

Exercice

Question 1

Supposez que le réseau ne nous garantit plus l'absence de perte de message. Quelles sont alors les garanties données par les différentes classes de fiabilité ? Pour répondre, notez pour chacun des 4 protocoles, s'il y a ou non risque de

- absence de traitement d'une requête
- duplication de traitement d'une requête
- absence de confirmation d'une requête
- duplication de confirmation de traitement

et si oui, un exemple de scénario qui peut le causer.

Si de nouveaux problèmes surgissent pour RR avec id et RRA, quelle solution proposez-vous ?

Question 2

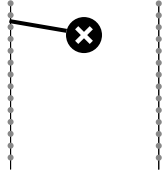
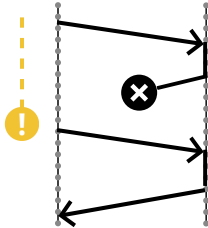
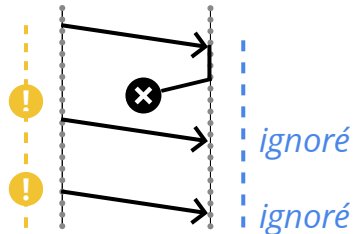
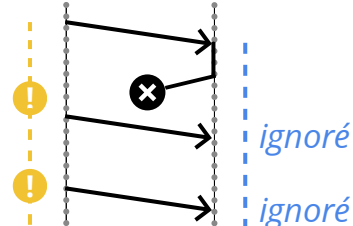
Que se passe-t-il dans RR version "exactement une fois" si le traitement est très long ?

Question 3

Que se passe-t-il dans RR version "exactement une fois" si le client tombe en panne temporairement avant d'avoir reçu une réponse ? Quelle solution proposer ?

Exercice

Solution 1

	Absence traitement	Duplication traitement	Absence confirmation	Duplication confirmation
R		<p>Impossible :</p> <p>Le client n'envoie jamais en double.</p>	<p>Inapplicable :</p> <p>Le client ne reçoit jamais de confirmation.</p>	<p>Inapplicable :</p> <p>Le client ne reçoit jamais de confirmation.</p>
RR	<p>Impossible :</p> <p>Le client réessaie après timeout</p>		<p>Impossible :</p> <p>Le client réessaie après timeout</p>	<p>Possible, mais pas à cause d'une perte de message.</p>
RR avec id	<p>Impossible :</p> <p>Le client réessaie après timeout</p>	<p>Impossible :</p> <p>L'id assure l'absence de doublon coté serveur</p>		<p>Impossible :</p> <p>l'identificateur supprime les doublons</p>
RRA	<p>Impossible :</p> <p>Le client réessaie après timeout</p>	<p>Impossible :</p> <p>L'id assure l'absence de doublon coté serveur</p>		<p>Impossible :</p> <p>l'identificateur supprime les doublons</p>

Exercice

Solution 1 (continued)

RR avec id et RRA ont un risque d'absence de confirmation en cas de perte de la réponse.

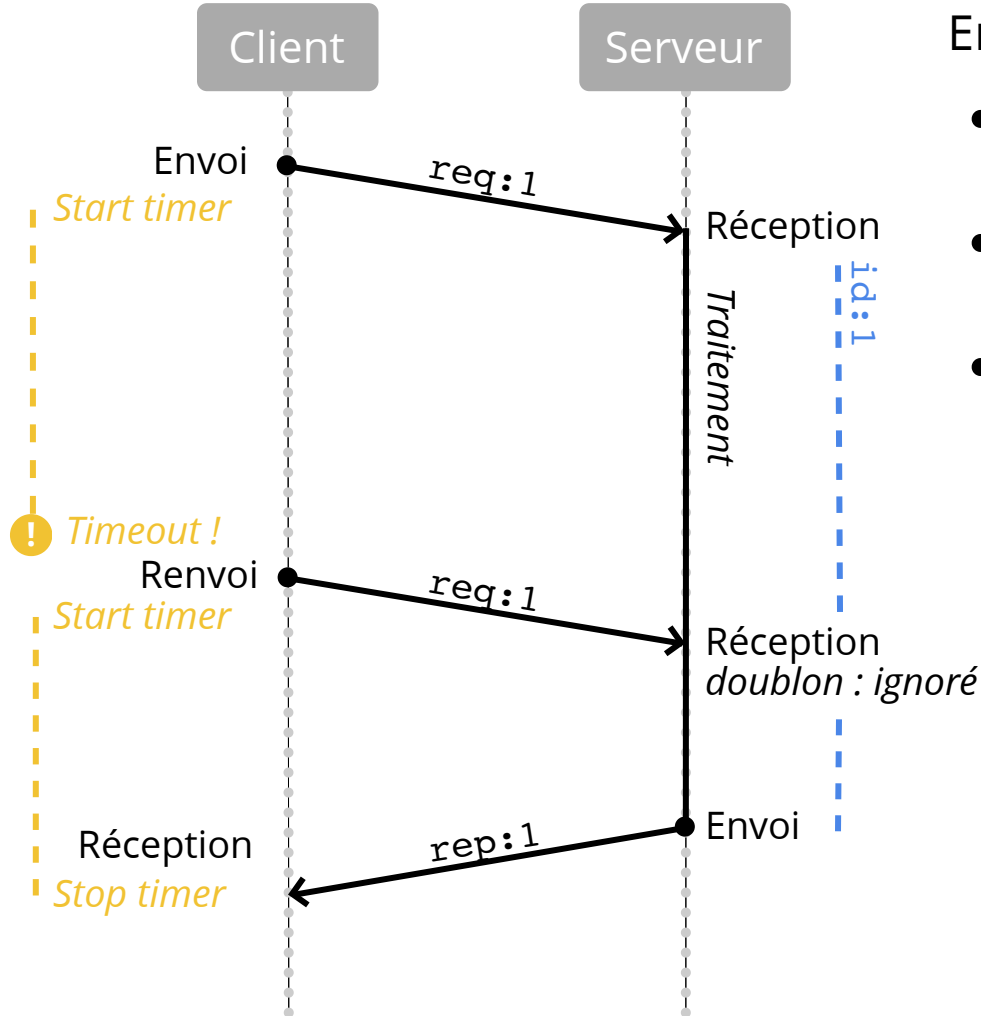
Le problème est que le serveur se reposait sur la garantie du réseau pour ignorer les doublons. Il doit maintenant renvoyer la réponse, même s'il pense être face à un doublon.

Afin d'éviter le cout de recalculer la réponse, celle-ci peut être stockée en cache avec l'id du message. Ceci sera couteux en mémoire pour RR avec id, mais résolu avec le système d'ack de RRA.

Noter aussi le problème de la perte du ACK dans RRA : l'id (et la réponse) ne seront jamais supprimés chez le serveur. La conséquence n'est cependant pas un échec fatal du serveur ou du protocole.

Exercice

Solution 2

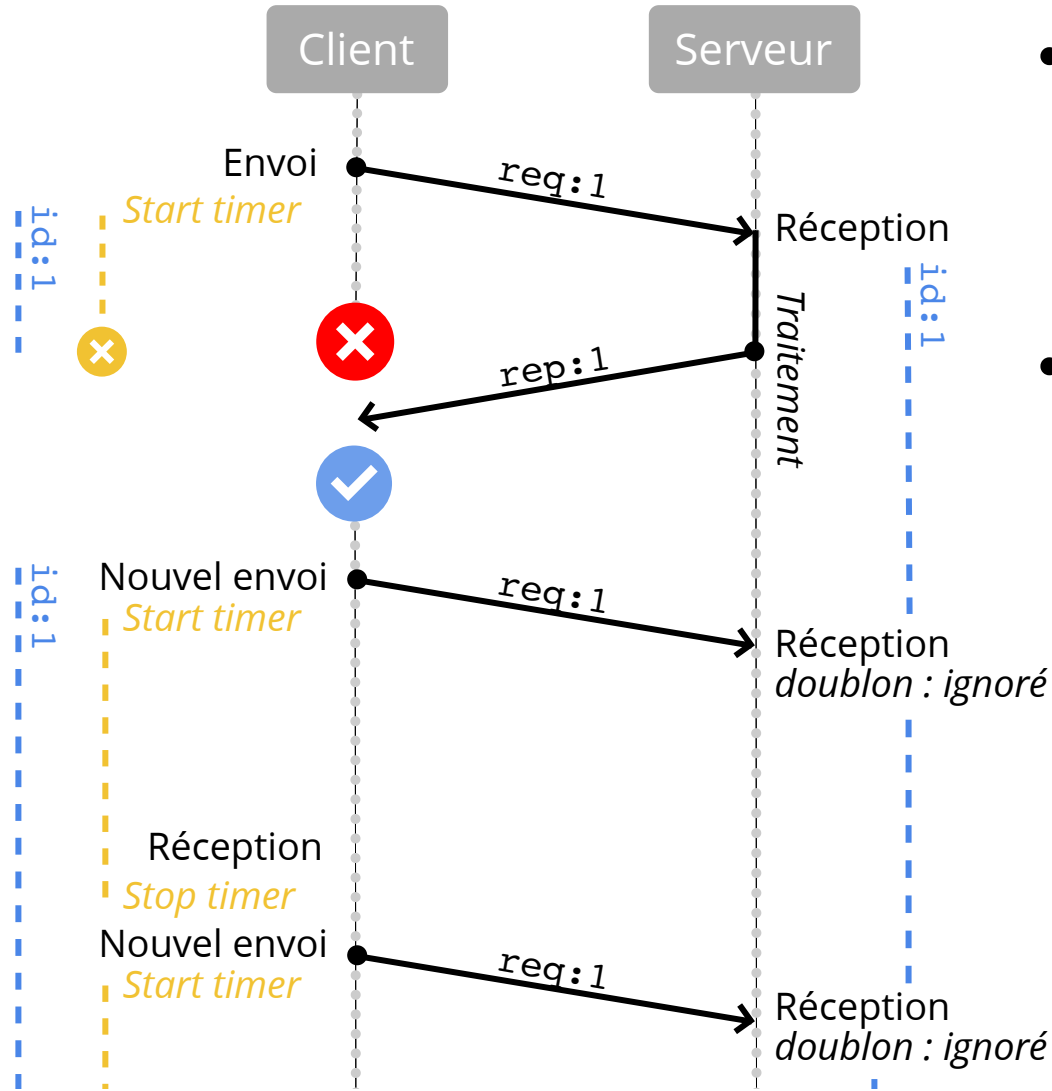


En cas de traitement long :

- Le client timeout et renvoie la requête.
- Le serveur utilise l'id pour détecter le doublon et l'ignore.
- La réponse du serveur est perçue comme réponse à la seconde requête du client et stoppe le timer.

Exercice

Solution 3

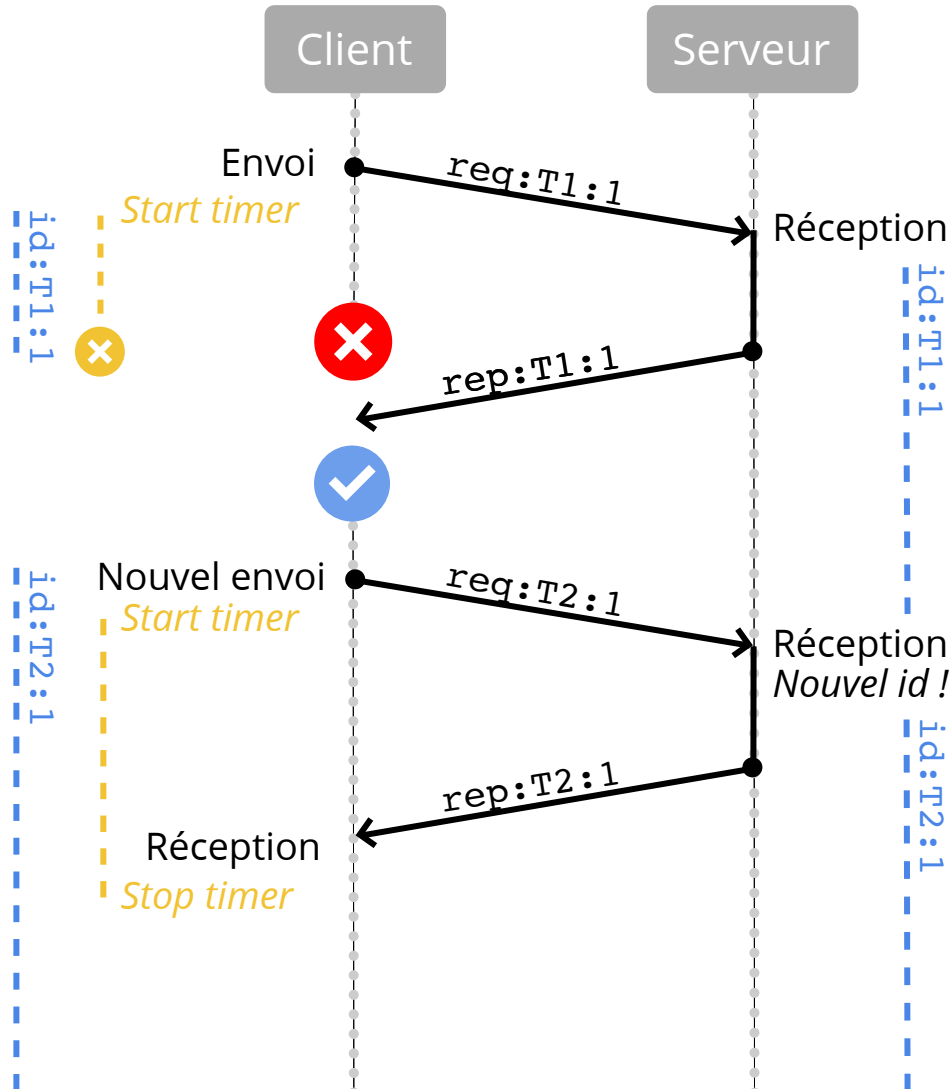


En cas de panne du client :

- Lorsqu'il revient, s'il veut envoyer un nouveau message (potentiellement indépendant du précédent), celui-ci sera ignoré par le serveur.
- Le client réessaiera infiniment, entrant dans une boucle infinie.

Exercice

Solution 3 *continued*



Solution

Le client doit envoyer un identifiant unique de l'exécution en cours. Ainsi, les ids seront différents lors de la seconde exécution, et le serveur ne croira pas à un doublon. Une manière de garantir cette unicité des identifiants peut être d'y accoler l'heure du début de son exécution.