Programmation Réseaux
Correction Examen
6 Mai 2008

1 – Protocoles

1.1

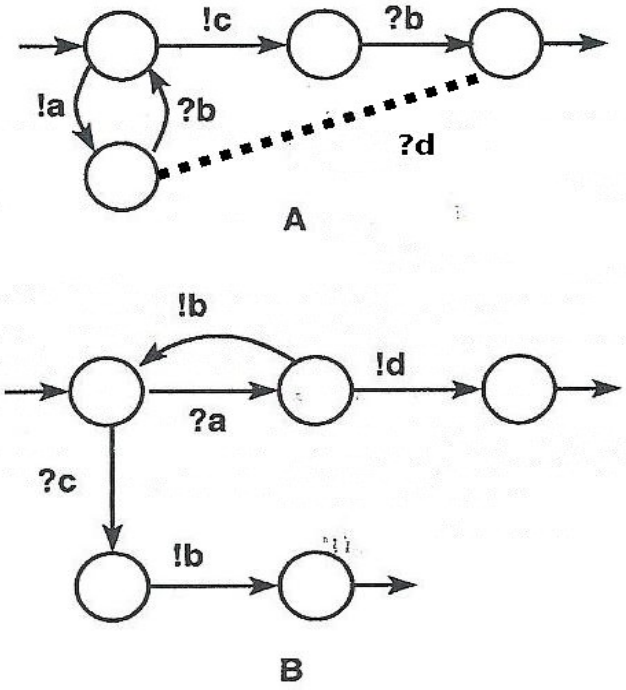
Schéma 1 --- OK
Schéma 2 --- OK
Schéma 3 --- PAS OK
-> A viole le protocole en premier:
A : !a!a?b?b!c?b
B : ?a?a!b!b?c!b
Le premier message qui est en faute: aa

1.2

A: !a?d
B: ?a!d

Dans ce cas de figure, il y a un problème.

1.3



2 – **Efficacité**

2.1

120 Mb / s = 120 x 10^6 b /s

1 Mo = 1 x 10^6 octets = 1 x 8 x 10^6 bits

=> (8 / 120) s = (1 / 15) s = 0,066667 s = 67 ms

Avec une latence de 23 ms, on a donc :
67 ms + 23 ms = 90 ms

Donc pour 1 Mo de données entre PPS et l'IML, il faut **90 ms.**

2.2

4 h = 4 x 60 minutes = 4 x 60 x 60 s = 4 x 60 x 60 x 10^3 ms
= 14400000 ms

Avec une latence de 23 ms, on a donc :
14400000 ms + 23 ms = **14400023 ms**

2.3

120 Mb / s = 120 x 10^6 b /s

1 To = 1 x 10^{12} octets = 1 x 8 x 10^{12} bits

=> ((8 x 10^6) / 120) s = 66667 s = 66667 x 10^3 ms

Avec une latence de 23 ms, on a donc :
66667 x 10^3 ms + 23 ms = **66690 ms**

Etc... pour la suite :)

3 – Programmation TCP

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.net.Socket;
import java.net.UnknownHostException;

public class Test {

  public static void main(String[] args) {
        String host = "news.jussieu.fr";
        String groupe = args[0];
        String ln;
        int port = 119;

        try {
                Socket s = new Socket(host, port);
                //Ouverture des flux sortants du socket
                OutputStream out = s.getOutputStream();
                //Ouverture des flux entrants du socket
                BufferedReader in =
                        new BufferedReader
                        (new InputStreamReader(s.getInputStream()));
                out.write(("GROUP "+groupe+"\n").getBytes());
                ln = in.readLine();
                if(ln.equals("411 No such group "+groupe))
                {
                        System.out.println("Le Groupe "+
                                        groupe+" n'existe pas.");
                        return;
                }
                else
                        System.out.println("Le Groupe "+groupe+
                                                " existe.");
                //On quitte correctement la session NNTP
                out.write(("QUIT\n").getBytes());
                s.close();
                return;
        } catch (UnknownHostException e) {
                System.err.println("Host inconnue: "+e);
        } catch (IOException e) {
                System.err.println("IOException: "+e);
        }
   }
}
```

4 – Programmation non-bloquante

```
#define BUFLEN 512

extern struct hostent *gethostbyname();
struct hostent *machine;

int main(int argv, char * argc[]) {
```

```
         struct sockaddr_in addr;
        int n, sock;
        int port = 7; /* Port echo()*/
        char *buffer = malloc(BUFLEN);
        struct timeval tv;
         tv.tv_sec = 3;

        sock = socket(PF_INET, SOCK_DGRAM, 0);
        if( sock < 0 )
         {
                  perror("socket()");
                  exit(errno);
         }

        n = setsockopt(sock,
                         SOL_SOCKET,
                SO_RCVTIMEO,(struct timeval *)&tv, sizeof(struct
timeval));
        if(n < 0)
        {
                      perror("setsockopt()");
                      exit(errno);
           }

         if((machine = gethostbyname("nivose.informatique.univ-paris-
diderot.fr")) == NULL)
         {
                  perror("gethostbyname()");
                  exit(errno);
         }

        memset((void*)&addr, 0, sizeof(addr));
         addr.sin_family = AF_INET;
         addr.sin_port = htons(port);
         memcpy(&addr.sin_addr, machine->h_addr, machine->h_length);
         buffer = "Packet a envoyer\0";

         n = sendto(sock, buffer,
                         BUFLEN, 0, (struct sockaddr*)&addr,
sizeof(addr));
         if(n < 0)
         {
                  printf("sendto()");
                  exit(errno);
        }
         n = -1;
         n = recvfrom(sock, buffer, BUFLEN, 0, NULL, NULL);
         if(n < 0)
          {
                  printf("pas de reponse\n");
                   close(sock);
          exit(errno);
         }
         printf("reponse recue\n");
        close(sock);
         return EXIT_SUCCESS;
}
```