Université de Savoie
Initiation aux réseaux IP

Travaux Pratiques

Introduction aux réseaux IP

Sylvain MONTAGNY
sylvain.montagny@univ-savoie.fr
Bâtiment chablais, bureau 13
04 79 75 86 86

TP1 : Analyse de trame, routage
TP2 : Simulation sur Packet Tracer
TP3 : Ethernet embarqué
TP4 : Programmation Client / Serveur par socket

Retrouver tous les documents de Cours/TD/TP sur le site
www.master-electronique.com

Université de Savoie

TP1

Analyse de trames et routage

Fichiers fournis : [www.master-electronique.com] > Espace étudiant

I. Analyse des paquets avec LANWATCH

LANWATCH comme son nom l'indique est un logiciel destiné à observer les réseaux locaux et notamment les réseaux « Ethernet ». Nous disposons de la version de démonstration qui met à notre disposition un exemple de capture d'un réseau particulier.
Les manipulations qui vont suivre vont permettre de mettre en évidence :

- l'encapsulation
- la structure d'une trame Ethernet
- la structure en couche du modèle OSI

Lancer la version de démonstration de lanwatch.exe Ouvrir le fichier « DemoDump.dmp », puis cliquez sur View > Examine afin de visualiser l'ensemble des trames. Cliquer sur View->View All Panels (ctrl A) pour afficher l'ensemble des fenêtres pour l'étude des trames.

1. Protocole Ethernet

Q1. Sur la trame 405.686, quels octets représentent le nom du constructeur de la carte? Que représentent les autres octets ? Quelle est la particularité d'une adresse Ethernet ?

Q2. Sur la trame 405.765, a quoi correspond l'étoile (*) dans le champs correspondant à l'adresse MAC destination ?

Q3. LANWATCH utilise différentes couleurs suivant le protocole. Comment fait-il pour connaître le protocole utilisé au dessus de la couche Ethernet. Donner les cas pour le protocole ARP et IP.

2. Protocole ARP

Etude des trames 405.765 et 405.934 :

Q4. Retrouvez les champs adresse Ethernet et IP dans la trame en hexadécimal.

Q5. Sachant que la taille d'un paquet ARP est de 28 octets, que remarquez-vous dans la trame hexadécimale?

Q6. LANWATCH ne fait pas l'affichage du champ CRC Ethernet, déduisez-en la longueur réelle de la trame Ethernet transportant des données du protocole ARP et justifier cette longueur.

3. Protocole de couche transport

Q7. Quels sont les deux protocoles de couche transport.que vous pouvez trouver dans les trames reçus.

Q8. Comment LANWATCH parvient t il à savoir le protocole utilisé pour la couche transport afin de gérer son affichage par couleur. Donner les cas pour les deux protocoles de transport cités ci-dessus.

II. Analyse du réseau local

Redémarrer votre machine sous Linux.

- Login : **root**
- Password : **root**

Wireshark est un outil puissant d'analyse des trames circulant sur le réseau > Lancer Wireshark.

Faire un test d'acquisition de trames circulant sur votre réseau pour vérifier que tout fonctionne. **Capture>Interface>Start**

1. Configuration de votre machine :

Q9. Déterminez l'adresse Ethernet et l'adresse IP de votre machine (ifconfig –a). Visualisez la table ARP de votre machine (arp –a).

2. Protocole ARP

Q10. En vérifiant au préalable que votre table ne possède pas l'hôte distante, analysez les trames générées par le test de la communication avec une autre machine (ping). Visualiser la table ARP après ce test. Détaillez le fonctionnement du protocole ARP lors du test ci-dessus.

Note : La suppression d'une entrée dans la table ARP se fait avec la commande :
arp –d @MAC

3. Confidentialité des données circulant sur Ethernet

Nous allons effectuer une acquisition de trames lors d'une consultation d'une boite mail en utilisant un logiciel client POP (Outlook sous Windows, client POP KDE sous Linux, ou d'autres…). Vous devez au préalable **configurer le client mail** en lui indiquant **le serveur mail** à contacter ainsi que le login utilisateur et le mot de passe. Il est important que vous cochiez l'option de **laisser les messages sur le serveur** afin que vous tout le monde puisse télécharger les messages.

Vérifier tout d'abord l'existence de ces comptes sur les serveurs de messagerie puis configurer le client mail de votre choix pour récupérer les emails. Vous ferez une capture de trame pour les deux essais suivants.

Q11. Pour quel compte a-t-on une défaillance en terme de sécurité, expliquer pourquoi ?

Essai avec un compte laposte.net :
Adresse mail : info324.univ@gmail.com
Password : etudiant
POP : pop.lapsote.net sans chiffrement
SMTP : smtp.laposte.net sans chiffrement

Essai avec un compte Gmail :
Adresse mail : info324.univ@gmail.com
Password : etudiant
POP : pop.gmail.com SSL
SMTP : smtp.gmail.com TLS

III. Analyse de la configuration IP :

1. Votre configuration IP

Consulter la configuration de votre machine avec la commande **ifconfig –a** .

Q12. Donnez le détail du plan d'adressage de votre interface qui est connectée au réseau local : sous-réseaux, plage d'adresse des machines possible, adresse de broadcast.

Q13. Dessiner le réseau auquel vous appartenez sur votre feuille en notant les adresses IP de vos plus proches voisins.

2. Test de communication entres machines

Q14. A l'aide de la commande "ping", testez la communication entre votre machine et les diverses machines de la salle (ping s'arrête par Ctrl-c).

- Essayez d'abord **ping *adresse-IP-machine***. Noter le résultat obtenu.
- Essayez ensuite **ping *nom-de-machine***. Noter le résultat obtenu.

Q15. Essayez ensuite les mêmes manipulations vers des machines du réseau Internet: en utilisant des noms Internet (ex. www.yahoo.fr), puis avec les adresses IP de ces machines.

3. Correspondance nom-machine >> adresse IP :

Il existe deux moyens pour effectuer la résolution d'un nom en adresse IP ou l'inverse.

3.1. Résolution locale

Le premier est un moyen local et consiste à définir un fichier de correspondance sur chaque machine. Ainsi chaque machine sera capable de résoudre les noms qui sont définis dans son fichier. Sur une machine de type Unix, ce fichier est **/etc/hosts**. Evidemment le fichier /etc/hosts ne permet pas de faire la résolution de nom de l'ensemble des machines du réseau Internet. Il ne peut comporter que les noms des machines proches.

Q16. Quelles est le contenu de votre fichier ? Expliquer les conséquences et faites des essais en associant des noms à des adresses IP.

3.2. Résolution par requête DNS

Le second moyen de résolution des adresses est un moyen global qui permet de résoudre n'importe quel nom de machine du réseau Internet. Il repose sur le principe DNS (Domain Name Service) qui consiste à définir des noms constitués de noms de domaines et sous-domaines. Chaque domaine est géré par un serveur DNS qui connaît les correspondances pour l'ensemble des machines de son domaine. Par contre chaque résolution nécessite une requête au serveur DNS.
Le fichier système qui indique l'adresse du serveur DNS est le fichier /etc/resolv.conf.

Q17. Donner l'adresse du serveur DNS de l'université.

Renommez le fichier /etc/resolv.conf existe, en l'appelant "dns.conf" .

Q18. Testez de nouveau la communication par ping. Conclusion ? Recréez ensuite le fichier resolv.conf.

Q19. Un serveur DNS est-il indispensable pour échanger des données entres 2 machines précises d'Internet ? Pour naviguer sur le web ?

IV. Modification de la configuration:

Vous allez maintenant constituer un réseau isolé avec les machines de la salle en modifiant les configurations IP de votre machine :

- L'adresse réseau que vous utiliserez sera 192.168.0.0/24
- Mettez-vous d'accord pour attribuer une adresse IP à chaque machine. Dessiner votre réseau
- Configurez votre machine en utilisant **ifconfig** et vérifiez ensuite la cohérence des différents paramètres renvoyés par **ifconfig -a**.

Ifconfig INTERFACE ADRESSE_IP netmask MASK

Q20. Afficher votre table de routage et expliquer sa cohérence. Testez ensuite la communication entre les diverses machines.

V. Routage direct

4. Consultation de la table de routage :

La table de routage se consulte avec la commande **route.**

Q21. Expliquer les 4 champs qui nous intéressent dans la table de routage.

La suppression d'une entrée dans la table s'effectue avec la commande :
route del –net ADRESSE_RESEAU netmask MASK gw ADRESSE_PASSERELLE

Q22. Essayez de nouveau la commande ping. La communication passe-t-elle toujours entre les machines ? Quel est le message renvoyé ? Expliquez vos réponses.

Recréer l'entrée pour le réseau local dans la table et vérifiez que le ping passe à nouveau :

route add –net ADRESSE_RESEAU netmask MASK gw ADRESSE_PASSERELLE

5. Routage

Le routage direct est celui qui s'effectue lorsque les machines source et destination sont sur le même réseau physique. Vous allez maintenant regarder ce qui se passe si ces machines n'ont pas la même adresse réseau IP (i.e. elles n'appartiennent pas au même réseau " logique "). Pour cela configurez une des machines de votre réseau isolé avec une adresse réseau différente des autres.

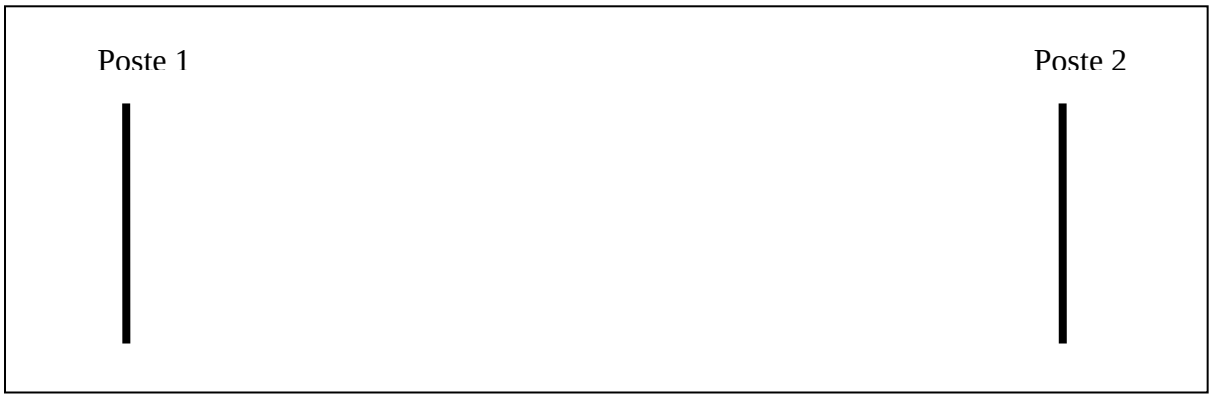
Q23. Testez la communication avec ping et expliquez.

6. Analyse des paquets échangés lors d'un ping :

Vous allez étudier les paquets échangés lors d'un routage direct entre deux machines sur un réseau isolé. Lancez un ping ne générant qu'une seule demande d'écho et faites une capture de trame en même temps.
Note : Utilisez l'option –c pour n'envoyer qu'un seul ping : **Ping -c 1** ***adresse-IP***

Q24. Remplir le schéma suivant en précisant toutes les trames qui circulent sur le réseau.

Note : Vous noterez pour chaque trame, les adresses MAC source et destination, les adresses IP et les protocoles utilisés

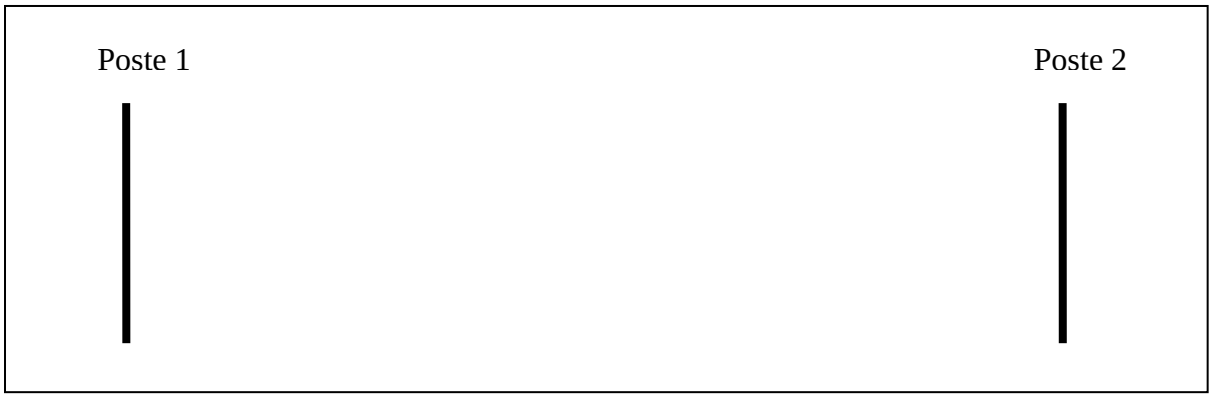


Consulter La table ARP .

Q25. Que contient-t-elle ?

Nettoyer la table ARP (arp –d) puis relancez la même manipulation.

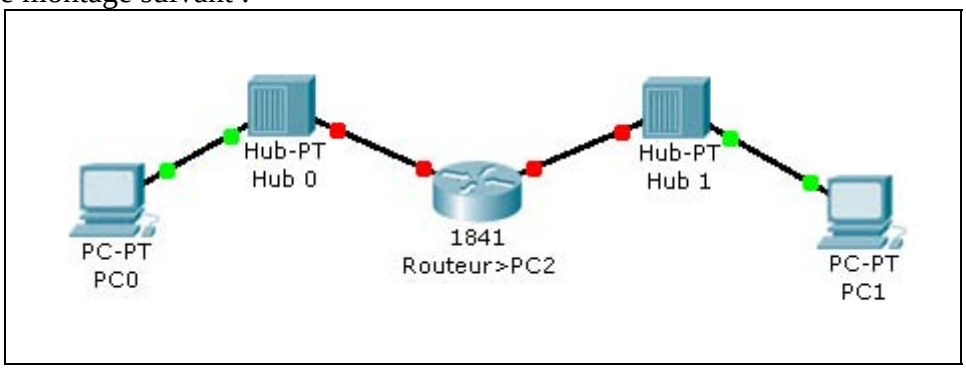
Q26. Remplir le schéma suivant en précisant toutes les trames qui circulent sur le réseau.



Q27. Si vous ne faites aucune manipulation avec votre machine pendant plusieurs minutes (typiquement 2 minutes). Que devient le contenu de votre table arp ? Faite l'essai en consultant régulièrement sa table.

VI. Mécanisme du routage:

Pour les plus avancées :
Vous avez besoin de 3 postes pour ce montage (mettez vous par groupe)
Réaliser le montage suivant :



Le routeur sera réalisé à l'aide d'une machine configurée en mode routeur en lançant la commande :
echo "1" > /proc/sys/net/ipv4/ip_forward

Q28. Configurer l'ensemble de votre réseau et valider le fait que toutes communications entre l'interface de tout les PC fonctionnent. Vous devez impérativement compléter votre schéma afin de faire figurer tout vos choix de configuration.

Q29. Pour les trames qui contiennent un paquet IP, analysez la valeur du champ TTL sur chacun des réseaux. Décrivez le mécanisme observé et expliquez son utilité.

Université de Savoie
Module Initiation réseau

TP2

Utilisation d'un simulateur réseau Packet Tracer

Présentation

1. Objectifs

Les objectifs de ce TP sont de simuler différents réseaux en utilisant le logiciel Packet Tracer afin de comprendre le fonctionnement du routage. Vous devrez complètement structurer un réseau afin de faire fonctionner l'ensemble des communications entres les ordinateurs.

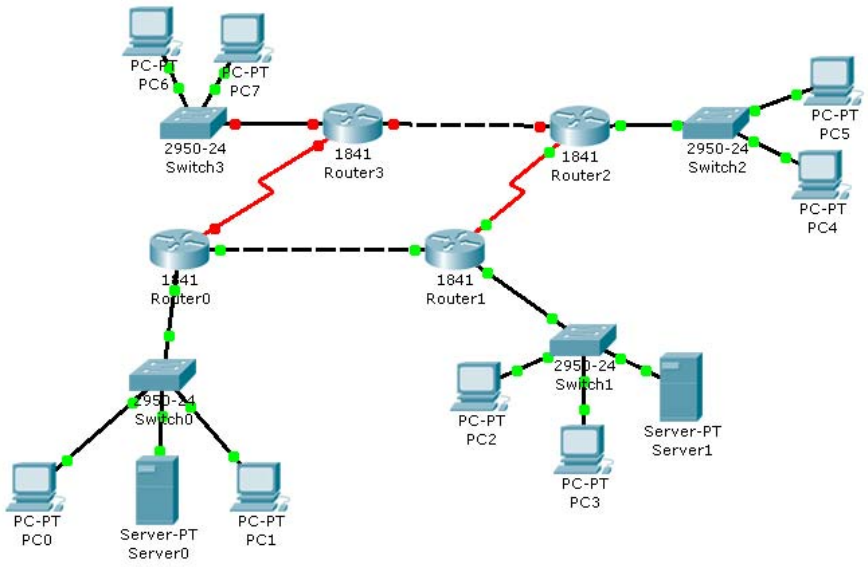
2. Première utilisation de Packet Tracer

Packet Tracer est un logiciel fourni par Cisco qui permet de simuler le fonctionnement de matériels réseaux. Des démonstrations d'utilisation des principales fonctionnalités de Packet Tracer sont disponibles dans Help | Tutorials. N'hésitez pas à les consulter. Dans ce TP, vous allez mettre en place un réseau relativement simple afin de vous familiariser avec les différentes fonctionnalités de Packet Tracer.

Réalisation

1. Schéma du réseau

Le réseau à simuler est le suivant :



Ce réseau comporte 4 sous réseaux reliés ensemble par des routeurs.

Les adresses IP des machines sont les suivantes :

Machines	@IP
PC0	200.6.0.10 / 24
PC1	200.6.0.11 / 24
PC2	200.6.1.10 / 24
PC3	200.6.1.11 / 24
PC4	200.6.2.10 / 24
PC5	200.6.2.11 / 24
PC6	200.6.3.10 / 24

Machines	@IP
PC7	200.6.3.11 / 24
Server0	200.6.0.250 / 24
Server1	200.6.1.250 / 24

Les routeurs ont les adresses IP suivantes :

Routeur	Int	@IP	Routeur	Int	@IP
Router0	Fa0/0	200.6.0.254/24	Router2	S0/0/0*	200.6.6.2 /30
Router0	Fa0/1	200.6.5.1/30	Router3	Fa0/0	200.6.3.254/24
Router0	S0/0/0*	200.6.8.1 /30	Router3	S0/0/0*	200.6.8.2 /30
Router1	Fa0/1	200.6.5.2 /30	Router3	Fa0/1	200.6.7.1 /30
Router1	Fa0/0	200.6.1.254 /24			
Router1	S0/0/0*	200.6.6.1/30			
Router2	Fa0/1	200.6.7.2 /30			
Router2	Fa0/0	200.6.2.254/24			

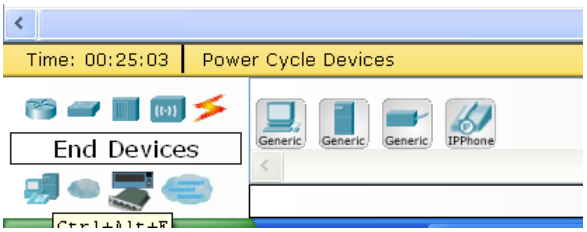
* Ces interfaces séries WAN doivent fournir l'horloge. Elles doivent être reliées avec un câble DCE.

Q1. Faite un schéma complet exhaustif de votre réseau au fur et à mesure que vous réalisez votre câblage. Les éléments qui doivent apparaître en un seul coup d'œil sont :

- Les groupes de machines appartenant au même réseau
- Les adresses des réseaux
- Les masques de sous réseaux
- Les adresses des machines
- Les numéros des interfaces utilisées pour chaque connexion.

2. Saisie du schéma

--> Démarrer Packet Tracer.
--> Cliquer sur End Device :



--> Placer les PC, les serveurs PT, les routeurs 1841, les switches 2950T
--> Cliquer ensuite sur Connections et relier les différents composants par des câbles en cuivre (Copper) droits ou croisés ou des liaisons Serial DCE ou Serial DTE.

Pour les liaisons, voila les types de câble réseau qu'il faut utiliser :
Câbles droits :

- PC à Hub
- PC à Switch
- Switch à Routeur

Câbles croisés :

- Switch à Switch
- Hub à Hub
- Routeur à Routeur
- PC à PC
- Hub à Switch
- PC à Routeur

Pour les liaisons du schéma, vous n'utiliserez pas le choix automatique proposé par packet tracer.

Pour les liaisons séries entres routeurs, il faudra rajouter le module WIC 2T au routeur. Ce module permet de rajouter une interface série afin de relier deux réseaux. Une des deux extrémités doit fournir une horloge (128000). Pour ajouter ce type d'interface, il faut utiliser la souris en « glisser/déposer » dans un slot libre. Il faut aussi penser à éteindre le module (en cliquer sur l'interrupteur du module).

3. Configuration IP

Pour configurer une machine, choisissez l'onglet Config, puis choisissez votre interface à configurer.
Une fois le schéma réalisé, vérifier que vous arrivez à communiquer au sein de l'ensemble des sous réseaux. Pour cela vous enverrez des Ping au sein du sous réseau, grâce à l'interface de commande : *cliquez sur une machine > onglet Desktop > Command prompt*.

Q2. Faites vérifier les communications au sein de tous les sous-réseaux.

4. Configuration du routage

Configurer l'ensemble des machines et des routeurs afin que le réseau fonctionne totalement. C'est-à-dire que n'importe quel PC/Routeur peut joindre n'importe quel PC/Routeur.
Pour cela, cliquer sur les PC ou les routeurs >onglet Config>Routing>Static.

Q3. Faites vérifier les communications de l'ensemble de votre réseau.

5. Test de la configuration

1. Cliquer sur l'enveloppe  pour ajourer une PDU (Protocol Data Unit) .
2. Cliquer ensuite sur une machine source, puis sur la machine cible.
3. Un paquet ICMP (ping) est alors envoyé à la machine cible.
4. Le résultat du ping s'affiche en bas à droite.

Remarque :
Le logiciel peut mettre un certain temps à se stabiliser. Commencez par tester le bon fonctionnement des machines les plus proches, puis, petit à petit, tester celui des machines les plus éloignées.
Il est aussi possible de tester via PC>Desktop >Web Browser le bon fonctionnement des serveurs http.

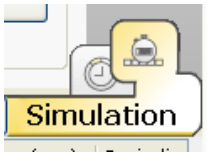
Q4. Tester l'envoi de trame par PDU.

Q5. Tester l'accès au serveur web de votre réseau via le Web Browser.

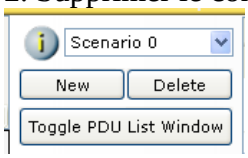
6. Simulation :

6.1. Fonctionnement de la simulation

1. Passez en mode simulation :



2. Supprimer le contenu du scénario en cours (delete) :



3. Envoyer un PDU simple entres deux PC de votre choix.

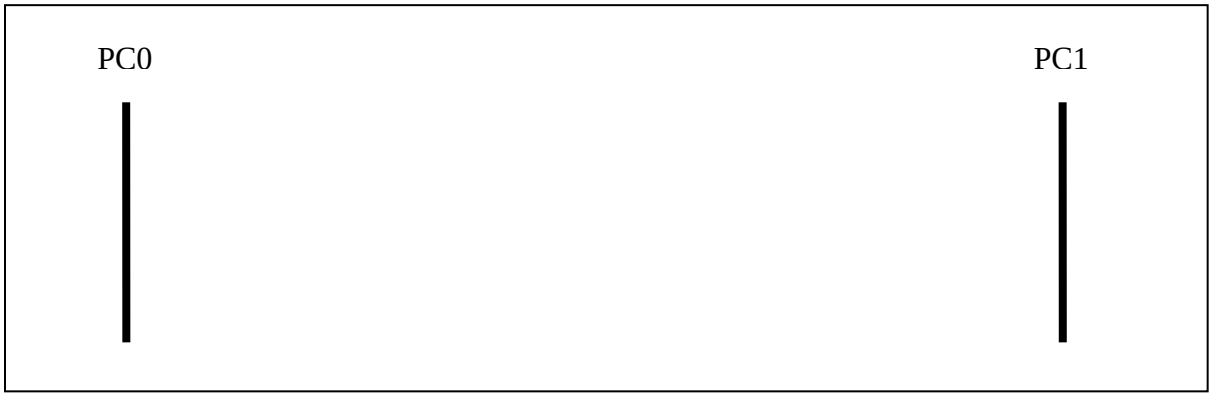
4. Cliquer sur Capture / Forward pour avancer pas à pas.

6.2. 1er test

Effacer le scénario de simulation en cours. Effacer la table ARP de PC0 (arp –d). Faites un envoi de PDU entre PC0 et PC1.

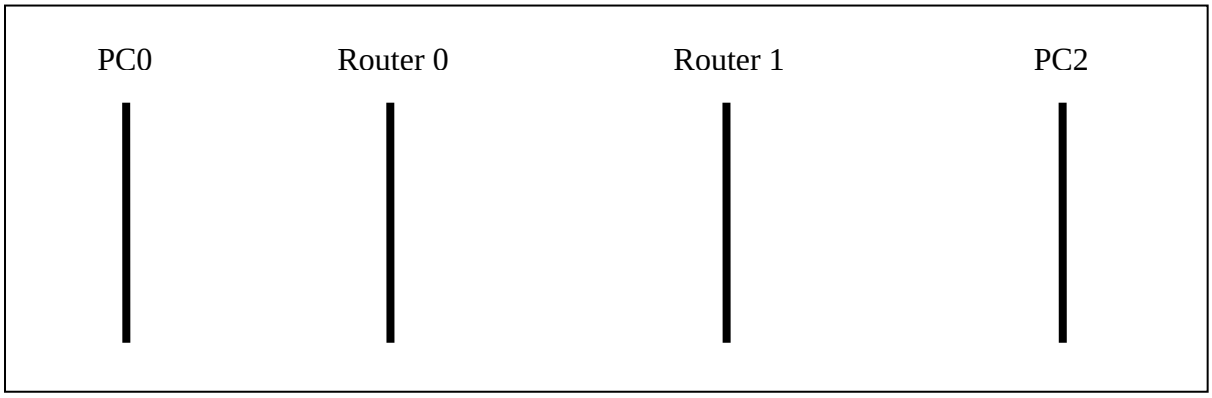
Q6. Combien de paquet sont prêt à être émis à partir de PC0 dans la simulation à venir ? Pourquoi ?

Cliquer une fois sur Capture / Forward, puis cliquer sur le paquet sur le Switch. Noter les adresses MAC source et destination. Faites la même chose à chaque Capture / Forward et remplissez le schéma suivant.



6.3. 2ème test

Effacer le scénario de simulation en cours. Effacer la table ARP de PC0 (arp –d) et celle du PC 2. Faites un envoi de PDU entre PC0 et PC2. Remplissez le schéma suivant.



Université de Savoie
Module ETRS-808
Initiation aux réseaux IP

Ethernet embarqué

Objectifs : Etude d'une documentation et mise en œuvre de composant numérique.

I. Documentations constructeurs

Lors d'une recherche d'information sur une technologie ou sur un composant en particulier, nous avons les types de documents à notre disposition :
Datasheet : Informations précises sur l'architecture du composant. Elle est toujours composée des catégories :

- *Features :* Caractéristique du composant
- *Descriptions*
- *Pin Diagram*
- *Device overview*
- *Pinout description*
- *Electrical Characteristic* : Absolute maximum rating, AC and DC caractéristics
- *Packaging information*
- *Index*
- *Worldwide sales and service*

Application note : Explication d'une application particulière pour un circuit ou une famille de circuits. L'application et sa réalisation sont expliquées pas à pas pour aider l'utilisateur à se familiariser avec le composant.
User guide : Guide d'utilisation, d'une carte de développement ou d'un logiciel de développement.
Selection guide : Un guide permettant de faire le tri pour aider l'utilisateur à choisir un composant précis.
Solution guide : Présentation d'une réalité industrielle puis des solutions qu'apporte le constructeur.
Errata : les corrections apportés aux documentations existantes, et les suivis de modifications.

II. Etude d'une documentation :

Q1. Repérer les 7 grands blocs de l'architecture du composant :

……...

……...

……...

……...

Q2. Quelle est la fréquence du quartz qu'il faut implémenter pour cadencer le composant ?

……...

Q3. A quoi sert l'OST : Oscillator Start-up Timer ?

……...

……...

Q4. Le composant possède une pin CLKOUT. Pour quelle utilisation cette clock peut être utilisée ?

……...

……...

Q5. Quel schéma faut il concevoir autour du ENC28J60 afin de faire fonctionner l'application.

Q6. A quel niveau logique fonctionne le composant ? Comment va-t-on gérer la commande du composant si le microcontrôleur que nous utilisons est en 5V. A l'inverse, comment va-t-on gérer les signaux à destination du microcontrôleur en provenance du composant ENC28J60. Proposez une solution.

……...

……...

……...

……...

Q7. Comment le composant arrive à détecter le sens la polarité des leds lors d'un RESET ? Quelle influence dans la configuration du composant, la polarité de la LEDB en particulier ?

……...

……...

Q8. A quoi servent les bit LACFG3 :LACFG0 et LBCFG3 :LBCFG0 ?

……...

Q9. Quelle technologie de mémoire est utilisée dans le composant ? Rappelez ces caractéristiques ? Quelles parties du composant seront stockées dans ces mémoires.

……...

……...

Q10. Quelle quantité mémoire est utilisée pour stocker l'ensemble des « Control Registers » ?

……...

Q11. Quelle quantité mémoire est utilisée pour stocker l'ensemble des « Ethernet Registers » ?

……...

Q12. Quelle quantité mémoire est utilisée pour stocker l'ensemble des « PHY registers » ?

……...

Q13. Quels sont les 3 groupes de registres appartenant aux « Control Registers » ?

……...

Q14. Repérer les méthodes de gestion des deux buffers circulaires FIFO pour la réception et pour l'émission.

Q15. Repérer la méthode pour lire et la méthode pour écrire un « PHY register ».

4.0. SERIAL PERIPHERAL INTERFACE (SPI)

Q16. Expliquer le rôle de CS, SCK, SI, SO.

……...

Q17. Combien d'instructions le composant est capable de comprendre ?

……...

5.0 ETHERNET OVERVIEW

Q18. Est-ce que le microcontrôleur doit gérer le start-of-Frame delimiter, le padding et le CRC ?

……...

6.0 INITIALIZATION

Q19. Quelle est la recommandation sur le choix de l'adresse de début (ERXST) du buffer de réception ?

……...

Q20. Quelle est l'action à mener pour l'initialisation du buffer de transmission ?

……...

Q21. A quoi faudra t il veiller lors de l'écriture des différents packet à transmettre dans le buffer de transmission ?

……...

Q22. Prendre connaissance des « MAC Initialization Settings »

……...

Q23. Dans quel cas faut il faire l'initialisation des PHY Module ?

……...

III. Mise en place d'une application de test

Vous considérerez la situation suivante :
Vous venez de prendre note de la documentation du composant ENC28J60. Ce composant paraît être une bonne alternative pour l'application que vous souhaitez mettre en place et vous venez de recevoir un échantillon de la carte implémentant ce composant.

Votre hiérarchie vous demande de mettre en œuvre une application de test le plus rapidement possible afin de valider le fonctionnement et les possibilités des connexions réseaux que vous pourriez mettre en place.

Vous avez tout à votre disposition :

- Carte mikroelectronika
- Exemple de code
- Libraires
- PC
- Carte et câble réseau
- Oscilloscope
- Wireshark
- Hub
- Etc…

Q24. Mettez en place l'application de test de votre choix.

Université de Savoie
Initiation aux réseaux IP

TP 4
Programmation client/serveur par socket

1. Information sur les sockets

1.1. Généralités sur les sockets

Les sockets sont issues de la culture UNIX. Ce terme désigne l'ensemble des fonctions API (interface de programmation) disponibles pour accéder au réseau et échanger des données. Il existe plusieurs familles de sockets : sockets Unix pour accéder aux fichiers, sockets Apple, sockets IPv6 et celles qui nous intéressent particulièrement : les sockets Internet IPv4 (Internet Sockets, AF_INET). Dans la suite, le terme de socket désigne systématiquement les sockets Internet.

Il existe différents types de socket :

- Les sockets orientées connexion ou "*Stream Sockets*" (SOCK_STREAM) ou. Elles permettent une communication selon le protocole TCP,
- Les sockets sans connexion ou "*Datagram Sockets*" (SOCK_DGRAM). Elles permettent une communication selon le protocole UDP,

Dernier point, une socket, après création, est simplement identifiée par un entier (numéro de socket). Ceci à l'avantage de simplifier leur manipulation et d'offrir un peu de cohérence avec les fichiers.

Dans la suite, pour les structures et prototypes de fonctions, il peut y avoir quelques différences légères suivant les plates-formes. Il ne faut alors pas hésiter à regarder les ***man pages***.

1.2. Les principales structures de données

Les structures de données suivantes sont utilisées pour interagir avec les sockets. **Les octets constituant les différentes données sont stockés selon l'ordre "*Big Endian*"** (octet de poids fort en premier) quel que soit la plate-forme d'exécution. Le "*Big Endian*" est nommé le "*Network Byte Order*". L'utilisation du *"Little Endian"* (octet de poids faible en premier) est appelé le *"Host Byte Order"*. Aussi, il existe de nombreuses fonctions assurant la conversion.

Les structures sont déclarées dans l'en-tête `#include <sys/socket.h>`

1.2.1 Structure décrivant les sockets

La structure qui décrit tous types de socket est de la forme suivante :

```
struct sockaddr
{
 unsigned short sa_family; // Type identifié par la constante AF_INET
 char sa_data[14];         // Données propres à chaque protocole
};
```

Une structure de taille identique a été créée spécialement pour les sockets Internet :

```
struct sockaddr_in
{
 short int sin_family;        // Type de socket
 unsigned short int sin_port; // Numéro de port
 struct in_addr sin_addr;     // adresse IP
 unsigned char sin_zero[8];   // zeros pour avoir la même taille
};
```

Le champs in_addr est de la forme suivante :

```
struct in_addr
{
 in_addr_t s_addr;
};
```

Aussi, lorsque les fonctions nécessitent le type **`struct sockaddr`**, il suffit de passer en paramètre la structure de type **`struct sockaddr_in`** avec conversion de type (cast explicite). Le champ **`sin_zero`** a été ajouté afin que cette structure ait la même taille. Le champ **`sin_family`** correspond à **`sa_family`** et doit contenir la valeur **`AF_INET`**. Les champs **`sin_addr`** et **`sin_port`** contiennent respectivement l'adresse IP et le port de la connexion. Plus précisemment, c'est le champ **`s_addr`** du champ **`sin_addr`** qui contient véritablement la valeur d'adresse IP.

1.2.2 Fonctions de conversion

Afin de gérer les problèmes de conversion (« Network Byte Order » et « Host Byte Order » sur les numéros de port et les adresses IP, il existe une série de fonctions :

- `htons() : Host TO Network Short`
- `htonl() : Host TO Network Long`
- `ntohs() : Network TO Host Short`
- `ntohl() : Network TO Host Long`

Ces fonctions sont déclarées dans l'en-tête **`#include <netinet/in.h>`**

Exemple : affectation d'un port dans une structure socket

```
int port;
struct sockaddr_in socket;

socket.sin_port = htons(8080);
port = ntohs(socket.sin_port);
```

1.2.3 Fonctions de manipulation des adresses IP

Il existe également des fonctions permettant de manipuler les adresses IP. Voici les deux principales assurant la conversion entre la représentation textuelle d'une adresse IP (notation en décimale pointée) et la représentation entière :

Transformation de l'adresse IP donnée au format « a.b.c.d » en un entier long non signé (valeur de retour de la fonction).

- `in_addr_t inet_addr(char *)`

Transformation d'une adresse IP au format struct in_addr au format « a.b.c.d » (valeur de retour)

- `char * inet_ntoa (struct in_addr in)`

Transformation de l'adresse IP donnée au format « a.b.c.d » (cp) en une structure in_addr(in).

- `int inet_aton(const char *cp, struct in_addr in)`

Ces fonctions sont déclarées dans l'en-tête **`#include <arpa/inet.h>`**

Exemple :

```
char * IP;
struct sockaddr_in socket;

socket.sin_addr.s_addr = inet_addr("193.48.120.109");
IP = inet_ntoa(socket.sin_addr); /* ATTENTION : chaîne constante retournée */
```

1.3. Les principales fonctions pour les sockets

1.3.1 Création d'une socket : `socket()`

Cette fonction permet la création d'une socket

```
#include <sys/socket.h>
int socket(int domain, int type, int protocol);
```

Paramètres :

- `domain` : pour des sockets Internet, toujours donner la valeur **`AF_INET`**
- `type` : **`SOCK_STREAM`** pour TCP ou **`SOCK_DGRAM`** pour UDP
- `protocol` : toujours à 0 (en fait dépend de la valeur donnée pour type)

Valeur de retour : numéro de socket. Valeur égale –1 en cas d'erreur.

Par rapport à la connexion future cette primitive ne fait que donner le premier élément du quintuplet :

{***protocole***, port source, adresse IP source, port dest, adesse IP dest}

1.3.2 Communiquer à travers un port : `bind()`

Cette fonction permet d'associer une socket à un port source et une adresse IP source afin de pouvoir échanger avec une machine cible.

```
#include <sys/socket.h>
int bind(int socketfd, struct sockaddr * addr, int addrlen);
```

Paramètres :

- `socketfd` : descripteur de socket à savoir la valeur retournée par **`socket()`**
- `addr` : pointeur sur une structure **`struct sockaddr`** contenant notamment le numéro du port sur lequel le système doit écouter les demandes de connexion

- `addrlen` : taille de la structure qui vient d'être donnée en paramètre

Valeur de retour : 0 en cas de réussite. Sinon –1 en cas d'erreur.

Remarques :
Dans beaucoup d'application (notament dans la programmation des clients), on a pas besoin de s'inquiéter du numéro de port, le protocole sous-jacent s'occupe de choisir un numéro de port pour l'application. On n'a donc pas besoin d'utiliser bind() pour la programmation client. En général, les serveurs ont des numéros de port bien connus du système. Bind() dit au système « c'est mon adresse, tout message reçu à cette adresse doit m'être envoyé ».

Cet appel système complète l'adresse locale et le numéro de port du quintuplet qui qualifie la connexion :

{***protocole***, **port source**, **adresse IP source**, port dest, adesse IP dest}

1.3.3 Se connecter : `connect()`

Cette fonction permet de réaliser une connexion TCP avec une machine cible.

```
#include <sys/socket.h>
int connect(int socketfd, struct sockaddr * destAddr, int addrlen);
```

Les paramètres sont identiques à la fonction **`bind()`** en dehors du fait que destAddr contient l'adresse et le port de la machine cible.

Le quintuplet est maintenant complet :

{***protocole***, **port source**, **adresse IP source**, **port dest**, **adesse IP dest**}

1.3.4 Ecouter les demandes de connexion : `listen()`

Cette fonction permet de configurer les demandes de connexion. C'est-à-dire de préciser une taille de file d'attente pour les connexion à venir, lié à la socket créer précédement. Cette fonction marche généralement avec les sockets de type `SOCK_STREAM`.

```
#include <sys/socket.h>
int listen(int socketfd, int backlog);
```

Paramètres :

- `socketfd` : descripteur de socket à savoir la valeur retournée par **`socket()`**
- `backlog` : taille de la file d'attente

Valeur de retour : 0 en cas de réussite. Sinon –1 en cas d'erreur.

1.3.5 Accepter une connexion : `accept()`

Il s'agit d'une fonction bloquante qui ne rendra la main que lorsqu'une demande de connexion est réellement reçue. Cette fonction accepte une demande de connexion et ouvre une socket pour que la communication puisse se faire.

```
#include <sys/socket.h>
int accept(int socketfd, struct sockaddr * addr, unsigned int * addrlen);
```

Paramètres :

- `socketfd` : descripteur de socket à savoir la valeur retournée par **`socket()`**
- `addr` : un pointeur sur une structure `struct sockaddr_in` qui va contenir les informations concernant l'appelant. Ceci est donc une nouvelle structure différente que de celle de la configuration du serveur. Cette structure vierge sera remplie par la fonction accept avec les informations de l'appelant.
- `addrlen` : pointeur sur une variable entière initialisée avec `sizeof(struct sockaddr_in)`. Si en retour cette valeur est modifiée, c'est que tous les champs n'ont pas été renseignés.

Valeur de retour : descripteur de la nouvelle socket. Sinon –1 en cas d'erreur. C'est ce descripteur qui sera utilisé par la suite pour toute communication avec le nouvel interlocuteur.

1.3.6 Echanger des données en mode connecté : `send()` et `recv()`

Ces fonctions servent à envoyer et à recevoir des données à travers des sockets de type `SOCK_STREAM`.

```
#include <sys/socket.h>
int send(int socketfd, void * data, int length, unsigned int flags);
int recv(int socketfd, void * data, int length, unsigned int flags);
```

Paramètres :

- `socketfd` : descripteur de socket à savoir la valeur retournée par **`socket()`**
- `data` : un pointeur sur la zone devant contenir ou recevoir les données.
- `length` : taille maximale de la zone contenant ou recevant les données.
- `flags` : options paramétrant la réception ou l'envoie de données (en général 0).

Valeur de retour : nombre d'octets réellement envoyés ou reçus. Sinon –1 en cas d'erreur. Dans le cas de la fonction **`recv()`**, ce nombre peut également être 0 ce qui signifie en général que la connexion a été rompue ou qu'un problème est apparu.

1.3.7 Echanger des données en mode non connecté : `sendto()` et `recvfrom()`

Ces fonctions sont identiques aux précédentes en dehors du fait qu'elles concernent les sockets de type `SOCK_DGRAM` et qu'elles prennent deux paramètres supplémentaires permettant d'identifier le correspondant.

```
#include <sys/socket.h>
int sendto(int socketfd, void * data, int length, unsigned int flags,
struct sockaddr * toaddr, int addrlen);
int recvfrom(int socketfd, void * data, int length, unsigned int flags,
struct sockaddr * fromaddr, int addrlen);
```

Paramètres et Valeur de retour : cf `accept()`, `send()` et `recv()`

1.3.8 Clore une connexion : `close()`

La fonction close() termine une connexion entièrement.

```
#include <sys/socket.h>
int close(int socketfd);
```

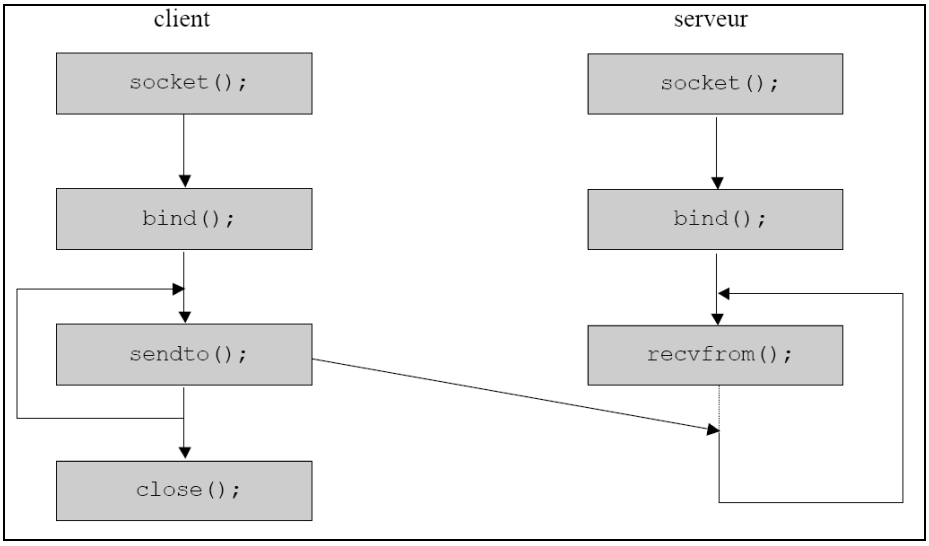
Paramètres :

- `socketfd` : descripteur de socket à savoir la valeur retournée par **`socket()`**

Valeur de retour : 0 en cas de succès. Sinon –1 en cas d'erreur.

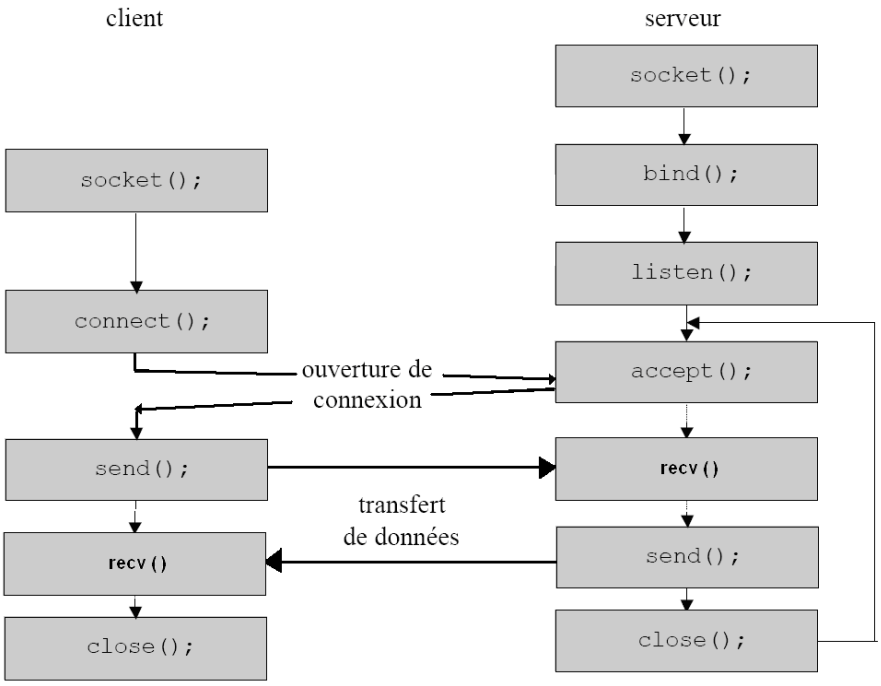
1.4. Exemple de client/serveur UDP

1.4.1 Schéma de fonctionnement général d'un client/serveur UDP



1.5. Exemple de client/serveur TCP

1.5.1 Schéma de fonctionnement général d'un client/serveur TCP



2. Fonctionnement d'un client

→ Démarrer une machine sur la première partition linux 2 (nécessaire si on veut faire des captures de trames), et s'identifier en tant que **root** / **TP_reseau**.
→ Se connecter à etu-bourget : Poste de travail | Fichier | Se connecter à un serveur | Type de service = Partage Windows | Serveur = etu-bourget | nom d'utilisateur =votre login
Dans un premier temps, vous allez observer comment fonctionne un client réseau. A partir d'un *shell* sous linux, effectuer la commande **`telnet www.univ-savoie.fr 80`**. Puis, lorsque vous êtes connecté, entrer la requête **`GET / HTTP/1.0`** suivi de deux retours à la ligne (touche Entrée/Return).

Q1. A quoi sert d'ordinaire la commande telnet ? Pourquoi a-t-on ajouté la valeur 80 ?

Q2. Quel est le protocole de la couche application utilisé ?

Q3. Qu'obtient-on ? Que fait donc la requête ?

Q4. Quelle est la taille de la réponse ?

En vue de la partie suivante, il va être nécessaire de récupérer l'adresse IP du serveur **`www.univ-savoie.fr`**.

Q5. Quelle commande utiliser et quelle est l'adresse IP du serveur ?

3. : Programmation d'un client minimal

Dans cette partie, il s'agit de programmer en langage C et à l'aide des *sockets* ce qui a été réalisé avec la commande **`telnet`**. Vous réaliserez donc le client TCP.
Faire valider, par une capture de trame et par l'affichage de la réponse du de la requete HTTP.

Pour les plus avancés (1), modifier le code pour que l'adresse IP du serveur puisse être passé sous la forme d'un argument depuis la ligne de commande au lieu d'être "codé en dur" dans le programme.

Pour les plus avancés (2), modifier le code pour utiliser le nom du serveur **`www.univ-savoie.fr`** au lieu de son adresse IP. Il va falloir ajouter le code nécessaire pour interroger le serveur DNS par défaut qui assurera la conversion du nom en son adresse IP.

4. Programmation d'un serveur minimal (partie facultative)

De la même façon, il s'agit de programmer la partie serveur. Pour le serveur, les champs sin_port et s_addr correspondent respectivement au « Port d'écoute » et à « l'adresse IP du serveur ».

Le client et le serveur s'exécuteront dans un premier temps sur la même machine locale, c'est-à-dire l'adresse IP 127.0.0.1.

4.1. Coté Serveur

Nous réaliserons un serveur dont le protocole nous est propre. On s'accorde sur le fait que le serveur pourra recevoir deux requêtes : 0 ou 1.
Cas de la réception d'une requête '1' :
Le serveur répond : « Nous avons reçu votre requête numéro 1 »

Cas de la réception d'une requête '0' :
Le serveur répond : « Nous avons reçu votre requête numéro 0 »

4.2. Coté Client

Vous reprendrez exactement le code de votre client web réalisé à la question précédente. Il vous suffit de remplacer la requête initiale par celle approprié à votre protocole.

Pour les plus avancés (1), choisissez une machine pour la salle, et modifier votre code afin de réaliser votre propre connexion client/serveur sur deux machines différentes.