Examen
Système et Réseaux
25 Mai 2011

Durée 3 heures

Chaque exercice doit être reporté sur une feuille séparée
Seules les polycopiés de cours sont autorisées
Les réponses doivent êtres clairs et dûment justifiées
Nous accorderons une attention particulière à la présentation

Exercice 1 : Adressage IP (5 points) (30 minutes)

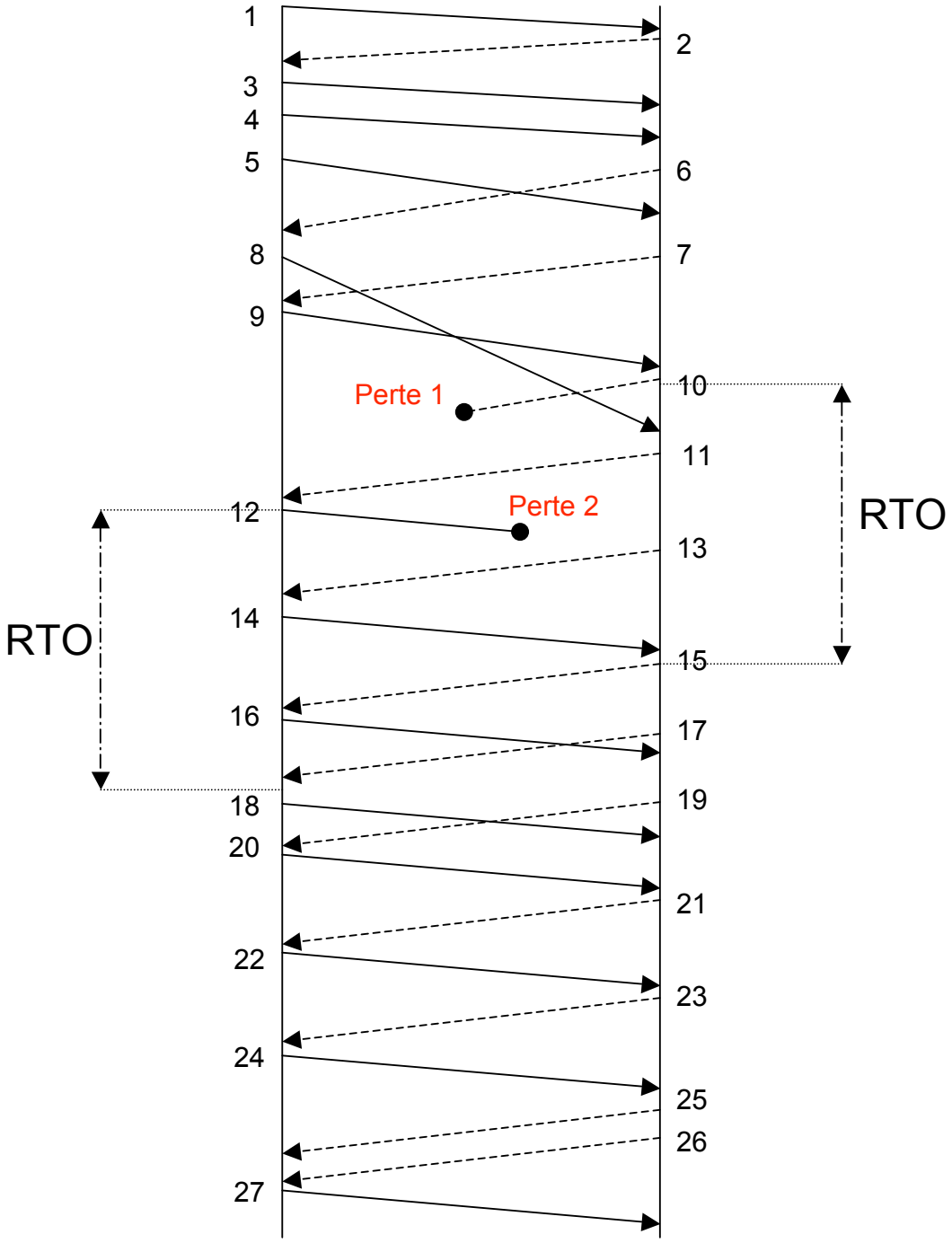
Une entreprise composée de 10 départements se voit affecter l'adresse IP 191.24.0.0. L'administrateur souhaite affecter un sous-réseau à chaque département.

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

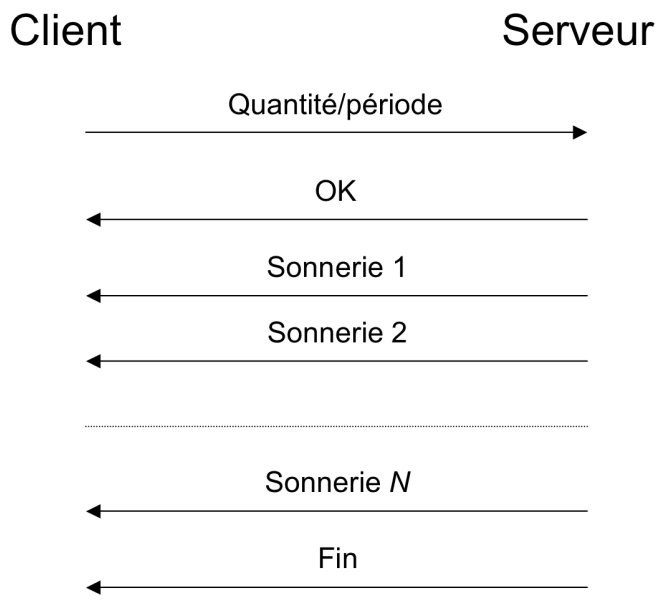
Exercice 2 : TCP (5 points) (45 minutes)

Nous considérons une connexion TCP bidirectionnelle entre deux stations A et B. Les deux stations échangent des données. Pour le transfert de A vers B, la taille maximale des données d'un segment (MSS) est 200 octets. Pour le transfert de B vers A, la taille maximale des données d'un segment (MSS) est 100 octets. Ces données sont envoyées à chaque fois où cela est possible.

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.


1
2
3
4
5
6
8
7
9
Perte 1
10
11
RTO
12
Perte 2
13
RTO
14
15
16
17
18
19
20
21
22
23
24
25
26
27

Exercice 3 : Programmation socket UDP (5 points) (45 minutes)



Nous souhaitons réaliser un dialogue client/serveur UDP selon le protocole suivant :
Le client :

1. envoie un message contenant 2 entiers (Q, P).
2. lit les messages et les affiche en clair (type et valeur).
3. se termine sur réception du message FIN.

Le serveur :

1. lit le message du client (Q, P).
2. répond immédiatement par le message OK.
3. répète Q fois :
    a. attendre P secondes,
    b. envoyer un message de sonnerie I, où I est le numéro de séquence.
4. envoie le message FIN.

Les programmes clients et serveurs UDP sont donnés dans l'annexe B. Le serveur aura 1 argument : le port. Le client aura 4 arguments : le serveur (format ascii), le port, Q et P.

Complétez les zones de textes manquantes, indiquées par **????** de ces programmes.
Reportez ces lignes sur votre copie, en référençant le numéro de la ligne.

Exercice 4 : Programmation Socket TCP (5 points) (45 minutes)

Nous souhaitons réaliser la même application que la précédente. Mais cette fois_ci en utilisant TCP. Le serveur aura 1 argument : le port. Le client aura 4 arguments : le serveur (format ascii), le port, Q et P.

Complétez les zones de textes manquantes, indiquées par ???? de ces programmes.
Reportez ces lignes sur votre copie, en référençant le numéro de la ligne.

ANNEXE B

```
 1 /* clientu.c (client UDP) */
 2
 3 #include <stdio.h>
 4 #include <errno.h>
 5 #include <netinet/in.h>
 6 #include <string.h>
 7
 8 short sport = 0;
 9 int sock = 0; /* socket de communication */
10
11
12 int main(int argc, char** argv) {
13   struct sockaddr_in moi; /* SAP du client */
14   struct sockaddr_in serveur; /* SAP du serveur */
15
16   int ret,len,p,q,i;
17
18   int serveur_len = sizeof(serveur);
19   char buf_read[101], buf_write[101];
20
21   if (argc != 5) {
22       fprintf(stderr,"usage: %s host sport Q P\n",argv[0]);
23       exit(1);
24   }
25   sport = atoi(argv[2]);
26   q = atoi(argv[3]);
27   p = atoi(argv[4]);
28
29   if ((sock = socket( ???????? )) == -1) {
30       fprintf(stderr,"%s: socket %s\n",argv[0],strerror(errno));
31       exit(1);
32   }
33
34   len = sizeof(moi);
35   getsockname( ???? );
36   serveur.sin_family = ???? ;
37   serveur.sin_port = ???? ;
38   inet_aton(argv[1], ???? );
39
40   sprintf(buf_write,"%d %d",q,p);
41
42   while (strlen(buf_write) < 100)
43       strcat(buf_write," ");
44
45   ret = sendto( ???? );
46
47   if (ret <= 0) {
48       printf("%s: erreur dans sendto (num=%d,
mess=%s)\n",argv[0],ret,strerror(errno));
49       return -1;
50   }
51
52   len = sizeof(moi);
53   getsockname( ???? );
54   printf("le client recoit : ");
55   ret = recvfrom( ???? );
```

```
 56   if (ret <= 0) {
 57       printf("%s: erreur dans recvfrom (num=%d,
mess=%s)\n",argv[0],ret,strerror(errno));
 58       return -1;
 59   }
 60
 61   printf("%s\n",buf_read);
 62   for (i = 0; i < q;i++) {
 63       getsockname( ???? );
 64       printf("le client recoit ");
 65       ret = recvfrom( ???? );
 66       if (ret <= 0) {
 67           printf("%s: erreur dans recvfrom (num=%d, mess=%s)\n",
argv[0],ret,strerror(errno));
 68           return -1;
 69       }
 70       printf("%s\n",buf_read);
 71   }
 72
 73   getsockname( ???? );
 74   printf("le client recoit : ");
 75   ret = recvfrom( ???? );
 76   if (ret <= 0) {
 77       printf("%s: erreur dans recvfrom (num=%d,
mess=%s)\n",argv[0],ret,strerror(errno));
 78       return -1;
 79   }
 80   printf("%s\n",buf_read);
 81   return 0;
 82 }
```

```
  1 /* serveuru.c (serveur UDP) */
  2
  3 #include <stdio.h>
  4 #include <errno.h>
  5 #include <string.h>
  6 #include <netinet/in.h>
  7
  8 short port = 0;
  9 int sock = 0; /* socket de communication */
 10 int nb_reponse = 0;
 11
 12 int main(int argc, char** argv) {
 13     int ret;
 14     struct sockaddr_in serveur; /* SAP du serveur */
 15
 16     if (argc != 2) {
 17         fprintf(stderr,"usage: %s port\n",argv[0]);
 18         exit(1);
 19     }
 20     port = atoi(argv[1]);
 21
 22     if ((sock = socket( ???? )) == -1) {
 23         fprintf(stderr,"%s: socket %s\n",argv[0],strerror(errno));
 24         exit(1);
 25     }
```

```
26
27      serveur.sin_family = ???? ;
28      serveur.sin_port = ???? ;
29      serveur.sin_addr.s_addr = ???? ;
30      if (bind( ???? ) < 0) {
31          fprintf(stderr,"%s: bind %s\n", argv[0],strerror(errno));
32          exit(1);
33      }
34
35      while (1) {
36          struct sockaddr_in client; /* SAP du client */
37          int client_len = sizeof(client);
38          char buf_read[101], buf_write[101];
39          char chaine[10];
40
41          int i,j,p,q;
42
43          ret = recvfrom( ???? );
44
45          if (ret <= 0) {
46              printf("%s: recvfrom=%d: %s\n", argv[0],ret,strerror(errno));
47              return -1;
48          }
49
50          chaine[0]='\0';
51          for (i = 0; buf_read[i] != ' '; i++)
52              chaine[i] = buf_read[i];
53
54          chaine[i]='\0';
55          q = atoi(chaine);
56          chaine[0]='\0';
57
58          for (i++, j = 0; buf_read[i] != ' '; i++, j++)
59           chaine[j] = buf_read[i];
60          chaine[j]='\0';
61
62          p = atoi(chaine);
63
64          printf("serveur a recu p=%d, q=%d\n",p,q);
65          strcpy(buf_write,"OK");
66          while (strlen(buf_write) < 100)
67              strcat(buf_write," ");
68
69          ret = sendto( ???? );
70          if (ret <= 0) {
71                printf("%s: sendto=%d: %s\n", argv[0],ret,strerror(errno));
72              return -1;
73          }
74
75          for (i = 0; i < q;) {
76              sleep(p);
77              sprintf(buf_write,"S %d",++i);
78              while (strlen(buf_write) < 100)
79                  strcat(buf_write," ");
80              ret = sendto( ???? );
81              if (ret <= 0) {
82                  printf("%s: sendto=%d: %s\n",argv[0],ret,strerror(errno));
83                  return -1;
```

```
84              }
85          }
86
87          strcpy(buf_write,"FIN");
88          while (strlen(buf_write) < 100)
89              strcat(buf_write," ");
90          ret = sendto( ???? );
91          if (ret <= 0) {
92              printf("%s: sendto=%d: %s\n", argv[0],ret,strerror(errno));
93              return -1;
94          }
95      }
96      return 0;
97  }
```

```
 1  /* clientt.c (client TCP) */
 2
 3  #include <stdio.h>
 4  #include <errno.h>
 5  #include <netinet/in.h>
 6  #include <string.h>
 7
 8  short sport = 0;
 9  int sock = 0; /* socket de communication */
10
11  int main(int argc, char** argv) {
12      struct sockaddr_in moi; /* SAP du client */
13      struct sockaddr_in serveur; /* SAP du serveur */
14      int ret,len,p,q,i,num;
15      char buf_read[4];
16
17      if (argc != 5) {
18          fprintf(stderr,"usage: %s serveur port Q P\n",argv[0]);
19          exit(1);
20      }
21      sport = atoi(argv[2]);
22      q = atoi(argv[3]);
23      p = atoi(argv[4]);
24      if ((sock = socket( ???? )) == -1) {
25          fprintf(stderr,"%s: socket %s\n",argv[0],strerror(errno));
26          exit(1);
27      }
28      serveur.sin_family = ???? ;
29      serveur.sin_port = ???? ;
30      inet_aton(argv[1], ???? );
31      if (connect( ???? ) < 0) {
32          fprintf(stderr,"%s: connect %s\n",argv[0],strerror(errno));
33          perror("bind");
34          exit(1);
35      }
36      len = sizeof(moi);
37      getsockname( ???? );
38      ret = write( ???? ,sizeof(q));
39      if (ret < 2) {
```

```
 40         printf("%s: erreur dans write (num=%d,
mess=%s)\n",argv[0],ret,strerror(errno));
 41         return -1;
 42     }
 43     ret = write( ???? ,sizeof(p));
 44     if (ret < 2) {
 45         printf("%s: erreur dans write (num=%d,
mess=%s)\n",argv[0],ret,strerror(errno));
 46         return -1;
 47     }
 48     ret = read(???? ,buf_read,2);
 49     if (ret <= 0) {
 50         printf("%s: erreur dans read (num=%d,
mess=%s)\n",argv[0],ret,strerror(errno));
 51         return -1;
 52     }
 53     printf("Le client recoit %s\n",buf_read);
 54     for (i = 0; i < q; i++) {
 55         ret = read( ???? ,buf_read,1);
 56         if (ret <= 0) {
 57             printf("%s: erreur dans read (num=%d,
mess=%s)\n",argv[0],ret,strerror(errno));
 58             return -1;
 59         }
 60         buf_read[1] = '\0';
 61         ret = read( ???? ,sizeof(num));
 62         if (ret <= 0) {
 63             printf("%s: erreur dans read (num=%d,
mess=%s)\n",argv[0],ret,strerror(errno));
 64             return -1;
 65         }
 66         printf("Le client recoit %s%d\n",buf_read,num);
 67     }
 68     ret = read( ???? ,3);
 69     if (ret <= 0) {
 70         printf("%s: erreur dans read (num=%d,
mess=%s)\n",argv[0],ret,strerror(errno));
 71         return -1;
 72     }
 73     buf_read[3] = '\0';
 74     printf("Le client recoit %s\n",buf_read);
 75     close( ???? );
 76     return 0;
 77 }
```

```
  1 /* serveurt.c (serveur TCP) */
  2
  3 #include <stdio.h>
  4 #include <errno.h>
  5 #include <netinet/in.h>
  6 #include <string.h>
  7
  8 short port = 0;
  9 int sock = 0; /* socket de communication */
 10
 11 int main(int argc, char** argv) {
```

```
12      struct sockaddr_in serveur; /* SAP du serveur */
13
14      if (argc != 2) {
15          fprintf(stderr,"usage: %s port\n",argv[0]);
16          exit(1);
17      }
18      port = atoi(argv[1]);
19      if ((sock = socket( ???? )) == -1) {
20          fprintf(stderr,"%s: socket %s\n",argv[0],strerror(errno));
21          exit(1);
22      }
23      serveur.sin_family = ???? ;
24      serveur.sin_port = ???? ;
25      serveur.sin_addr.s_addr = ???? ;
26      if (bind( ???? )) < 0) {
27          fprintf(stderr,"%s: bind %s\n",argv[0],strerror(errno));
28          exit(1);
29      }
30      if (listen( ???? ) != 0) {
31          fprintf(stderr,"%s: listen %s\n",argv[0],strerror(errno));
32          exit(1);
33      }
34      while (1) {
35          struct sockaddr_in client; /* SAP du client */
36          int len = sizeof(client);
37          int sock_pipe; /* socket de dialogue */
38          int ret,p,q,i;
39
40          sock_pipe = accept( ???? );
41          ret = read( ???? ,sizeof(q));
42          if (ret <= 0) {
43              printf("%s: read=%d: %s\n",argv[0], ret, strerror(errno));
44              return -1;
45          }
46          ret = read( ???? ,sizeof(p));
47          if (ret <= 0) {
48              printf("%s: read=%d: %s\n",argv[0], ret, strerror(errno));
49              return -1;
50          }
51          printf("serveur a recu p=%d, q=%d\n",p,q);
52          ret = write( ???? ,"OK",2);
53          if (ret <= 0) {
54              printf("%s: write=%d: %s\n", argv[0], ret, strerror(errno));
55              return -1;
56          }
57          for (i = 0 ; i < q;) {
58              sleep(p);
59              ret = write( ???? ,"S",1);
60              if (ret <= 0) {
61                  printf("%s: write=%d: %s\n",argv[0], ret, strerror(errno));
62                  return -1;
63              }
64              i++;
65              ret = write( ???? ,&i,sizeof(i));
66              if (ret <= 0) {
67                  printf("%s: write=%d: %s\n",argv[0], ret, strerror(errno));
68                  return -1;
69              }
```

```
70          }
71          ret = write( ???? ,"END",3);
72          if (ret <= 0) {
73              printf("%s: write=%d: %s\n",argv[0], ret, strerror(errno));
74              return -1;
75          }
76          close( ???? );
77      }
78      return 0;
79  }
80
```

Nom : Num Etudiant :
Prenom :

Exercice 2 :

#	Num Seq	Num Ack	URG	ACK	PSH	RST	SYN	FIN	Taille Données
1	2499	-					X		0
2	3523			X			X		0
3				X					0
4									200
5									200
6									100
7									
8									
9									
10									
11									
12									
13									
14									
15									
16									
17									
18									
19									
20									
21									
22									
23									
24								X	0
25				X					100
26								X	0
27				X					

Nom : Num Etudiant :
Prenom :