Institut Galilée — Systèmes d'exploitation, Réseaux
Année 2014-2015 — Licence 2

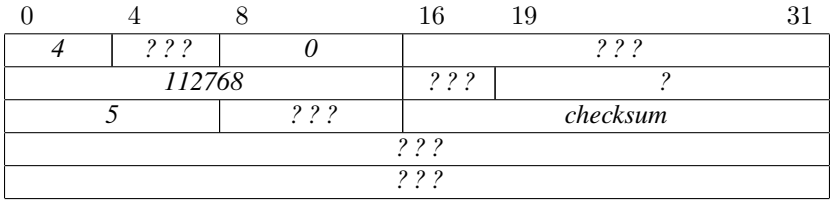
Contrôle Systèmes d'exploitation, Réseaux

Mardi 19 Mai 2015 — 9h - 12h
Aucun document n'est autorisé

Exercice 1 (4 = 2 + 2)

1. *Donnez dans chacun des cas suivants l'adresse du réseau, du sous réseau, le numéro de la machine pour les configurations suivantes :*
   (a) *Adresse IP = 209.123.31.6 ; masque de réseau = 255.255.255.240*
   (b) *Adresse IP = 23.96.108.109 ; masque de réseau = 255.192.0.0*
2. *Une entreprise s'est vu affecter l'adresse IP 197.151.0.0 pour une gestion plus fine de ses sous réseaux, le responsable informatique désire pouvoir affecter une adresse IP propre à chaque sous réseau des 10 succursales.*
   (a) *De quelle classe s'agit-il ?*
   (b) *Donner et expliquer la valeur du masque de sous réseau correspondant à ce besoin.*
   (c) *Combien de machines chaque sous réseau pourra-t-il comporter et pourquoi ?*
   (d) *Quelle est l'adresse de diffusion (broadcast) du sous réseau 3 ?*

Exercice 2 (3,5 = 3 + 0,5) *Une station X souhaite transmettre à une autre station Y un datagramme TCP dont la taille (incluant l'entête TCP) est égale à 2000 octets. La machine X est dans un réseau A alors que Y est dans un réseau C. L'adresse de X est 111.33.55.88, celle de Y est 124.15.170.166. La MTU du réseau A est égale à 1500 octets. Le datagramme envoyé par X quitte le réseau A en passant par un routeur R1, il atteint le réseau B de MTU = 2048 octets. Il passe ensuite par un routeur R2 pour atteindre le réseau C., dont la MTU est égale à 1468 octets. La structure de l'en-tête IP du datagramme dans le réseau A est présentée ci-dessous :*



1. *Déterminez les paquets IP (en précisant l'en-tête de chaque paquet) qui passent dans les réseaux A, B et C. (La somme de contrôle de l'en-tête n'est pas à calculer)*
2. *Pourquoi le réassemblage de paquets IP n'est-il réalisé que par le destinataire final et pas par un noeud intermédiaire ?*

Exercice 3 (2,5 = 1 + 1 + 0,5)

1. *Préciser le fonctionnement du protocole UDP*
2. *Préciser le fonctionnement du protocole TCP*
3. *Quelle est la différence entre IPv4 et IPv6 ?*

Exercice 4 (5 = 2,5 + 2,5)

On considère pour cet exercice le programme utilisant les primitives parbegin/parend donné ci-dessous.

1. *Donner le programme correspondant de parallélisme maximal, en C, avec un processus par tâche et des tubes pour les communications (ne mettez pas les fichiers en-tête nécessaires).*
2. *Donner le programme correspondant de parallélisme maximal en C, avec un thread par tâche et des mutex pour la synchronisation d'exécution (ne mettez pas les fichiers en-tête nécessaires).*

Programme avec primitives parbegin/parend *:*

```
begin
  lire(a) ;
  lire(b) ;
  parbegin
    c := a*a |
d := a*b
  parend ;
  e := a + b ;
  c := c + d ;
  e := c + d + e
end
```

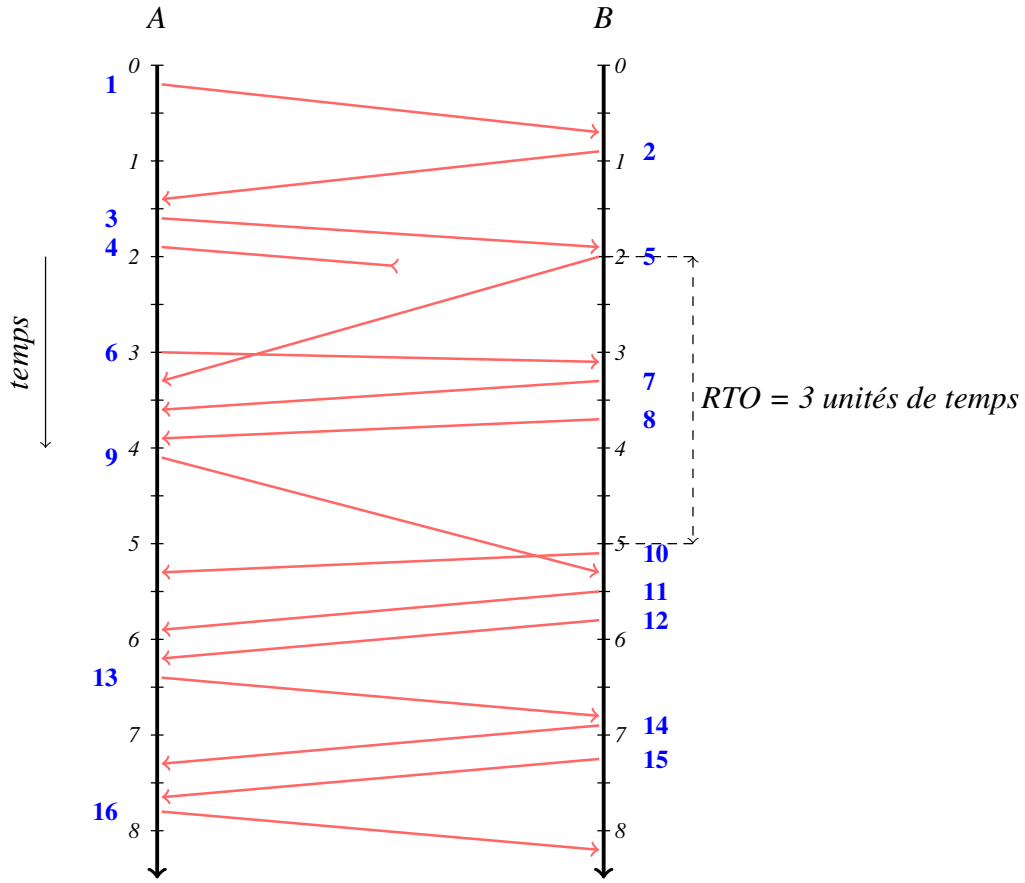
Exercice 5 (5 = 3 + 2)

Nous considérons une connexion TCP entre deux stations A et B. A doit envoyer 200 octets, B va envoyer 600 octets. Pour le transfert de A vers B, la taille des données d'un segment est fixée à 100 octets. Pour le transfert de B vers A, la taille des données d'un segment est fixée à 200 octets.

La transmission complète (initialisation, transfert, fin) est représentée dans la figure 1 par un chronogramme. Complétez les paramètres associés à chaque segment (Numéros de séquence et d'acquittement, flags et taille des données) dans un tableau dont les deux premières lignes sont données ci-dessous :

#	Num Seq	Num Ack	ACK	SYN	FIN	Taille données
1	1032	–		X		0
2	8066	1033	X	X		0

Décrivez précisément les cas d'erreurs et les protocoles de récupération mis en jeu.



(Annexe 1) Fonctions et structures utiles :

```
pid_t fork(void);
int pipe(int tableau[2]);

int pthread_create(pthread_t *thread,
                   const pthread_attr_t *attr,
                   void *(*start_routine)(void*), void *arg);
int pthread_join(pthread_t thread, void **value_ptr);
void pthread_exit(void *value_ptr);

int pthread_mutex_lock(pthread_mutex_t *mutex);
int pthread_mutex_unlock(pthread_mutex_t *mutex);
int pthread_mutex_init(pthread_mutex_t *mutex, const pthread_mutexattr_t *attr);
int pthread_mutex_destroy(pthread_mutex_t *mutex);

int open(const char *pathname, int flags);
int close(int fd);
ssize_t read(int fd, void *buf, size_t count);q
ssize_t write(int fd, const void *buf, size_t count);
int getchar(void);
```

(Annexe 2) Structure d'un paquet IPv4 :