arrivent les uns après les autres dans l’ordre dans lequel ils ont été envoyés. Le protocole UDP respecte donc l'ordre d’envoi.
* Quand on déclenche la réception avant l’envoi, la socket reste en attente de données. A la réception, lorsqu’on indique un nombre d’octets supérieur ou égal à celui du paquet envoyé, le message est entièrement reçu. Dans le cas inverse, nous perdons une partie du message.
* Il est à noter aussi que lorsqu’on débranche la machine réceptrice et qu’on l’envoi des paquets, la machine émettrice continue à envoyer, car elle n’est pas au courant que la machine distante est déconnectée du réseau. Le fonctionnement du protocole UDP est similaire à celui de la poste. Il garantit seulement l’envoi des données mais aucune garantie quant à la réception des données, ce qui fait que les données peuvent êtres perdues ou modifiées en cours d’envoi sans que l’on sache.
* Lors qu’on envoie plusieurs paquets de tailles importantes vers une machine de façon à saturer le récepteur, à remplir
* Lors qu’on envoie sur un port inexistant, le protocole UDP envoie les données. C’est le protocole ICMP qui s’occupe d’informer la machine emettrice que le port n’existe pas.

1. le protocole TCP

Sur chacune des deux machines connectées au réseau, nous avons lancé l’interface Socklab en choisissant TCP comme protocole pour créer une socket TCP.

|  |  |
| --- | --- |
| Liste commandes exécutées sur machine 1 | Liste commandes exécutées sur machine 2 |
| # socklab tcp  socklab-tcp> passive  socklab-tcp> accept | # socklab tcp  socklab-tcp> active  socklab-tcp> connect <nom de machine> <numero de port> |

**2.2 Le Protocole TCP**

**Question 7 : pourquoi la socket est dite active ou passive ?**

La première socket est dite active car elle effectue la demande d’établissement de connexion à l’aide du protocole TCP. La seconde est dite passive, car elle reçoit une demande de connexion et confirme la réception de cette demande. La socket active a un rôle d’écriture tandis que la socket passive à un rôle de lecture.

La socket active: permet de demander à établir une connexion avec un serveur distant.

La socket passive: permet de rester en écoute et traiter les demandes des connexions des clients.

**Question 8 : pourquoi deux socket existent du coté passif après avoir fait accept ?**

Car une socket dediée au client est crée par la socket du serveur après avoir accepté la connection. La première socket du serveur sert juste pour l’acceptation de toutes les demandes de connection.

**Question 9 : analysez les echanges lors d’une connection**

Le schéma ci-dessous illustre l’échange des paquets les paquets :

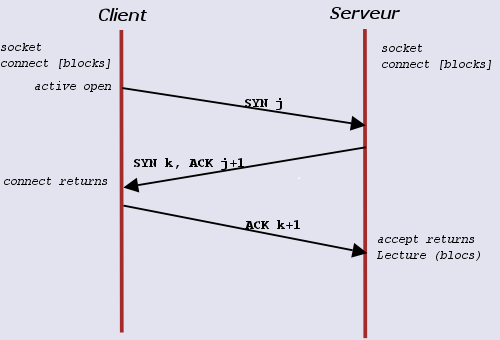
****

Figure 2

On remarque trois paquets différents:

* Envoi d'un paquet avec le flag SYN par la socket client pour demander la connexion.
* Confirmation de l’établissement de la connexion par la socket serveur( flags SYN et ACK)
* La socket client renvoie un accusé de réception relatif à l’etablissement de la connexion (le flag ACK).

Voici les informatiosn obtenues par la capture d’écran sous Wireshark:

Nous avons les trois premiers paquets qui permettent l’ouverture de la connexion du protocol TCP :

-le premier paquet contient le flag SYN par le socket client pour demander la connexion.

- Le paquet 2 pour la Confirmation de l’établissement de la connexion du coté socket serveur avec les deux flags SYN et ACK.

- Le socket client renvoie un accusé de réception relatif à l’établissement de la connexion.

**Question 10 : Quel est le role du flag SYN…. ?**

Lors de la connexion, le premier paquet SYN envoyé par la socket active sert à demander l’établissement de la connexion avec la socket passive de la machine distante. Le paquet SYN,ACK en réponse permet à la socket passive d’accepter la demande d’établissement de connexion tout en précisant que cette demande à bien été reçue. A la réception de ce paquet la socket active envoie un paquet ACK précisant qu’elle a bien reçue la confirmation envoyé préalablement. Le flag SYN sert a identifier les paquets utiles à l’établissement d’une connexion. Les numéros de séquences servent à identifier une connexion. Ici, les trois paquets nécessaires à l’établissement de la connexion ont tous le même numéro de séquence. Chaque système utilise son propre algorithme pour générer des numéros de séquence initiaux. Les numéros d’acquittement servent à signaler à la socket distante la bonne réception du paquet précédent. Lors de l’envoie du premier paquet, le numéro d’acquittement est égale à 0 car aucun paquet n’a été reçu avant. Le premier paquet SYN,ACK envoyé a un numéro d’acquittement égal à 1 ce qui correspond au numéro de séquence plus 1. Enfin, le dernier paquet ACK envoyé a un numéro d’acquittement égal à 1 ce qui correspond aussi au numéro de séquence plus un car la socket passive n’a pas envoyée d’autre donnée. Le champs WINDOW permet de préciser le nombre d’octets que le récepteur souhaite recevoir sans accusé de réception. A ne pas confondre avec le paramètre WINDOW scale qui permet de connaître l’état des buffers de réception. La taille maximale des segments autorisés à recevoir est aussi transmise dans les paramètres option des paquets TCP.

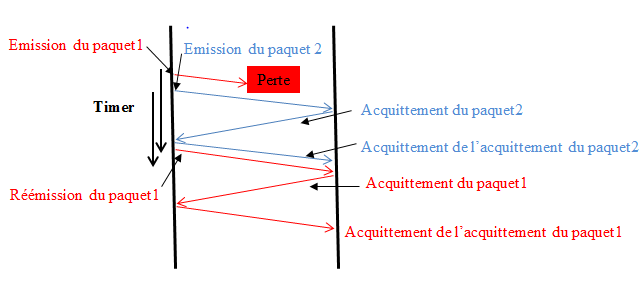
****

Figure 3

**Question 11 : Expliquez ce qu’il se passe au moment de la primitive des sockets "accept". Essayer de faire "accept" après "connect", que se passe-t-il ?**

Lorsqu’on fait un accept avant un connect la connexion est établie comme décrit précédemment. Le connect est fait avant le accept, la connexion est établie sur une socket qui écoute la demande de connexion. Après le accept, une nouvelle socket est crée dédiée au client.

**Question 13 :****Qu’est ce qui identifie réellement une connexion, c’est-à-dire, comment TCP associe les messages reçus aux différentes connexions en cours ?**

On observe que malgré le fait que plusieurs connexions soient ouvertes, le port d’écriture est différent mais le port d’écoute est toujours le même. On en conclue que c’est le couple port d’écoute - port d’écriture qui identifie réellement une connexion.

**Question 14 : Retrouvez la signification de l’état de la connexion dans le résultat de commande netstat.**

Sur le serveur, avant le accept, la connexion n’est pas établie, le serveur écoute la demande de connexion sur un port : état LISTEN. Après le accept la connexion est établie, une nouvelle socket dédiée au client est créée l’état de connexion est ESTABLISHED. Côté client aucun port en écoute n’est visible, on observe juste les états ESTABLISHED.

**Question 15 : Faites une demande de connexion vers un port inexistant. Expliquez ce qu’il se passe.**

La procédure de demande de connexion est initialisée avec l’envoi d’un paquet TCP flag SYN par la socket active comme on a pu l’observer précédemment. La socket passive étant inexistante (aucune socket n’a été initialisée pour le port auquel a été adressé le paquet TCP SYN), un paquet ACK,RST connexion refusée est renvoyé. Celui-ci permet à la machine de s’acquitter de la demande de connexion et signale à la socket active que la connexion ne pourra pas être établie grâce au flag Reset. Un schéma est présenté ci-dessous.

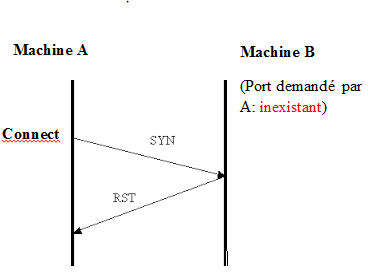
****

Figure 4

**2.2.2 Etude du séquencement et de la récupération d'erreur**

Une connexion TCP est établie entre deux machines distantes grâce aux commandes accept et connect de l’utilitaire Socklab. Des données sont échangées entre les deux machines grâce aux commandes read et write. A la place du message normal, la commande #5000 permet d’envoyer un message de 5000 octets.

**Question 16 : Analysez les paquets engendrés par le transport des données. Expliquez le rôle des champs SEQUENCE NUMBER et ACK NUMBER dans l’entête des paquets TCP.**

Étant donné que la taille du paquet est supérieure à 1500 octets, ce qui est la taille maximale qu’on peut envoyer sur un réseau Ethernet, on remarque que le paquet est découpé en plusieurs petits paquets. La socket active envoie un premier paquet avec le flags push PSH. On imagine qu’il pousse les données dans le canal de connexion entre les sockets. La socket passive répond à l’aide de packet TCP ACK afin de s’acquitter des données reçues. Lors des prochains envois, les paquets TCP on le flag PSH,ACK. Ceci permet à la socket active de s’acquitter du paquet ACK envoyé par la socket passive. Le champ SEQUENCE NUMBER a déjà été initialisé lors de la connexion des sockets. Il sert maintenant à informer jusqu’à quel octet des données ont été envoyées. Lors de l’envoi d’un paquet PSH de 1500 octets, le champ SEQUENCE NUMBER sera égal au SEQUENCE NUMBER initial plus 1500 octets. Comme pour la connexion des sockets, le champ ACK NUMBER permet de s’acquitter du nombre d’octets reçu et correspond au dernier SEQEUNCE NUMBER reçu plus 1 octet. Par exemple, si une socket active écrit des données vers une socket passive, le champ SEQUENCE NUMBER de la socket active augmentera au fur et à mesure que les données sont envoyées. La socket passive s’acquitte de la réception des données en renvoyant des paquets sans donnée mais avec un champ ACK NUMBER correspondant au numéro de la séquence plus 1. Puisque la socket passive n’envoie pas de donnée, son champ SEQUENCE NUMBER n’est pas incrémenté. La socket active confirmera la réception du paquet de la socket passive lors d’un nouvel envoi de données PSH en rajoutant un flag ACK et un champ ACK NUMBER égal au SEQUENCE NUMBER de la socket passive plus un.

**Un exemple de capture est présenté dans la Figure 5.**

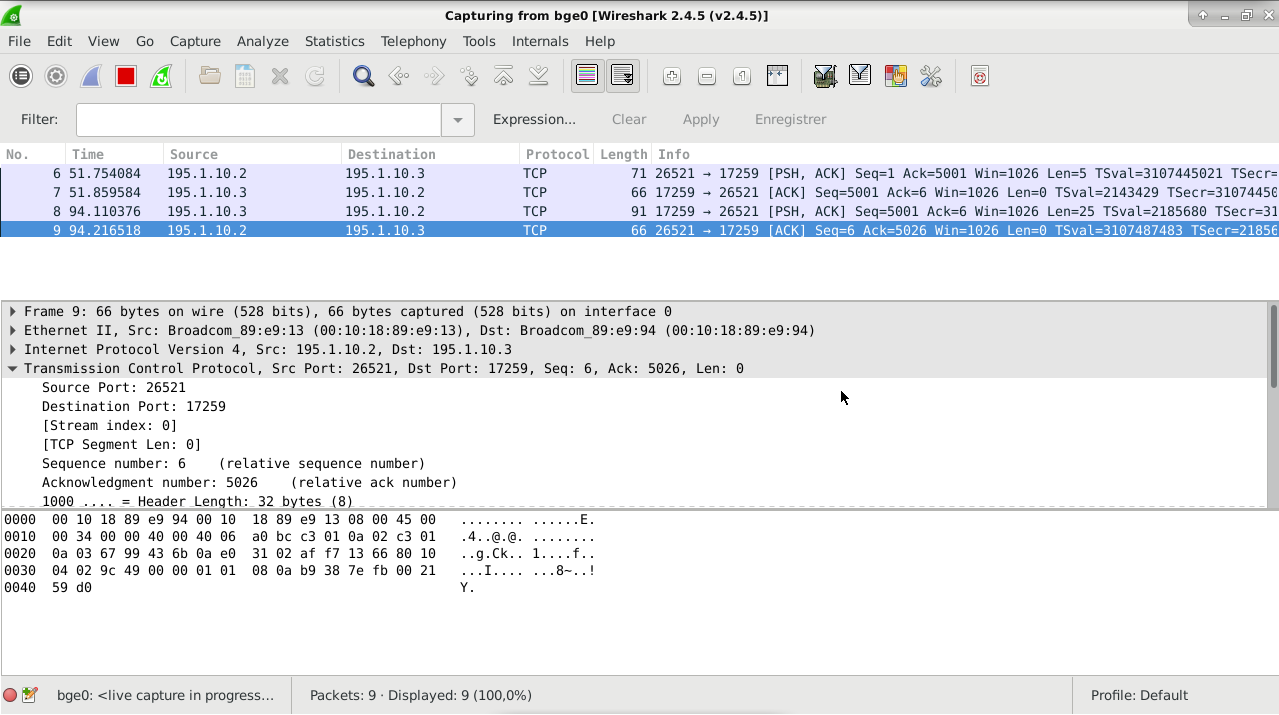
****

Figure 5

**Question 17: Il y a-t-il toujours un acquittement par paquet de donnée ? Pourquoi ?**

Non. Il peut avoir plusieurs acquittements par paquet de données. Parce que d’une part l’envoi selon le protocole TCP se fait par octet. Les données sont envoyées indépendamment donc elles peuvent être acquittées ensemble ou séparées. D’autre part le nombre d’octet acquitté dépend de la taille disponible sur le buffer de reception.

**Question 15 : Lorsqu’on tente d’envoyer un message vers une machine distance après l’avoir débranchée, l’envoie échoue. En effet, le protocole TCP envoie des paquets ARP request.**

**Contrairement à l’UDP, TCP s’assure d’abord qu’une connexion est établie entre les deux machines avant d’envoyer les paquets.**

**Question 16 : Lorsque la taille du buffer d’émission diminue, le débit applicatif diminue.**

**Envoie de paquets de 3000 octets, attend l’acquittement avant d’envoyer les autres paquets.**

**Question 17: fonctionnalité du buffer d’émission de TCP**

**C’est le buffer d’émission qui gère l’envoi des paquets. Mais son fonctionnement est principalement lié à l’état d’avancement de la fenêtre du contrôle de flux par rapport à celle de la récupération d'erreur. Lorsqu’une fenêtre est moins avancée que l’autre l’émetteur reste bloquer sur celle qui est moins avancée.**

**Question 18 :**

**Question 21: un buffer d'émission de taille non adéquate a la latence du réseau a un impact direct sur la vitesse à laquelle la fenêtre de congestion augmente et sur la taille qu’elle atteint et par conséquent le débit. Une latence élevée, entraîne une augmentation du temps d’attente de l'expéditeur, ce qui réduit la vitesse à laquelle le débit de la session TCP augmente.**

**Question 22 : Tous les paquets envoyés ont été reçus.**

**Question 23 :**

**La valeur de WIN est :**

* **10000 octets au moment de l’ouverture de la connexion**
* **0 après acquittement**

**L’émetteur ne peut pas émettre plus d’octets que la taille du buffer de réception. Il reste bloquer jusqu’à la libération du tampon par une lecture de l’application côté récepteur**

**Question 26: Il n’est pas intéressant d’avoir un buffer d’émission plus grand que le buffer de réception, car si le récepteur suit le rythme d’émission on aurait des blocages permanents en émission et donc des performances moindres. Dans le cas où le rythme du récepteur est sporadique, il serait plutôt intéressant d’avoir un tampon en réception plus grand qu’en émission ou donner la même taille pour les deux.**

**Question 28: on peut plus envoyer des paquets**

**Conclusion :**

Ce TP nous permis d’avoir des notions de base sur le fonctionnement d’un réseau Ethernet. Nous avons pu constater quelles sont les différentes composantes du temps de transmission des données d’une machine à une autre, et nous avons-nous initier à la gestion d’un réseau avec plusieurs machines connectées ensemble.