



CSC4509 : Algorithmique répartie et communications entre processus distants

Revision : 1499

Denis Conan, avec Éric Lallet

Avril 2021





Plan de la présentation

1. Communications au cœur des systèmes répartis
2. Algorithmique répartie
3. Objectifs du module
4. Organisation du module
5. Organisation des études d'articles de recherche
6. Évaluation du module

1 Communications au cœur des systèmes répartis

■ Communications (réparties)

⇒ espace mémoire réparti sans unicité de temps

- Réseaux transmettant l'information au mieux (en anglais, *best effort*), donc :
 - Débits et délais variables ; pertes, duplications, et ré-ordonnancements de paquets ; bruits attaquant l'intégrité des données
- Machines et réseaux hétérogènes (matériel et logiciel)

■ Propriétés physiques des communications

- Délais de lecture et d'écriture de plusieurs ordres de grandeur plus importants que ceux d'une mémoire locale
- Entraves multiples : dépendances nombreuses vis-à-vis de services distants potentiellement défaillants, administrations différentes, etc.
- Bande passante utile limitée : fonction du bruit et des contre-mesures pour le respect de l'intégrité des données

⇒ Nombre important de paramètres et variabilité importante

1.1 P.ex. sémantique des op. de lecture et d'écriture

- « **Exactement une fois** » : c'est la sémantique idéale car le client peut ignorer le fait que son appel est distant ou non
 - Mais, comme le client et le service sont indépendants, cela est par principe impossible à garantir dans tous les cas à cause par exemple des défaillances
 - Les mises en œuvre fournissant cette sémantique approchent le cas idéal en ajoutant de la surveillance mutuelle
- « **Au plus une fois** » : si le client ne reçoit pas de réponse en un temps donné, l'appel revient en erreur, indiquant que le service ne peut pas être rendu
 - La mise en œuvre est moins triviale qu'il n'y paraît, par exemple, si c'est le message de réponse qui est perdu alors que le service a été rendu
- « **Au moins une fois** » : si le client ne reçoit pas de réponse en un temps donné, retransmission autant de fois que nécessaire jusqu'à la réception d'une réponse
 - Le processus rendant le service doit tolérer les appels redondants

1.1.1 Illustration avec OASIS MQTT pour l'IoT

- MQTT v.5.0: an OASIS standard in Dec. 2019
 - *For Machine-to-Machine (M2M)/Internet of Things (IoT)*
 - *Use of the publish/subscribe message pattern which provides one-to-many message distribution and decoupling of applications*
 - *Three qualities of service for message delivery :*
 1. *“At most once” : Message loss can occur. This level could be used, for example, with ambient sensor data where it does not matter if an individual reading is lost as the next one will be published soon after*
 2. *“At least once” : Messages are assured to arrive but duplicates can occur*
 3. *“Exactly once” : Messages are assured to arrive exactly once. This level could be used, for example, with billing systems where duplicate or lost messages could lead to incorrect charges being applied*
- Ces propriétés sont garanties dans le cadre de sessions
 - Une session = un ensemble de connexions

1.1.2 Plus précisément, sémantique de l'acquittement

- Qu'est-ce qu'un acquittement ?
 1. Le traitement demandé par le message a été effectué
 2. Le traitement du message est accepté et sera effectué à terme, c.-à-d. *in fine*
 3. La temporisation associée au message envoyé peut être arrêtée
 4. Les zones mémoires associées au message envoyé peuvent être libérées
 5. Un autre message peut être émis
 6. Etc.

2 Algorithmique répartie

- Une application répartie est un ensemble de processus
 - Avec dans ce module
 - Les processus qui sont modélisés comme des machines à états
 - = Un état initial
 - + une séquence d'actions donnant une séquence d'états
 - Les processus communiquent par échange de messages
- La répartition est à la base de la tolérance aux fautes
 - Lorsqu'un service est défaillant, un autre « prend le relais »
 - Elle repose sur l'étude des algorithmes répartis de base étudiés et mis en œuvre
 - La problématique de la tolérance aux fautes est étudiée dans le cadre d'études d'articles de recherche (en quadrinôme)

2.1 P. ex., communication de groupe ?

Il existe des cas où un processus doit envoyer le même message ou la même suite de messages à tous les autres processus : c'est la diffusion.

- Solution de départ : on utilise *IP multicast* ou de multiples envois TCP-UDP
 - **Pb 1** : Que se passe-t-il si un processus (l'émetteur ou l'un des récepteurs) défaille pendant la diffusion d'un message ?
 - **Pb 2** : Supposons une application répartie de réservation de places d'avions. Un client réserve des places (1^{er} message) puis supprime sa réservation (2nd message).
Que se passe-t-il si le premier message est perdu ou non encore arrivé/traité ?
 - **Pb 3** : Supposons une application de tchat. Un premier utilisateur annonce la réservation d'une table dans un restaurant pour ce soir 20h.
Un second utilisateur reçoit ce 1^{er} message et répond avec uniquement l'heure de réservation changée dans sa réponse (2nd message).
Que se passe-t-il si un troisième utilisateur reçoit le 2nd message sans le 1^{er} ?
 - Etc.

2.2 P. ex., terminaison d'une application répartie

Comment arrêter une application répartie dont l'algorithme de chaque processus est structuré en une boucle infinie d'attente de message en provenance du réseau ?

- Solution de départ : on exécute des commandes `ssh` sur les différentes machines pour arrêter les processus
 - **Pb 1** : C'est trop violent si chaque processus doit laisser le système de fichiers de la machine hôte dans un certain état
 - **Pb 2** : Quand bien même cette première solution serait acceptable, comment sait-on que c'est terminé ?
 - **Pb 3** : Quand bien même un processus serait capable de dire qu'il ne fait plus rien, que fait-on des messages en transit dans les canaux de communication ?
 - **Pb 4** : Et quid des processus qui créent d'autres processus, c'est-à-dire d'une application répartie dont la « configuration » évolue constamment ?
 - Etc.

3 Objectifs du module

- À l'issue du module, vous serez capables, dans le cadre du développement d'une application répartie de petite taille, mais réaliste, et dont le cahier des charges est fourni, de :
 1. **développer entièrement une application répartie** (par exemple un tchat multicient multiserveur) avec TCP/IP en utilisant la bibliothèque JAVA NIO avec des communications en modes connecté et déconnecté, et en modes synchrone et asynchrone,
 2. **lire, donner l'exécution**, et expliquer les preuves **d'algorithmes répartis** de base (élection, diffusion, exclusion mutuelle, interblocage, et détection de terminaison),
 3. **mettre en œuvre les algorithmes répartis** étudiés par insertion dans l'application répartie développée dans le module et avec écriture de scénarios de tests simples (c.-à-d. sans utilisation de canevas logiciel d'émulation ou de simulation) pour vérifier les propriétés de correction et de progression.
- Par ailleurs, vous serez aussi capables de :
 4. **étudier** en groupe des articles de recherche sur la thématique de **la tolérance aux fautes** pour en faire une présentation synthétique à la classe.

4 Organisation du module I

1. Concevoir et mettre en œuvre une application répartie

- Mise en contexte + abstraction en concepts + devoir maison
- Diapositives présentées en cours intégrés
- Application des concepts avec JAVA NIO en TP
- Travail personnel : finir les TPs, lire des tutoriels, **compléter le devoir maison**

Soutenance individuelle à partir du devoir maison

4 Organisation du module II

2. Comprendre un algorithme réparti

- Mise en contexte + abstraction en propriétés + exercices et QCM
- Classe inversée : cours en auto-apprentissage et exercices en classe
- Travail personnel : auto-apprentissage puis **QCM** après chaque séance d'exos

3. Mettre en œuvre et tester un algorithme réparti dans une application répartie

- Réalisation d'une étude de cas en binôme
- Fonctionnement en mode projet : **code + rapport (compte rendu des TPs)**

4 Organisation du module III

4. Présenter un article de recherche en tolérance aux fautes

http://www-inf.telecom-sudparis.eu/COURS/CSC4509/?page=articles_recherche

- Étude d'un état de l'art par groupe de 4
- Étude et préparation avec un tuteur
- **Présentation devant la classe en fin de module**

Évaluation individuelle de la présentation avec la compréhension, et aussi de la préparation en tutorat

5 Organisation des études d'articles de recherche

1. Mise à disposition des **articles en ligne** pour effectuer vos choix
2. Sondage pour émettre vos préférences (\approx mardi 5 mai)
3. Constitution des **quadrinômes** à partir du sondage
 - Entre 1^{ers} arrivés/1^{ers} servis et affectation par le coordinateur (\approx lundi 11 mai)
4. Travail en équipe avec **tutorat**, par exemple
 - 1^è rencontre avec le tuteur pour poser des questions sur l'article (\approx mardi 19 mai)
 - 2^è rencontre avec le tuteur pour discuter des diapositives de la présentation (\approx mardi 3 juin)
 - 3^è rencontre avec le tuteur pour répéter la présentation avant de la donner devant la classe (\approx mardi 8 juin)
5. **Livraison des diapositives le mercredi soir 10 juin**
6. **Présentation devant la classe les jeudi 11**
 - **30mn de présentation et 15mn de questions/réponses avec discussion**
 - **Présence obligatoire de tous**

6 Évaluation du module

- Note finale =
 - Devoir maison en monôme avec soutenance individuelle sur la programmation réseau avec JAVA NIO : 8 points
 - Le barème est construit principalement sur la soutenance : 6 points
 - Projet en binôme avec soutenance individuelle sur la mise en œuvre d'algorithmes répartis : 9 points
 - Code + rapport de réalisation en binôme
 - Étude d'un article de recherche en tutorat et avec présentation devant la classe : 3 points
 - Le barème est construit sur la préparation, la présentation ainsi que la compréhension de l'article
- **La présence en cours et la participation sont nécessaires**