TD4 - Sujet de Synthèse

Q2
Marc Haye - Partie 1
Evan Nunes - Partie 2
Damien Rostaing - Partie 3

<u>1 ère partie : Remise en état du réseau</u>

1 - Quel est le rôle de la commande ping ?

Le rôle de la commande ping est de vérifier la disponibilité du réseau ou d'un ordinateur au sein d'un réseau local ou privé, il peut aussi permettre de vérifier si le DNS est bien configuré, c'est-à-dire que si le DNS est mal mis, faire ping un nom de domaine pourrait poser un problème.

2 - Quel est le rôle de la commande IPCONFIG ?

La commande IPCONFIG permet d'afficher des informations sur la configuration du réseau et d'actualiser les paramètres DHCP et DNS.

3 - À quoi correspondent les informations affichées ? Détaillez vos réponses et donnez également les classes d'adresse utilisées.

Les informations affichées correspondent à la carte Ethernet de la carte mère au sein de son réseau local et distant. En l'occurrence, ces commandes sont effectuées sur le PC COM, et sur la station météo. Pour PC COM, les champs d'adresses utilisés sont respectivement de 2.1.0.0 à 2.1.255.255.

4 - Que déduisez-vous de ces résultats ?

Je peux déduire pour le PC COM qu'il est dans un réseau local de classe B, que son adresse réseau est 2.1.0.0, que le nombre d'adresses possible est 255^2. On peut également en déduire que la station météo n'est connectée à aucun réseau.

5 - Ya-t-il un routeur dans ce réseau ?

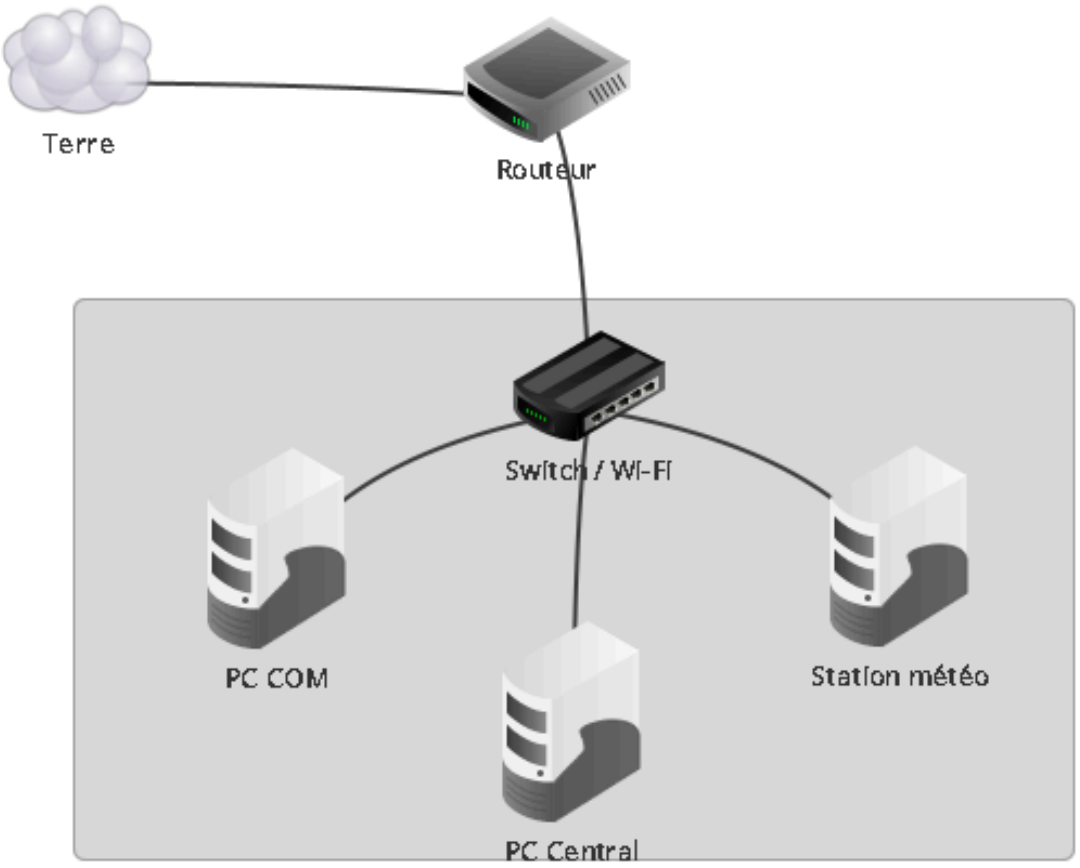
Il doit nécessairement avoir un routeur dans ce réseau afin de pouvoir envoyer un SOS avec la Terre.

6 - Déterminez quel(s) équipement(s) de connexion (HUB, Switch, Routeur, ...) sont nécessaires et pourquoi?

Le routeur est nécessaire car il permet de relier le réseau local de PC COM à celui de la station météo, dans notre cas pour pouvoir communiquer avec la Terre. Le HUB est facultatif, et un switch n'est pas non plus forcément nécessaire mais pourrait être utile dans le cas où plusieurs ordinateurs doivent être connectés au sein du même réseau.

7 - Déterminez le plan de câblage de votre réseau local

(Plus d'informations sur le réseaux (IP, routage) sont données dans le pdf en pièce jointe)



/home/haye/Documents/S3/Architecture-Reseaux/TD/TD4-Synthese/mars.txt.fls

2ème partie : Création de l'application de communication

10 - Expliquez l'échange de données de l'annexe 1

Les 3 premières lignes sont l'établissement de la connexion :

- 1 : Demande de **syn**chronisation au serveur
- 2 : Signale que le paquet est un accusé de réception et établissement de la connexion coté Serveur
- 3 : établissement de la connexion coté Client

La ligne 4, est donc les données envoyées par la Station Météo

Les 3 dernières lignes sont la rupture de la connexion :

- 5 : Signale que le paquet est un accusé de réception coté client
- 6 : Demande la **fin** de la connexion
- 7 : Demande la **fin** de la connexion et un accusé de réception

Ensuite, de la ligne 9 à 14, c'est la même chose, mais la différence, c'est que c'est le Client qui va envoyer des données cette fois-ci.

11 & 12 - Proposer le code

```c
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>
#include <netinet/in.h>
        #include <sys/socket.h>
        #include <sys/types.h>
        #include <strings.h>
#include <unistd.h>
#define PORT 12345
        int sock, socket2, lg;
char mess[80];
struct sockaddr_in local;   // champs d entete local
struct sockaddr_in distant; // champs d entete distant
void creer_socket(){
    sock = socket(AF_INET, SOCK_STREAM, 0);
    distant.sin_family = AF_INET;
    distant.sin_port = htons(PORT);
}
int main(){
    // creation socket
    creer_socket();
    // boucle sans fin pour la gestion des connexions
    while (1){ // attente connexion client
        printf("En attente de la connection au serveur\n");
        if(connect(sock,(struct sockaddr *)&distant, sizeof(distant))){
            printf("connecté au server \n");
            strcpy(mess, "");
            while (strncmp(mess, "fin", 3) =/= 0){
                send(sock, "SOS-Code d'activation de la capsule attendu !!", 80, 0);
            }
            close(sock); // on lui ferme la socket
        }
    }
}
```

3ème partie : AIE, ça ne marche toujours pas …

13 - Qu'est-ce qu'un pare-feu ? Comment fonctionne-t-il ?

un pare-feu fait en sorte de bloquer les connexions suspecte et empêche souvent les connexions malveillantes de réseau extérieur. Le pare-feu analyse tout ce qui est envoyé dans le réseau pour être sûr qu'il n'y ait aucune chose suspicieuse sur le réseau.

14 - Qu'est ce qu'un DoS ? Quelle peuvent être les conséquences de ce type d'attaque sur un serveur ?

un DoS a ne pas confondre avec un DDoS représente le fait de faire avec une machine une attaque à déni de service, c'est-à-dire qu'on va faire énormément de requête avec la même machine
ce qui n'est pas à confondre avec le DDoS qui lui se fait avec plusieurs machines. Les attaques DoS peuvent causer des problèmes de service à la machine attaquée en faisant crash la machine ou le service utilisé.

15 - Qu'elles modifications proposez vous dans le code précédent ?

Dans le code, on pourrait faire en sorte de créer 2000 threads en même temps pour faire le même message 2000 fois en simultané.

```c
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>
#include <netinet/in.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <unistd.h>
#include <pthread.h>

#define PORT 12345
#define NUM_THREADS 5
#define MESSAGE_COUNT 2000

int sock;
struct sockaddr_in distant;

void send_sos(voidarg) {
    char mess[80] = "SOS-Code d'activation de la capsule attaquant !!";
    for (int i = 0; i < MESSAGE_COUNT; i++) {
        send(sock, mess, strlen(mess), 0);
    }
    pthread_exit(NULL);
}

void creer_socket() {
    sock = socket(AF_INET, SOCK_STREAM, 0);
    distant.sin_family = AF_INET;
    distant.sin_port = htons(PORT);
    inet_pton(AF_INET, "127.0.0.1", &distant.sin_addr);
}

int main() {
    pthread_t threads[NUM_THREADS];
    creer_socket();

    printf("En attente de la connexion au serveur\n");
    if (connect(sock, (struct sockaddr *)&distant, sizeof(distant)) < 0) {
        perror("Erreur de connexion");
        exit(1);
    }
    printf("Connecté au serveur\n");

    for (int i = 0; i < NUM_THREADS; i++) {
        pthread_create(&threads[i], NULL, send_sos, NULL);
    }

    for (int i = 0; i < NUM_THREADS; i++) {
        pthread_join(threads[i], NULL);
    }

    close(sock);
    return 0;
}
```