Contrôle Systèmes d'exploitation, Réseaux

Vendredi 24 Mai 2013
14h - 17h
Aucun document n'est autorisé

Exercice 1 : (5 = 1 + 2 + 1 + 1)

1. Citez les 4 couches du modèle TCP/IP. Nous ne vous demandons pas d'expliquer leur fonctionnalité.
2. Quels sont les 2 principaux protocoles de transport dans le modèle TCP/IP ? Indiquez en quelques lignes leurs différences principales, les cas d'utilisation.
3. Une trame Ethernet se compose de 5 parties dans le modèle TCP/IP. Rappelez ce que sont ces 5 parties.
4. Qu'est-ce que le MTU ? Que définit-il ?

Exercice 2 : (5 = 1 + 1 + 1 + 1 + 1)

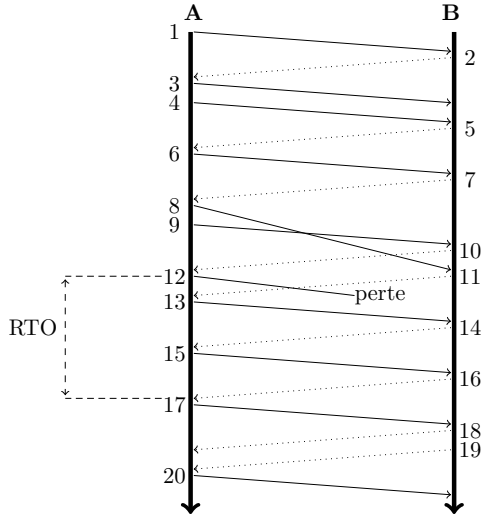
Une entreprise composée de 12 départements se voit affecter l'adresse IP 202.12.123.0. L'administrateur souhaite affecter un sous-réseau à chaque département.

1. De quelle classe d'adressage s'agit-il? Combien de machines cela permet-il d'adresser?
2. En supposant que le nombre de départements de l'entreprise ne va pas tellement évoluer, quel est le masque de sous-réseau optimal ?
3. Combien de départements peuvent être ajoutés avec ce masque ?
4. Combien de machines chaque département peut-il comporter ?
5. Quelle est l'adresse de broadcast du premier sous-réseau

Exercice 3 : (5)

Nous considérons une connexion TCP entre deux stations A et B. Seul A veut envoyer des données vers B. B n'a rien à envoyer, à l'exception des messages d'acquittement. Pour le transfert de A vers B, la taille des données d'un segment est fixée à 100 octets.

La transmission démarre, nous représentons dans la figure 1 le chronogramme obtenu : complétez les paramètres associés à chaque segment (Numéros de séquence et d'acquittement, flags et Taille des données) dans le tableau fourni en annexe A.



Exercice 4 : Client-Serveur(5 = 1+1+1+1+1)

On considère pour toute cette épreuve les deux programmes B et C dont les codes sont donnés en annexe.

1. Préciser le programme qui correspond au serveur en justifiant le rôle de chacune des primitives associées (écrites soulignées)
2. Même question pour le client.
3. Le programme qui correspond au serveur est exécuté sur la machine TERRE et le client sur la machine LUNE.
   - Dans quel ordre doit-on lancer l'exécution de ces programmes pour avoir une exécution correcte ? Décrivez graphiquement le processus d'exécution.
   - Quels doivent être les paramètres du programme client pour que le serveur affiche :
     `Message reçu : Test de transfert`
     `Message sans espace : Testdetransfert`
4. Compléter les codes (en rappelant les numéros de ligne).
5. On substitue dans le code du programme B les lignes 31 à 40 par les lignes suivantes :

```
31 listen(s, 3);
32
33 i = sizeof(struct sockaddr_in);
34 for (;;) {                                    /* Boucle infinie */
35         t = accept(s,(struct sockaddr *) &isa, &i);
36         if (t < 0) {
37         perror("accept"); exit(1);
38         }
39         switch(fork()) {
40                 case -1 :    perror("fork"); close(s); close(t); exit(1);
41                 case 0 :     close(s); travaille(t); close(t); exit(0);
42                 default :    close(t);
43         }
44 }
```

   Expliquer pourquoi on utilise la fonction `travaille(t)` avec le paramètre `t` et non le paramètre `s` à la nouvelle ligne 41 ?

Fonctions et structures utiles:

```
1  int socket(int domain, int type, int protocol);
2  int connect(int sockfd, struct sockaddr *serv_addr, socklen_t addlen);
3  int bind(int sockfd, struct sockaddr *my_addr,socklen_t addlen);
4  int listen(int sockfd, int size);
5  int accept(int sockfd, struct sockaddr *addr, socklen_t addlen);
6  struct hostent *gethostbyname(const char *name);
7
8  struct hostent {
9                 char  *h_name;            /* official name of host */
10                char **h_aliases;         /* alias list */
11                int    h_addrtype;        /* host address type */
12                int    h_length;          /* length of address */
13                char **h_addr_list;       /* list of addresses */
14 };
15 #define h_addr h_addr_list[0] /* for backward compatibility */
16
17 struct sockaddr {
18                unsigned short  sa_family;
19                char            sa_data[14];
20 };
21 struct sockaddr_in {
22         short   sin_family;
23         u_short sin_port;
24         struct  in_addr sin_addr;
25         char    sin_zero[8];
26 };
```

Annexe B

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/utsname.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <unistd.h>

#define PORTNUM 50000
void travaille (int s) ;

int main(int argc, char *argv[]) {
int s, t, i;
struct sockaddr_in sa, isa;

if ((s = socket(COMPLETER, COMPLETER, COMPLETER)) < 0) {
        perror("socket"); exit(2) ;
}

sa.sin_family = COMPLETER;
sa.sin_port =   htons(PORTNUM);
sa.sin_addr.s_addr = INADDR_ANY;

if (bind(COMPLETER,COMPLETER,COMPLETER) < 0) {
        close(s); perror("bind") ; exit(3) ;
}

listen(s, 1);

i = sizeof(struct sockaddr_in);
for (;;) {                                  /* Boucle infinie */
        t = accept(s,(struct sockaddr *) &isa, &i);
        if (t < 0) {
        perror("accept"); exit(1);
        }
        travaille(t);
}
}

void travaille (int s) {
char msg[256] ;
char new_msg[256] ;
char *dst, *src;
int n;

n = read(s,msg,100) ;  /* On ne lit que les 255 premiers caracteres */
msg[n] = '\0';
printf("Message recu : %s\n",msg) ;

/* Portion de code pour garder la chaine d'entree sans espace */
/* isblank(c) retourne vrai si le caractere c est un espace ou une tabulation, faux sinon */
src=msg ;
dst=new_msg ;
while (*src != '\0') {if (!isblank(*src)) {*dst=*src; dst++;}; src++;};
*dst='\0';

/* Affichage du message sans espace */
printf("Message sans espace : %s\n",new_msg) ;
close(s) ;
}
```

Annexe C

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include <sys/utsname.h>
5  #include <sys/types.h>
6  #include <sys/socket.h>
7  #include <netinet/in.h>
8  #include <arpa/inet.h>
9  #include <netdb.h>
10 #include <unistd.h>
11
12 #define PORTNUM 0
13
14 int main(int argc, char *argv[]) {
15 struct sockaddr_in sa;
16 struct hostent *hp;
17 int a,s,t ;
18
19 if (argc != 4) {
20         perror("Il faut 3 parametres\n") ; exit(2) ;
21 }
22
23 if ((hp = gethostbyname(argv[1])) == NULL) {
24         perror("gethostbyname") ; exit(1);
25 }
26
27 if ((s = socket(COMPLETER, COMPLETER, COMPLETER)) < 0) {
28         perror("socket") ; exit(3);
29 }
30
31 sa.sin_family = AF_INET;
32 sa.sin_port = htons((u_short)atoi(argv[2]));
33 memcpy(&sa.sin_addr.s_addr, hp->h_addr, hp->h_length);
34
35 if (connect(COMPLETER, COMPLETER, COMPLETER) < 0) {
36         close(s); perror("connect"); exit(4);
37 }
38
39 if (write(s,argv[3],strlen(argv[3])) != strlen(argv[3])) {
40         perror("Write on socket") ; exit(3) ;
41 }
42
43 close(s) ;
44 }
```