Étude d'un système informatisé – Session 2004

Le réseau mis en œuvre est à base d'une architecture Ethernet 100baseT. La communication entre le système temps réel et le système de supervision utilise un protocole propriétaire. Le réseau possède une adresse IP de classe C : 192.168.17.0
La communication des informations capteur entre le système temps réel et le système de supervision utilise le réseau Ethernet 100baseT et le protocole IP .
On décide d'établir une communication en mode **connecté**. Le système temps réel joue le rôle de serveur ; le PC de supervision est un client de ce dernier. Le port d'écoute fixé par le service est le port **2467**.

Les lignes suivantes proposent des solutions en langage C pour l'ouverture de la socket de communication sur le PC de supervision et sur le système temps réel. Pour chaque ligne, préciser, sans justifier, si elle répond aux besoins de l'application.

Réponse : (Barrer les réponses inexactes)

int Sock = socket(AF_INET, SOCK_STREAM, 0) ;

int Sock = socket(AF_INET, SOCK_DGRAM, 0) ;

int Sock = socket(AF_INET, SOCK_RAW, IPPROTO_ICMP) ;

int Sock = socket(AF_UNIX, SOCK_STREAM, 0) ;

int Sock = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP) ;

int Sock = socket(AF_IPX, SOCK_STREAM, 0) ;

int Sock = socket(AF_INET, SOCK_STREAM, IPPROTO_UDP) ;

La socket de communication étant ouverte avec succès tant au niveau du PC de supervision que du système temps réel, l'application serveur attache maintenant cette socket à un point de communication. Cet attachement (binding) nécessite la fourniture d'une structure de type **sockaddr_in**.

```
struct sockaddr_in
{
    sa_family_t        sin_family; /* address family: AF_INET */
    u_int16_t          sin_port;   /* port in network byte order */
    struct in_addr     sin_addr;   /* internet address */
};

/* Internet address. */
struct in_addr
{
    u_int32_t    s_addr;     /* IPv4 address in network byte order */
};
```

Les questions qui suivent permettent de définir les valeurs des différents membres de la structure de type **sockaddr_in** à fournir à l'appel bind() lors de l'initialisation de la socket serveur.
(prototype : int bind(int Socket, struct sockaddr *Name, int NameLen);)

Le type **u_int16_t** correspond à un entier non signé sur 16 bits.
Quelle valeur doit être fournie pour le champ **sin_port** de la structure sur le système temps réel?

Réponse :

sin_port =

A quoi correspond le champ **in_addr** de cette structure dans ce cas ? Ce champ est souvent initialisé avec la valeur symbolique **INADDR_ANY**. Que signifie cette valeur ?

Réponse :

in_addr :

INADDR_ANY :

L'appel à **bind()** sera suivi d'un appel à la primitive **listen()** puis à la primitive **accept()**. Quels rôles jouent chacun de ces appels systèmes dans une communication en mode connecté ?

Réponse:

listen() :

accept() :

Les deux trames reproduites ci-dessous sont extraites d'un relevé réalisé par l'analyseur de protocole lors de l'envoi de la séquence informant la supervision que le robot R1 est passé au-dessus de l'un des marqueurs d'un bain.

L'analyse présente, pour les trames 13 et 14, la description des différents protocoles utilisés : Ethernet, IP, TCP.
Pour chaque protocole, la première ligne donne les caractéristiques principales, les lignes suivantes les valeurs et la signification des différents champs.

La dernière partie donne pour chaque trame le contenu brut en hexadécimal suivi d'une représentation ASCII.

```
Trame   Heure       Adr MAC src     Adr MAC dst   Protocole   Description
13      5.643554    000BDB14E06B    LOCAL         TCP         .AP..., len:    3,

Frame: Total frame length: 60 bytes
  ETHERNET: ETYPE = 0x0800 : Protocol = IP:  DOD Internet Protocol
    + ETHERNET: Destination address : 00E018B96B0B
    + ETHERNET: Source address : 000BDB14E06B
      ETHERNET: Frame Length : 60 (0x003C)
      ETHERNET: Ethernet Type : 0x0800 (IP:  DOD Internet Protocol)
      ETHERNET: Ethernet Data: Number of data bytes remaining = 46 (0x002E)
  IP: ID = 0x6C12; Proto = TCP; Len: 43
      IP: Version = 4 (0x4)
      IP: Header Length = 20 (0x14)
      IP: Precedence = Routine
      IP: Type of Service = Normal Service
      IP: Total Length = 43 (0x2B)
      IP: Identification = 27666 (0x6C12)
      IP: Flags Summary = 2 (0x2)
          IP: .......0 = Last fragment in datagram
          IP: ......1. = Cannot fragment datagram
      IP: Fragment Offset = 0 (0x0) bytes
      IP: Time to Live = 128 (0x80)
      IP: Protocol = TCP - Transmission Control
      IP: Checksum = 0xEB49
      IP: Source Address = 192.168.17.22
      IP: Destination Address = 192.168.17.10
      IP: Data: Number of data bytes remaining = 23 (0x0017)
      IP: Padding: Number of data bytes remaining = 3 (0x0003)
  TCP: .AP..., len:    3, seq:2700201124-2700201127, ack:2053738035, src: 2467  dst: 4425
      TCP: Source Port = 0x09A3
      TCP: Destination Port = 0x1149
      TCP: Sequence Number = 2700201124 (0xA0F1CCA4)
      TCP: Acknowledgement Number = 2053738035 (0x7A698E33)
      TCP: Data Offset = 20 (0x14)
      TCP: Flags = 0x18 : .AP...
          TCP: ..0..... = No urgent data
          TCP: ...1.... = Acknowledgement field significant
          TCP: ....1... = Push function
          TCP: .....0.. = No Reset
          TCP: ......0. = No Synchronize
          TCP: .......0 = No Fin
      TCP: Checksum = 0x3D42
      TCP: Data: Number of data bytes remaining = 3 (0x0003)

00000:  00 E0 18 B9 6B 0B 00 0B DB 14 E0 6B 08 00 45 00   .à.¹k...Û.àk..E.
00010:  00 2B 6C 12 40 00 80 06 EB 49 C0 A8 11 16 C0 A8   .+l.@.€.ëIÀ¨..À¨
00020:  11 0A 09 A3 11 49 A0 F1 CC A4 7A 69 8E 33 50 18   ...£.I ñÌ¤ziŽ3P.
00030:  FA F0 3D 42 00 00 43 06 00 00 00 00               úð=B..C.....

*******************************************************************************************
```



```
Trame   Heure       Adr MAC src     Adr MAC dst   Protocole   Description
14      5.846679    LOCAL           000BDB14E06B  TCP         .A...., len:    0

Frame: Total frame length: 54 bytes
  ETHERNET: ETYPE = 0x0800 : Protocol = IP:  DOD Internet Protocol
    + ETHERNET: Destination address : 000BDB14E06B     + ETHERNET: Source address : 00E018B96B0B
      ETHERNET: Frame Length : 54 (0x0036)
      ETHERNET: Ethernet Type : 0x0800 (IP:  DOD Internet Protocol)
      ETHERNET: Ethernet Data: Number of data bytes remaining = 40 (0x0028)
  IP: ID = 0xD73F; Proto = TCP; Len: 40
      IP: Version = 4 (0x4)
      IP: Header Length = 20 (0x14)
      IP: Type of Service = Normal Service
      IP: Total Length = 40 (0x28)
      IP: Identification = 55103 (0xD73F)
      IP: Flags Summary = 2 (0x2)
          IP: .......0 = Last fragment in datagram
          IP: ......1. = Cannot fragment datagram
      IP: Fragment Offset = 0 (0x0) bytes
      IP: Time to Live = 64 (0x40)
      IP: Protocol = TCP - Transmission Control
      IP: Checksum = 0xC01F
      IP: Source Address = 192.168.17.10      IP: Destination Address = 192.168.17.22
      IP: Data: Number of data bytes remaining = 20 (0x0014)
  TCP: .A...., len:    0, seq:2053738035-2053738035, ack:2700201127, src: 4425  dst: 2467
      TCP: Source Port = 0x1149
      TCP: Destination Port = 0x09A3
      TCP: Sequence Number = 2053738035 (0x7A698E33)
      TCP: Acknowledgement Number = 2700201127 (0xA0F1CCA7)
      TCP: Data Offset = 20 (0x14)
      TCP: Flags = 0x10 : .A....
          TCP: ..0..... = No urgent data
          TCP: ...1.... = Acknowledgement field significant
          TCP: ....0... = No Push function
          TCP: .....0.. = No Reset
          TCP: ......0. = No Synchronize
          TCP: .......0 = No Fin
      TCP: Checksum = 0x8053

00000:  00 0B DB 14 E0 6B 00 E0 18 B9 6B 0B 08 00 45 00   ..Û.àk.à.¹k...E.
00010:  00 28 D7 3F 40 00 40 06 C0 1F C0 A8 11 0A C0 A8   .(×?@.@.À.À¨..À¨
00020:  11 16 11 49 09 A3 7A 69 8E 33 A0 F1 CC A7 50 10   ...I.£ziŽ3 ñÌ§P.
00030:  FA ED 80 53 00 00                                 úí€S..
```

****

Indiquez les adresses Ethernet et les adresses IP du système de supervision et du système temps réel.

Réponse :

	PC de supervision	Système temps réel
Adresse Ethernet		
Adresse IP		

En identifiant dans la trame 13 le numéro de port source, préciser si cette trame a été émise par le serveur ou par le client.

Réponse :

Port source = **Emise par :**

Les paquets IP sont-ils fragmentés ? Justifier la réponse.

Réponse :

Paquets fragmentés : Oui / Non (entourer la réponse exacte)

Justification :

Les données spécifiques au service de l'application commence à l'octet 36 hexadécimal de la trame 13. Quel est l'identifiant du capteur concerné par cette capture ?

Réponse :

Capteur concerné =

Quel est le rôle de la trame 14 ?

Réponse :

Trame 14 :

Étude d'un système informatisé – Session 2007

Quelle information des protocoles TCP et UDP identifie sans équivoque le processus destinataire du message ?

Réponse :

Citer les trois informations définies par /etc/services pour chacun des services.

Réponse :

1)

2)

3)

Un même numéro de port peut-il être utilisé simultanément en TCP et en UDP par deux processus distincts ? Justifier votre réponse.

Réponse :

Particularités des protocoles TCP et UDP ?

Réponses :

	TCP	**UDP**
numéro et nom de couche OSI		
protocole sous-jacent		
fiabilité du transport		
séquencement des données		
mode de connexion		
taille maximale des données		
possibilité de diffusion		

Protocoles
Le réseau mis en œuvre entre le robot et le poste de supervision est de type WIFI 802.11g. Ce type de liaison radio peut s'assimiler pour le cas présent à une connexion Ethernet très peu fiable (perturbations dues à la conformation du terrain et à l'électromagnétisme).
Les données à transmettre sont des ordres pour donner une trajectoire ou pour piloter le robot en mode manuel. Ces ordres sont transmis via une transmission UDP.
La communication WIFI étant très peu fiable, un mécanisme de détection de rupture de flux est mis en place.

La détection de rupture de flux est réalisée par le mécanisme suivant :

```
// coté robot
N <- valeur aléatoire
ERREUR <- 0
FAIRE
      Envoyer N au superviseur // via protocole UDP sur le port 3333
      Attendre la réception d'un entier R du superviseur pendant X ms
      SI pas de réception dans le délai imparti
            ERREUR <-  ERREUR + 1
      SINON
            SI (  R == N )
                  ERREUR <- 0
                  N <- N + 1
            SINON
                  ERREUR <-  ERREUR + 1
            FIN SI
      FIN SI
      SI ( ERREUR >= 5 )
            Appeler procédure Robot::enCasDeRuptureDeFlux()
            N <- valeur aléatoire
      FIN SI
TANT QUE ( VRAI )
```

```
// coté poste de supervision
MEM <- 0
ERREUR <- 0
FAIRE
      Attendre la réception d'un entier R du robot pendant X ms
      SI pas de réception dans le délai imparti
            ERREUR <-  ERREUR + 1
      SINON
            SI ( ( MEM + 1 ) == R )
                  ERREUR <- 0
            SINON
                  ERREUR <-  ERREUR + 1
            FINSI
            MEM <- R
            Envoyer R au robot
      FIN SI
      SI ( ERREUR >= 5 )
            Appeler procédure Supervision::enCasDeRuptureDeFlux()
      FIN SI
TANT QUE ( VRAI )
```

Remarque : Cet algorithme sera implémenté à la question D.3.2.

Au bout de combien de tentatives infructueuses de réception de données les procédures enCasDeRuptureDeFlux() sont-elles appelées ?

Réponse :

Programmation réseau

Soit la classe DatagramSocket permettant d'envoyer et de recevoir des paquets de données via le protocole UDP :

```
class DatagramSocket
{
private:
    unsigned short port ;
    int sock ;
    struct sockaddr_in source ;

public:
    DatagramSocket( unsigned short port = 0 ) ;
    ~DatagramSocket() ;

    // fixe le timeOut en ms

    void setTimeOut( unsigned int timeOut ) ;

    // La méthode readDatagram() attend la réception d'un datagramme
    //
    // Le résultat est stocké dans data, maxlen est la taille de data
    //
    // Valeur renvoyée : la longueur du datagramme lu,
    //                   ou -1 en cas d'erreur,
    //                   ou -2 en cas de timeout

    long readDatagram( void *data, long maxlen ) ;

    long writeDatagram(
                const void *data,       // données à émettre
                long len,               // taille des données à émettre
                const char* host,       // machine cible
                unsigned short port     // port destinataire
              ) ;
};
```

Fonction de lecture d'un datagramme.

Fonction d'écriture d'un datagramme.

Le constructeur de DatagramSocket prend en argument le numéro du port local, le sauvegarde, créer une socket et l'attache si nécessaire (port > 0).

A l'aide de l'annexe 4, proposer une implémentation du constructeur DatagramSocket(unsigned short port = 0) ;

Réponse :

Au bout de la durée X ms fixée par setTimeOut(), la méthode readDatagram() renvoie -2 si aucun datagramme n'a été reçu.

En phase de mise au point des algorithmes précédents, le robot ouvre sa socket de communication sur le port 4444, tandis que le superviseur ouvre la sienne sur le port 5555.

Question D.3.2.
Compléter, d'après le pseudo-code des pages précédentes, l'implémentation de la méthode

```
void Superviseur::keepAlive(
      DatagramSocket *ds,     // Instance de Datagram pour communication
      char* robotName,        // Nom (hostname) du destinataire
      int port );             // Port à utiliser
```

permettant la détection de rupture de flux coté poste de supervision.

Réponse :

```
void Superviseur::keepAlive (Datagram* ds, char* robotName, int port)
{
      unsigned short r ;
      unsigned short mem = 0;
      int erreur = 0 ;

      do
      {
            if( ds->readDatagram( (void *)&r , sizeof(r) ) < (int)sizeof(r) )
            {
                  erreur ++ ; // pas de réception dans le délai imparti
            }
            else
            {
```


ANNEXE A4

Programmation Réseau (Syntaxe de quelques appels systèmes)

NAME

ip - Implémentation Linux du protocole IPv4.

SYNOPSIS

```
#include <sys/socket.h>
#include <netinet/in.h>

tcp_socket = socket(PF_INET, SOCK_STREAM, 0);
raw_socket = socket(PF_INET, SOCK_RAW, protocol);
udp_socket = socket(PF_INET, SOCK_DGRAM, protocol);
```

NOM

socket - Créer un point de communication.

SYNOPSIS

```
#include <sys/types.h>
#include <sys/socket.h>

int socket(int domain, int type, int protocol);
```

DESCRIPTION

Socket crée un point de communication, et renvoie un descripteur.

VALEUR RENVOYÉE

socket retourne un descripteur référençant la socket créée en cas de réussite. En cas d'échec -1 est renvoyé, et errno contient le code d'erreur.

NOM

bind - Fournir un nom à une socket.

SYNOPSIS

```
#include <sys/types.h>
#include <sys/socket.h>

int  bind(int  sockfd, struct sockaddr *my_addr, int addrlen);
```

DESCRIPTION

bind fournit à la socket sockfd l'adresse locale my_addr. my_addr est longue de addrlen octets. Traditionnellement cette operation est appelée "assignation d'un nom à une socket" (Quand une socket est créée, par l'appel-système socket(2),elle existe dans l'espace des noms mais n'a pas de nom assigné).

VALEUR RENVOYÉE

bind renvoie 0 s'il réussit, ou -1 s'il échoue, auquel cas errno contient le code d'erreur.

Étude d'un système informatisé – Session 2008

ROUTAGE IP

On donne un extrait de la table de routage du routeur du sous réseau « Expéditions ».

	Destination	Passerelle	Genmask	Iface
1:	172.16.32.0	*	255.255.224.0	eth0
2:	172.16.64.0	172.16.0.3	255.255.224.0	eth1
3:	172.16.96.0	172.16.0.4	255.255.224.0	eth1
	...			
6:	172.16.0.0	*	255.255.0.0	eth1
7:	default	172.16.255.254	0.0.0.0	eth1

Expliquer la signification des quatre colonnes de la ligne 3.

Réponse :

Destination :

Passerelle :

Genmask :

Iface :

CAPTURE D'UN ECHANGE CLIENT-SERVEUR

On a effectué la capture d'un échange entre une station et le serveur.

No.	Time	Source	Destination	Protocol	Port	Info
3	0.000136	172.16.128.5	172.16.128.1	TCP	4589 > 7505	[SYN] Seq=0 Ack=0 Win=16384 Len=0 MSS=1446
4	0.000177	172.16.128.1	172.16.128.5	TCP	7505 > 4589	[SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460
5	0 000297	172.16.128.5	172.16.128.1	TCP	4589 > 7505	[ACK] Seq=1 Ack=1 Win=17352 Len=0
6	0.000558	172.16.128.5	172.16.128.1	TCP	4589 > 7505	[PSH, ACK] Seq=1 Ack=1 Win=17352 Len=372
7	0.000609	172.16.128.1	172.16.128.5	TCP	7505 > 4589	[PSH, ACK] Seq=1 Ack=373 Win=65163 Len=372
8	0.001075	172.16.128.5	172.16.128.1	TCP	4589 > 7505	[FIN, ACK] Seq=373 Ack=373 Win=16980 Len=0
9	0.001092	172.16.128.1	172.16.128.5	TCP	7505 > 4589	[ACK] Seq=373 Ack=374 Win=65163 Len=0
10	0.001146	172.16.128.1	172.16.128.5	TCP	7505 > 4589	[FIN, ACK] Seq=373 Ack=374 Win=65163 Len=0
11	0.001370	172.16.128.5	172.16.128.1	TCP	4589 > 7505	[ACK] Seq=374 Ack=374 Win=16980 Len=0

Quel est le protocole de transport utilisé ?

Réponse :

Donner le rôle des trames 3,4 et 5 ?

Réponse :

Trame 3 :

Trame 4 :

Trame 5 :

A partir de la capture précédente donner en le justifiant, les adresses IP du serveur et du client.

Réponse :

Client :

Serveur :

Étude d'un système informatisé – Session 2010

Étude de la volumétrie

Chaque station de tramway génère des flux entrants et sortants impliquant des besoins en bande passante réseau.

Le besoin en bande passante est le rapport de la quantité de bits occupés (en émission ou en réception sur le support de transmission) sur la période de référence. La période de référence est ici de 1 seconde.

Les communications téléphoniques et les messages de sonorisation sont transportés par le réseau multiservice en utilisant la technologie de Voix sur IP (VoIP).

En VoIP, le signal sonore est numérisé et traité par un CODEC. Les échantillons issus du CODEC sont regroupés en paquets, et les paquets sont transmis périodiquement (toutes les 10 à 30ms selon la configuration choisie) via le protocole RTP (realtime transport protocol). On cherche à déterminer le besoin en bande passante Ethernet selon le CODEC utilisé. Pour cela, il est nécessaire de calculer le nombre total de bits occupés sur le support de transmission. Cela comprend les bits d'information transmis mais aussi le préambule et la période de silence imposée entre 2 trames (IFG = InterFrame Gap).

En se reportant à « Annexe 9 : Réseau Multi Service (RMS) », Compléter le tableau ci-dessous.

Réponse :

CODEC	**Taille de la trame complète** avec préambule et IFG **(en bits)**	**Nombre de trames par seconde**	**Bande passante Ethernet (en bits/s)**
G.711 (PCM)			
G.729a (CS-CELP)			

Chaque station est équipée de 4 à 6 caméras de vidéosurveillance. Ces caméras sont des caméras IP qui encodent le flux vidéo avec un codec H264. Chaque caméra produit sur sa sortie Ethernet un flux sortant de 250kbits/s. Le flux entrant d'une caméra, pour les données de contrôle, est de 10kbit/s.

Un serveur d'enregistrement vidéo, situé au PCC, reçoit les flux vidéo de l'ensemble des caméras. Le serveur d'enregistrement est raccordé au réseau par une liaison Ethernet 100BASE-TX full duplex.

Pour conserver une marge de sécurité, on limite le débit maximal exploitable à 75% du débit nominal de liaison.

Combien de caméras cette liaison permet-elle de traiter au maximum (détailler le calcul) ?

Réponse :

On envisage de diffuser une chaîne de TV dédiée « TV TRAM» sur une trentaine de bornes situées aux points d'attente en stations. Cette chaîne serait diffusée en protocole RTSP (realtime streaming protocol) sur une adresse IP multicast

Justifier le choix du multicast par rapport à l'unicast pour cette application.

Réponse :

Annexe 9 :**Réseau Multi Service (RMS)**

CODEC	Bande passante données audio	Période échantillonnage	Taille échantillon (octets)	Nb d'échantillons par paquet
G.711 (PCM)	64 kbps	30 ms	240	1
G.726 (ADPCM)	32 kbps	20 ms	80	1
G.728 (LD-CELP)	16 kbps	2.5 ms	5	4(*)
G.729a (CS-CELP)	8 kbps	10 ms	10	2(*)

(*) : Lorsque les paquets regroupent chacun **n** échantillons, la période d'émission des paquets est de **n** fois la période d'échantillonnage. Ex : en G.728, la période d'émission des paquets est de 4x2,5ms = 10ms.

Tableau An.10.1 : codecs et constitution des paquets VoIP

Débit	Norme	Câble	Longueur maxi	Coût relatif
Ethernet	10BASE5	Thick Ethernet (coax)	500m	obsolète
	10BASE-T	Twisted Pair (cat 3)	100m	Très faible
Fast Ethernet	100BASE-TX	Twisted Pair (cat 5)	100m	Très faible
	100BASE-FX	Multimode Optical Fiber	412m (half duplex)	moyen
		Multimode Optical Fiber	2000m (full duplex)	moyen
		Singlemode Optical Fiber	15km (full duplex)	élevé
Gigabit Ethernet	1000BASE-T	Twisted Pair (cat 5e ou 6)	100m	faible
	1000BASE-SX	Multimode Optical Fiber	550m	moyen
	1000BASE-LX	Singlemode Optical Fiber	5km	élevé
	1000BASE-LH	Singlemode Optical Fiber	Jusqu'à 70km	Très élevé
10 Gigabit Ethernet	10GBASE-LX4	Multimode Optical Fiber	300m	Très élevé
	10GBASE-ER/EW	Singlemode Optical Fiber	jusqu'à 40km	extrêmement élevé

Tableau An.10.2 : Caractéristiques des supports Ethernet

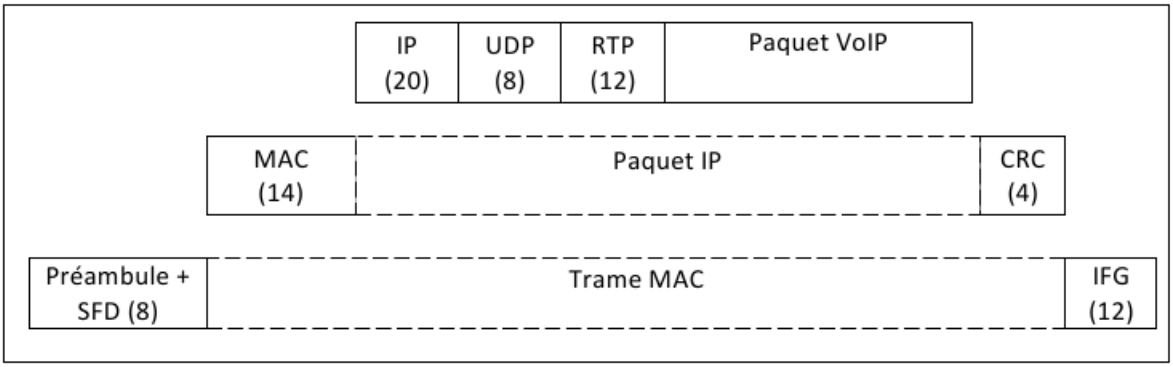

Figure An.10.3 : encapsulation VoIP sur Ethernet (Nb octets entre parenthèses)

