BLOT William CHEVOLLEAU Marin DAUDIN Clément JOUBERT Hugo LE GALL Tristan

Info 3 - Réseaux 2 - Conception de protocoles de communication

Protocole d'Architecture Tokenisé pour Réseau Interne avec Contrôle Kalité (PATRICK)

Exercice 1 :

Rappel du cahier des charges

- Problématique : spécifier et implémenter un protocole réseau permettant la communication entre des machines liées entre elles sous forme d'un anneau (les machines sont simplement chaînées circulairement).
- Pré-requis : topologie en anneau, un seul paquet à la fois sur le réseau.

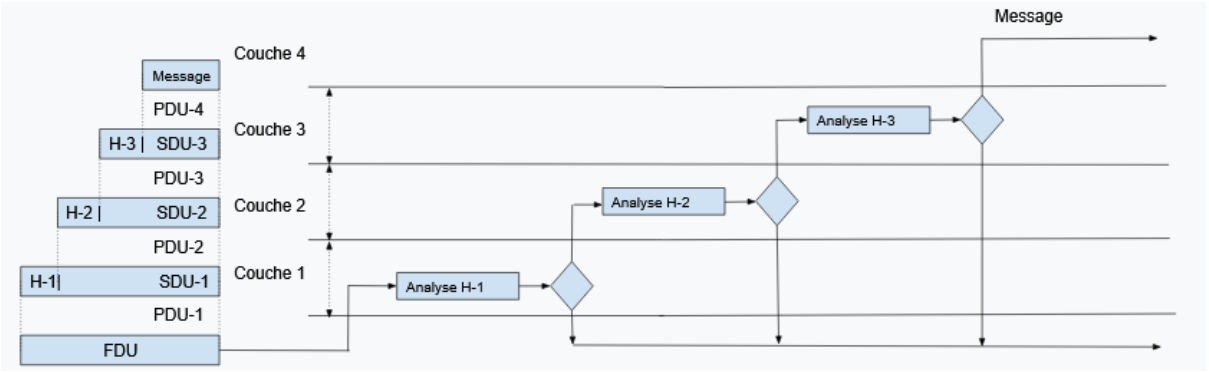
Description du PDU

Token 1 bit	@Emetteur 6 bits	@Destinataire 6 bits	Flag 2 bits	Taille message 8 bits	Message ←→ Variable Max: 256 bits	CI 5 bits message%29
Liaison (couche 1)	Réseau (couche 2)			Application (couche 3)		

- Le token est un nombre défini sur 1 bit indiquant si la machine peut envoyer un nouveau message ou si elle doit en transmettre un.
- Les adresses d'émetteur et de destinataire sont définies sur 6 bit et pas plus, car il est peu probable qu'un grand nombre de machines soient interconnectées via un réseau en anneau.
- Le flag défini sur 2 bits permet d'identifier la nature du paquet : envoi/transmission de message (00), acquittement de message (01) ou acquittement d'erreur (10).

- La taille du message est définie sur 8 bit et indique la taille du message. Cela permet ainsi de réduire la taille théorique des paquets pour des messages courts ou au contraire de l'augmenter pour des messages longs.
- Le CI (contrôle d'intégrité) est un hash du message. Il permet au destinataire de vérifier que le message n'a pas été altéré au cours de la transmission. Une bonne fonction de hachage serait de prendre la valeur du message et de lui appliquer un modulo 29, pour obtenir un résultat stockable sur 5 bits et un risque modéré de collision inopinée.

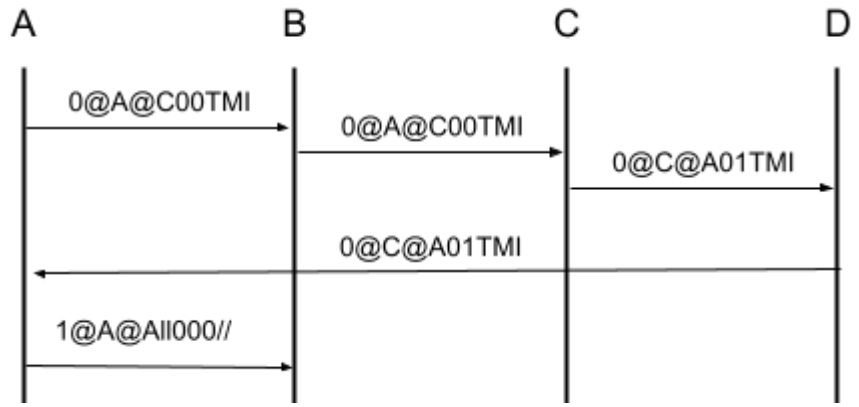
Traitement logique par les différentes couches de la pile



Diagrammes de séquence du protocole

Lorsqu'un hôte reçoit un acquittement sans erreur de son message. Il peut le dépiler de sa liste de messages (si les hôtes ont plusieurs messages à envoyer).

<u>Scénario nominal de transmission d'un paquet entre A et C :</u>



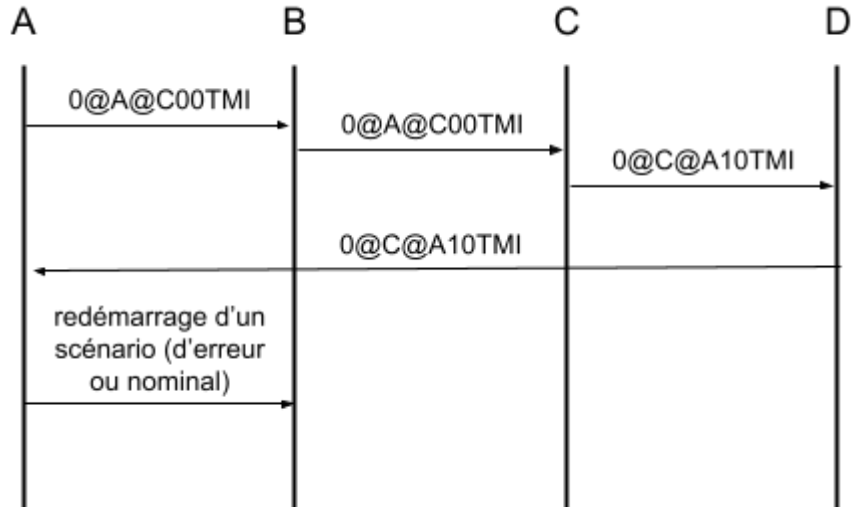
Le message est constitué de l'indicateur de disponibilité du token (0) l'adresse de départ (@A), de l'adresse de destination (@C), des flags d'erreurs et d'acquittement dans ce même ordre (00 → envoi/transmission d'un message sans erreurs, puis 01, acquittement sans erreur). À cela, s'ajoutent la taille du message (T), le message (M), ainsi que le contrôleur d'intégrité (I).
Dans un premier temps, le message est transmis de A vers C en passant par B. Dès lors, un acquittement est renvoyé (le flag d'ACK passe à 1) vers A en passant par D. Une fois

l'information reçue, le token est envoyé vers l'adresse suivante (All → B) avec un flag d'acquittement à 0 et une longueur de message vide.

Exemple d'une transmission nominale d'un paquet entre A et C :

PC Courant	Paquet						
	Token	@Emetteur	@Destinataire	Flag	Taille Message	Message	Contrôle d'intégrité
A	0	A	C	00	T	M	I
B	0	A	C	00	T	M	I
C	0	C	A	01	T	M	I
D	0	C	A	01	T	M	I
A	1	A	All	00	0	/	/
B	Si B veut le Token il le prend, sinon il transmet le message.						

Scénario d'erreur de transmission d'un paquet entre A et C :



La Transmission de A vers C est identique au scénario nominal, mais un flag d'erreur est ajouté au moment du retour, car le message reçu par C n'est pas correct. Le paquet avec l'indication d'erreur est renvoyé vers A qui va alors réexécuter l'envoi.

Exemple d'une transmission erronée d'un paquet entre A et C :

PC Courant	Paquet						
	Token	@Emetteur	@Destinataire	Flag	Taille Message	Message	Contrôle d'intégrité
A	0	A	C	00	T	M	I
B	0	A	C	00	T	M	I
C	0	C	A	10	T	M	I
D	0	C	A	10	T	M	I
A	A renvoie le message à C (scénario nominal ou erroné)						

Algorithme (pseudo-code)

Étape 1 : Construire la première trame comme vu ci-dessus
(i.e @source, @destinataire, flag initialisé à 00, le message M à transmettre et μ la valeur du hash code du message M)

Étape 2 : Émettre le paquet sur le réseau
Tant Que @pc_courant != @destinataire OU @pc_courant = @pc_emetteur :
On laisse passer le paquet et le transmet au pc suivant
Fin Tant Que

Étape 3 : Lecture du paquet
Si flag == 00 :
Flag ← 01
hc ← On calcule le hash code du message reçu
Si hc (hash code réel) != μ (hash code théorique) :
flag ← 10
Fin Si
Echanger @pc_emetteur et @pc_destinataire
Retour à l'étape 2
Sinon flag == 10 :
Retour à l'étape 1 (On renvoie le message)
Fin Sinon
Sinon flag == 01 :
Token ← 1
@pc_destinataire ← @broadcast
Envoie de nouveau sur le réseau
Fin Sinon
Fin Si

Étape 4 : Prochain pc_emetteur de message
    Parcours dans le sens de la topologie du réseau
    Si pc_courant a un message à transmettre
        Token ← 0
        Retour à l'Étape 1
    Fin Si

Description de l'algorithme

Le principe de notre protocole est que lorsque le PC source envoie un message, à un PC destinataire, on transmet dans la trame (outre le message en lui-même et sa taille), un flag qui est sur 2 bits. Ce flag est l'ACK (accusé de réception).
Lorsque le PC destinataire a reçu le message qui lui est destiné (égalité entre l'adresse du destinataire dans la trame et son adresse sur le réseau), le message est renvoyé avec un flag à 01 s'il n'y a pas eu d'erreur, indiquant que le message a bel et bien été reçu (intact).

Pour détecter une erreur : on inclut dans la trame un hash code du message (réalisé par le PC source) et on compare le hash code reçu par le destinataire avec le hash code du message haché reçu. Si ces deux valeurs coïncident, il n'y a pas eu d'erreur et le champ flag du paquet renvoyé est 01. Si on reçoit le message, mais cette fois-ci avec erreur, le flag du paquet renvoyé est 10 (acquittement, mais avec erreur).

Par la suite, comme le token (i.e. ce qui permet à un PC de notre réseau d'envoyer un message) n'a pas été transmis, on boucle jusqu'à notre PC source avec la même trame à l'exception du flag, désormais à 01 (dans le cas sans erreur). Une fois le PC source atteint, il diffuse le token en broadcast et son voisin direct (dans le sens du parcours de la topologie de notre réseau) "décide" de le prendre ou non, en fonction de s'il a un message à envoyer.

NB : en cas d'erreur, on recommence l'envoi du message du même PC source vers le même PC destinataire.