

Codage de trame :

Les caractères suivantes doivent être envoyés dans une trame : A; B; FLAG; ESC.

Quelle sera la séquence de caractères envoyés par une couche de liaison de données utilisant :

- a) Le Comptage de caractères.
- b) Des Octets de flag et début et fin de trame.

La séquence suivante est incluse dans un flux de données : A B ESC C ESC FLAG FLAG D. Quelle sera sa transformation si on utilise l'algorithme de "byte-stuffing" ?

Débit effectif et taille de trames

Une ligne de transmission a un taux d'erreur bit de 10^{-3} en moyenne. Un protocole de niveau 2 utilise des trames de 250 octets.

- a) Quel est le pourcentage de trames erronées ?
- b) Quel est le débit effectif moyen si 100 trames sont envoyées en 2 s ? (On peut considérer que l'efficacité du protocole est 1)
- c) Quel est le débit effectif moyen si les trames ont une taille de 53 octets ?

Protocole à retransmissions sélectives.

Dans le cas d'un média de mauvaise qualité, il est nécessaire d'adapter le protocole DLL afin de minimiser l'impact des retransmissions dues aux erreurs sur la bande passante. Une stratégie possible pour la gestion des erreurs et des retransmissions est d'implanter une fenêtre permettant la réception de N trames.

L'algorithme ci-joint (protocole 6), implémente ce protocole de liaison de données.

- ✓ a) Quelle est la procédure suivie lors de la réception d'un NAK ?
- ✓ b) Quelle est la procédure suivie lors de la réception d'une trame erronée (erreur de checksum) ?
- ✓ c) Quelles sont les données de protocoles transmises dans une trame ?
- d) Lors de la réception d'une nouvelle trame :
 - ✓ a. Dans quel cas un nack est-il envoyé ?
 - ✓ b. Dans quel cas le timer start_ack_timer est-il lancé ? quelle est son utilité ?

CSMA/CD

Un réseau local utilise la technologie CSMA/CD à une vitesse de 10 Mbps. La vitesse de propagation dans le medium est de 200m/ μ sec ; pas de répéteur. La taille des trame est de 256 octets, dont 32 octets d'en-tête et de données de protocole. La tranche de temps (9.6 μ s) suivant la transmission d'une trame de données, est réservée à l'émission de acquittement par le récepteur. Chaque trame d'acquittement est composée d'un entête complet (32 octets).

- ✓ a) Quelle est le débit utile de ce protocole ?
- b) Quelle est la distance maximale entre deux terminaux afin d'assurer une bonne détection des collisions ?

IP / Routage

Le schéma suivant (fig. 1) représente un réseau IP constitué de sept routeurs IP, de 8 réseaux locaux IP, et d'un maillage d'interconnexion des routeurs. Les adresses des réseaux sont les suivantes :

Réseau A : 112.0.0.0

Réseau E : 193.57.96.0

Réseau B : 123.0.0.0

Réseau F : 193.57.97.0

Réseau C : 163.44.0.0

Réseau G : 193.252.95.0

Réseau D : 113.0.0.0

Réseau H : 114.0.0.0

De quelle classe d'adresses IP les réseaux A,C,F,H font-ils partie ?

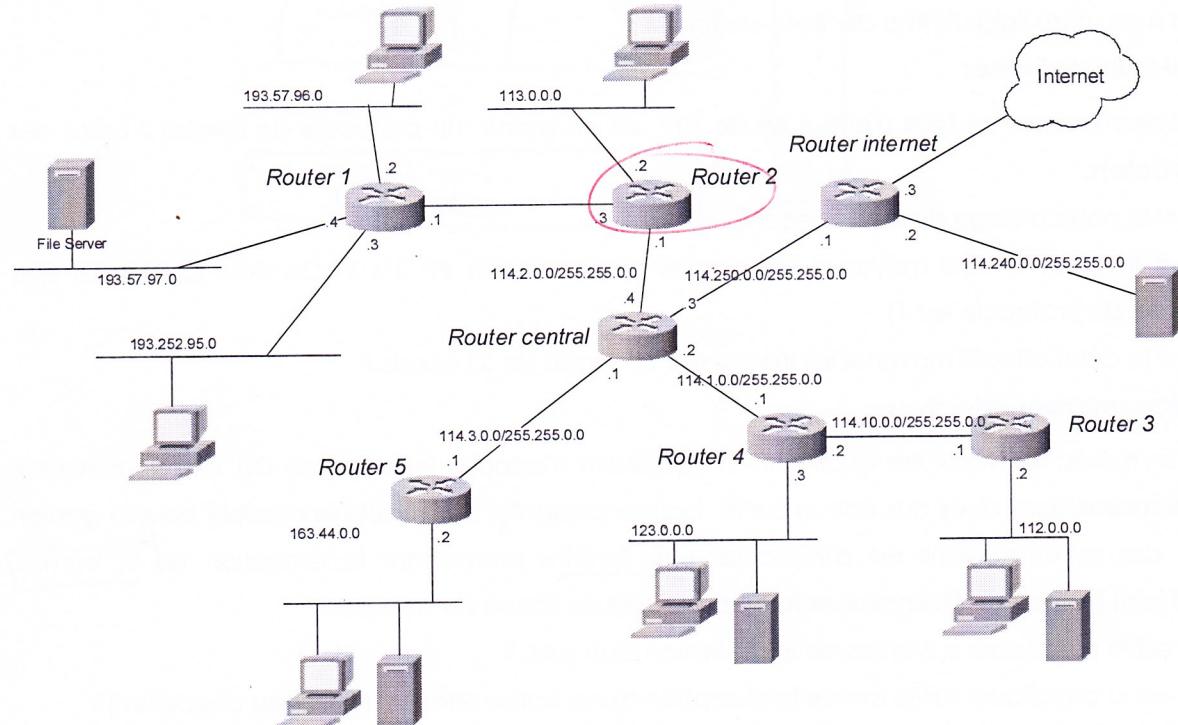


Figure 1

Les routeurs sont configurés pour utiliser le protocole RIP (Routing Information Protocol).

Décrivez l'évolution de la table de routage du routeur n°2.

Combien de temps faut-il pour que celui-ci ait pris connaissance du réseau ?

Le routeur 4 tombe en panne; sachant que la durée de vie d'une ligne dans les tables de routage est de 60 secondes, Combien de cycles du protocole RIP faudra-t-il pour que la table de routage du routeur Internet soit à jour ?

l'ingénieur réseau connecte le réseau 123.0.0.0 et sur le routeur 3, et le routeur 3 sur le routeur central.

Combien de cycles du protocole RIP faudra-t-il pour que la table de routage du routeur Internet soit à jour ?

L'ingénieur titulaire du réseau 163.44.0.0 souhaite découper le plan d'adressage en 16 sous-réseaux.

Quelle solution peut-il utiliser ?

Quel est le netmask associé ?

Si on utilise le netmask 255.255.252.0, dans quel sous réseaux se trouve la machine titulaire de l'adresse 163.44.195.10?

ANNEXE 1 :

```

/* Protocol 6 (selective repeat) accepts frames out of order but passes packets to the
network layer in order. Associated with each outstanding frame is a timer. When the timer
expires, only that frame is retransmitted, not all the outstanding frames, as in protocol 5. */

#define MAX_SEQ 7
#define NR_BUFS ((MAX_SEQ + 1)/2)
typedef enum {frame_arrival, oksum_err, timeout, network_layer_ready, ack_timeout, event_type};

boolean no_nak = true;
seq_nr_oldest_frame = MAX_SEQ + 1;

static boolean between(seq_nr_a, seq_nr_b, seq_nr_c) {
    /* same as between in protocol 5, but shorter and more obscure */
    return ((a <= b) && (b < c)) || ((b < a) && (a <= c));
}

static void send_frame(frame_kid tk, seq_nr frame_nr, seq_nr frame_expected, packet_buffer*) {
    /* Construct and send a data, ack, or nak frame */
    frame_s;
    /* scratch variable */

    s.kind = tk;
    /* kind == data, ack, or nak */
    if (tk == data) s.info = buffer[frame_nr % NR_BUFS];
    /* s.seq = frame_nr;
    s.ack = (frame_expected + MAX_SEQ) % (MAX_SEQ + 1);
    if (tk == nak) no_nak = false;
    to_physical_layer(&s);
    if (tk == data) start_timer(frame_nr % NR_BUFS);
    stop_ack_timer();
    */
}

void protocol6(void) {
    /*seq_nr ack_expected;
    seq_nr next_frame_to_send;
    seq_nr frame_expected;
    seq_nr too_lar;
    int k;
    frame_r;
    packet_out buf[NR_BUFS];
    packet_in buf[NR_BUFS];
    boolean arrived[NR_BUFS];
    seq_nr_nbffered;
    event_type event;
    enable_network_layer();
    ack_expected = 0;
    next_frame_to_send = 0;
    frame_expected = 0;
    too_far = NR_BUFS;
    nbffered = 0;
    for (i = 0; i < NR_BUFS; ++i) arrived[i] = false;
    while (true) {
        wait_for_event(&event);
        /* five possibilities: see event type above */
    }
}

switch(event) {
    case network_layer_ready: /* accept, save, and transmit a new frame */
        nbffered = nbffered + 1;
        /* expand the window */
        from_network_layer(&out_buf, buf); /* fetch new packet */
        send_frame(data, next_frame_to_send, frame_expected, out_buf); /* transmit the frame */
        incnext_frame_to_send();
        break;
    case frame_arrival: /* a data or control frame has arrived */
        from_physical_layer(&r);
        if (r.kind == data) {
            /* An undamaged frame has arrived */
            if ((r.seq != frame_expected) && no_nak) {
                send_frame(rak, 0, frame_expected, out_buf); /* start ack timer */;
                if (between(frame_expected, r.seq, too_lar) && (arrived[r.seq % NR_BUFS] == false)) {
                    /* frames may be accepted in any order */
                    arrived[r.seq % NR_BUFS] = true; /* mark buffer as full */
                    in_buf[seq % NR_BUFS] = r.info; /* insert data into buffer */
                    while (arrived[frame_expected % NR_BUFS]) {
                        /* Pass frames and advance window */
                        to_network_layer(&in_buf[frame_expected % NR_BUFS]);
                        no_nak = true;
                        arrived[frame_expected % NR_BUFS] = false;
                        /* advance lower edge of receiver's window */
                        incframe_expected();
                        incloop_fan: /* to see if a separate ack is needed */
                        start_ack_timer();
                    }
                }
            }
            /* if (r.kind==nak) && between(ack_expected(r.ack+1)%MAX_SEQ+1) next frame to send */
            send_frame(data, (r.ack+1) % (MAX_SEQ + 1), frame_expected, out_buf);
        }
        while (between(rack_expected, r.ack, next_frame_to_send)) {
            nbffered = nbffered - 1; /* handle piggybacked ack */
            stop_timer(rack, /* frame arrived intact */
            incrack_expected); /* advance upper edge of receiver's window */
            break;
        }
        /* case cksum_err:
        if (no_nak) send_frame(nak, 0, frame_expected, out_buf); /* damaged frame */
        break;
        */
        case timeout:
        send_frame(data, oldest_frame, frame_expected, out_buf); /* we timed out */
        break;
        /* case ack_timeout:
        send_frame(rak, 0, frame_expected, out_buf); /* ack timer expired; send ack */
        */
    if (nbffered < NR_BUFS) enable_network_layer(); else disable_network_layer();
}
}

```

- **Zero** - This field is not actually used by [RFC 1058](#) RIP; it was added solely to provide backward compatibility with prestandard varieties of RIP. Its name comes from its defaulted value: zero.
- **Address-family identifier (AFI)** - Specifies the address family used. RIP is designed to carry routing information for several different protocols. Each entry has an address-family identifier to indicate the type of address being specified. The AFI for IP is 2.
- **Address** - Specifies the IP address for the entry.
- **Metric** - Indicates how many internetwork hops (routers) have been traversed in the trip to the destination. This value is between 1 and 15 for a valid route, or 16 for an unreachable route.

Note: Up to 25 occurrences of the AFI, Address, and Metric fields are permitted in a single IP RIP packet. (Up to 25 destinations can be listed in a single RIP packet.)

RIP 2 Packet Format

The RIP 2 specification (described in [RFC 1723](#)) allows more information to be included in RIP packets and provides a simple authentication mechanism that is not supported by RIP.

[Figure: An IP RIP 2 Packet Consists of Fields Similar to Those of an IP RIP Packet](#) shows the IP RIP 2 packet format.

[Figure: An IP RIP 2 Packet Consists of Fields Similar to Those of an IP RIP Packet](#)

1-octet command field	1-octet version number field	2-octet unused field	2-octet AFI field	2-octet route tag field	4-octet network address field	4-octet subnet mask field	4-octet next hop field	4-octet metric field
-----------------------	------------------------------	----------------------	-------------------	-------------------------	-------------------------------	---------------------------	------------------------	----------------------

The following descriptions summarize the IP RIP 2 packet format fields illustrated in [Figure: An IP RIP 2 Packet Consists of Fields Similar to Those of an IP RIP Packet](#):

- **Command** - Indicates whether the packet is a request or a response. The request asks that a router send all or a part of its routing table. The response can be an unsolicited regular routing update or a reply to a request. Responses contain routing table entries. Multiple RIP packets are used to convey information from large routing tables.
- **Version** - Specifies the RIP version used. In a RIP packet implementing any of the RIP 2 fields or using authentication, this value is set to 2.
- **Unused** - Has a value set to zero.
- **Address-family identifier (AFI)** - Specifies the address family used. RIPv2's AFI field functions identically to [RFC 1058](#) RIP's AFI field, with one exception: If the AFI for the first entry in the message is 0xFFFF, the remainder of the entry contains authentication information. Currently, the only authentication type is simple password.
- **Route tag** - Provides a method for distinguishing between internal routes (learned by RIP) and external routes (learned from other protocols).
- **IP address** - Specifies the IP address for the entry.
- **Subnet mask** - Contains the subnet mask for the entry. If this field is zero, no subnet mask has been specified for the entry.
- **Next hop** - Indicates the IP address of the next hop to which packets for the entry should be forwarded.
- **Metric** - Indicates how many internetwork hops (routers) have been traversed in the trip to the destination. This value is between 1 and 15 for a valid route, or 16 for an unreachable route.

Note: Up to 25 occurrences of the AFI, Address, and Metric fields are permitted in a single IP RIP packet. That is, up to 25 routing table entries can be listed in a single RIP packet. If the AFI specifies an authenticated message, only 24 routing table entries can be specified. Given that individual table entries aren't fragmented into multiple packets, RIP does not need a mechanism to resequence datagrams bearing routing table updates from neighboring routers.

Plan salle TP:

2) Codage de trame

a)	<table border="1"> <tr> <td>S</td><td>A</td><td>B</td><td>Flag</td><td>Esc</td></tr> </table>	S	A	B	Flag	Esc
S	A	B	Flag	Esc		

bit de longueur

b)	<table border="1"> <tr> <td>Flag</td><td>A</td><td>B</td><td>ESC</td><td>Flag</td><td>ESC</td><td>ESC</td><td>Flag</td></tr> </table>	Flag	A	B	ESC	Flag	ESC	ESC	Flag
Flag	A	B	ESC	Flag	ESC	ESC	Flag		

X juste mettre flag au début et à la fin sans bytes stuffing.

c)

A B Esc C Esc Esc Esc Flag Esc Flag D

2) Débit effectif & taille de trame

a) 1 trame = 250 octets = 2000 bits

proba qu'une trame soit entièrement correcte

$$\left(\frac{999}{1000}\right)^{2000}$$

proba qu'une trame soit erronée = $1 - \left(\frac{999}{1000}\right)^{2000} = 86,5\%$

b) 100 trames en 2 s = 200 000 bits = 100 kb/s

c) $\frac{53 \times 100 \times 8}{2} = \frac{42400}{2} = 21200 \text{ b/s} = 21,2 \text{ kb/s}$

Protocole à retransmises sélectives

a) Lors de la réception d'un NACK, on renvoie la trame qui suit la dernière trame acquittée.

b) Lors de la détection d'un checksum error, on envoie un ACK indiquant le num de la trame erronée.

- c) - num séquence
 - type
 - num d'acquittement

d)

- a- Envie d'un Nack lorsque :
 - checksum error
 - récept^e mauvaise trame & pas de nack en cours
- b- Start ack timer est appellé lors de la récept^e d'une bonne trame. Il sert à renvoyer la dernière trame en cas de non réponse.
 Il sert à renvoyer un ACK si le dernier n'a pas été reçu.

CSNA/CD

a) 1 trame = 256 o = 2048 bits
~~donc~~ ~~256 bits~~

$$\frac{2048}{10 \text{ M}} = 2,048 \mu\text{s} \text{ temps d'envoie d'une trame}$$

$$2,048 + 0,6 = 2,648 \mu\text{s pour un échange.}$$

$$\frac{4096}{10 \text{ M}} \times 8 = \frac{32 \times 8}{10 \text{ M}} = 256 \cdot 10^{-7} = 25,6 \mu\text{s}$$

$$\text{Tps total} = 214,4 + 25,6 = 240 \mu\text{s}$$

$$\frac{2048 + 256}{240 \cdot 10^{-6}} = 9,6 \text{ Mb/s}$$

7,5 Mb/s
 car charge utile
= sans entêtes

b) en $9,6 \mu\text{s}$, il faut faire un aller-retour :

$$\frac{200 \times 9,6}{2} = \cancel{1920} \rightarrow 960 \text{ m.}$$

car aller-retour en $9,6 \mu\text{s}$.

IP/ROUTAGE

a)

A : classe A

C : classe B

F : classe C

H : classe A

b)

itérat° 0 : rien

	IP	dist.
itérat° 1 :	113.0.0.0	0

itérat° 2 :	113.0.0.0	0
	193.57.96.0	1
	193.57.97.0	1
	193.25.95.0	1

#

... - 2 - ...

...