Modélisation et construction des applications réparties

Introduction aux algorithmes de consensus (DUT Informatique - M4102C)

J. Christian Attiogbé

Février 2015, maj 2022

IUT Nantes
Pôle Sciences et technologie

Analyse du système réparti de surveillance de zone

Exemple de problème

Plusieurs processus préleveurs envoient des valeurs de capteurs à un contrôleur qui doit prendre une décision et agir sur l'environnement physique contrôlé.

- Que se passe-t-il si les valeurs transmises par les processus ne sont pas cohérentes ?
  (par exemple, dans le cas de températures prélevées, que doit faire le contrôleur si un préleveur envoie une température négative et l'autre une température positive).

Ce problème sous sa forme généralisée, est connu ; il est récurrent dans les systèmes répartis : problème des généraux byzantins.
`fr.wikipedia.org/wiki/Probleme_des_generaux_byzantins`

Problème des généraux byzantins

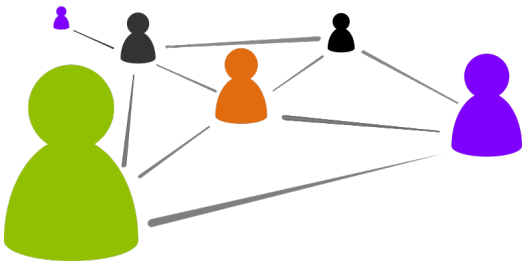


Figure: Echange - interaction dans une appli sur le réseau

Une armée avec plusieurs généraux assiège une ville ennemie. Les généraux (un général et ses lieutenants) commandent chacun un campement ; ils échangent des messages entre eux et doivent élaborer un plan de bataille commun pour éviter la défaite.

Problème des généraux byzantins

Une armée avec plusieurs généraux assiège une ville ennemie. Les généraux (un général et ses lieutenants) commandent chacun un campement ; ils échangent des messages entre eux et doivent élaborer un plan de bataille commun pour éviter la défaite.

Le problème est que les généraux ne communiquent pas deux à deux ; de plus il y a parmi eux des traîtres (byzantins) qui veulent empêcher les généraux loyaux de s'accorder pour prendre la bonne décision.

La question est de trouver une méthode (un algorithme) pour s'assurer que les généraux loyaux arrivent tout de même à se mettre d'accord sur un plan de bataille.

Problème des généraux byzantins

Il a été démontré (Lamport, 1982) qu'en utilisant uniquement des messages oraux, ce problème des généraux byzantins peut être résolu, si et seulement si plus des deux tiers des généraux (messagers) sont loyaux (nb loyaux > 2/3 généraux).
Ainsi, un seul traître peut confondre deux généraux loyaux.

De plus, le problème peut être résolu pour un nombre quelconque de généraux traîtres si les messages sont écrits (et donc non falsifiables).

Applications courantes, aujourdhui

Solutions et algorithmes largement utilisés dans les bases de données réparties, les services Web, les *blockchains*, etc

Analyse et position du problème

- Chaque général est vu comme un processus ;
  il a une valeur à transmettre aux autres processus distants.
- Après la transmission de toutes les valeurs par les processus, on veut :
  un accord sur la même valeur transmise par chacun des n généraux (un vecteur de n valeurs).
- Les valeurs sont valides, chaque valeur représente bien la valeur envoyée par le processus.
- Les processus corrects doivent avoir la même perception de l'état global (consensus).
- Il s'agit de résoudre ainsi le problème de la diffusion cohérente dans les applications réparties.

Traiter la non fiabilité des transmissions et de la non intégrité des interlocuteurs

Les algorithmes OM et SM de L. Lamport

- *Oral Message* : OM
- *Signed Message* : SM

Les algorithmes proposés par Lamport sont basés sur la relation entre le nombre de processus et le nombre de messages échangés entre eux.

The Byzantine Generals Problem,
Leslie Lamport, Robert Shostak, Marshall Pease;
ACM Transactions on Programming Languages and Systems; Vol. 4, No. 3, juillet 1982

Algorithme (Cas "oral message" - OM)

Considérons 1 général et 3 lieutenants (dont un seul est byzantin/traître : il retransmet un message erroné).
On numérote ces généraux de 0..3 (général = 0, lieutenants = 1, 2, 3).

- Le général (G0) envoie un ordre V aux 3 lieutenants (L1, L2, L3)
- Chaque lieutenant a un vecteur et y note la valeur reçue de son émetteur (à l'indice correspondant) ; notation $V_0.V_1.V_2.V_3$.
- Chaque lieutenant non byzantin renvoie aux autres lieutenants la valeur effective reçue (V_i)
- Un lieutenant byzantin renvoie une mauvaise valeur ($\neq V_b$) aux autres lieutenants.

Hypothèse : les messages mettent un temps limité pour arriver à destination, sinon ils sont considérés perdus.

Algorithme (suite OM)

- **G0 : diffuse V à ses lieutenants** (L1, L2, L3)
- Chaque lieutenant loyal qui a reçu un message V, diffuse le même message V aux autres lieutenants.
- Chaque lieutenant byzantin qui a reçu un message V, diffuse un autre message (X, Y, ...) aux autres lieutenants.

Analyse

Combien de messages en tout sont échangés ?
Terminaison de l'algorithme (pour prendre une décision) ?

Algorithme (suite OM)

Résultat de l'algorithme (si le lieutenant L2 est byzantin)
G0 : a la valeur V
Après les émissions par chacun des 4 généraux,
le vecteur de chacun des trois lieutenants est :
L1 : V.V.X.V
L2 : V.V.V.V
L3 : V.V.Y.V
Les généraux loyaux (L1 et L3) on une **majorité de** V ;
☞ la bonne décision sera prise.
Ainsi, par consensus sur l'état global, on arrive à prendre une bonne décision. (*Imaginer des mises à jour réparties, des disques, etc*)

Montrer que cela ne marche pas avec 3 généraux dont 1 traître.

Utilisation de la solution au problème des généraux

Nombreuses applications dans les applications réparties :

- Diffusion asynchrone avec **pannes byzantines**,
- Système de gestion de transactions
  (**bases de données réparties**, réplications)
  *A transaction is a collection of operations on the state of an object (database, object composition, etc.) that satisfies the **ACID properties***
- ...

ACID = *Atomic, Consistency, Isolation, and Durability*

Exercice : Algorithme (cas SM)

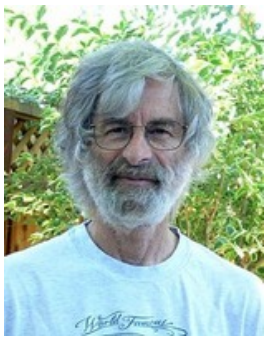


Figure: Leslie Lamport

Application avec le protocole blockchain

Block

Une structure avec un contenu, la référence à un block précédent, une empreinte (*hash*)

A toute modification, le hash change donc le block est changé (n'est plus le même) !

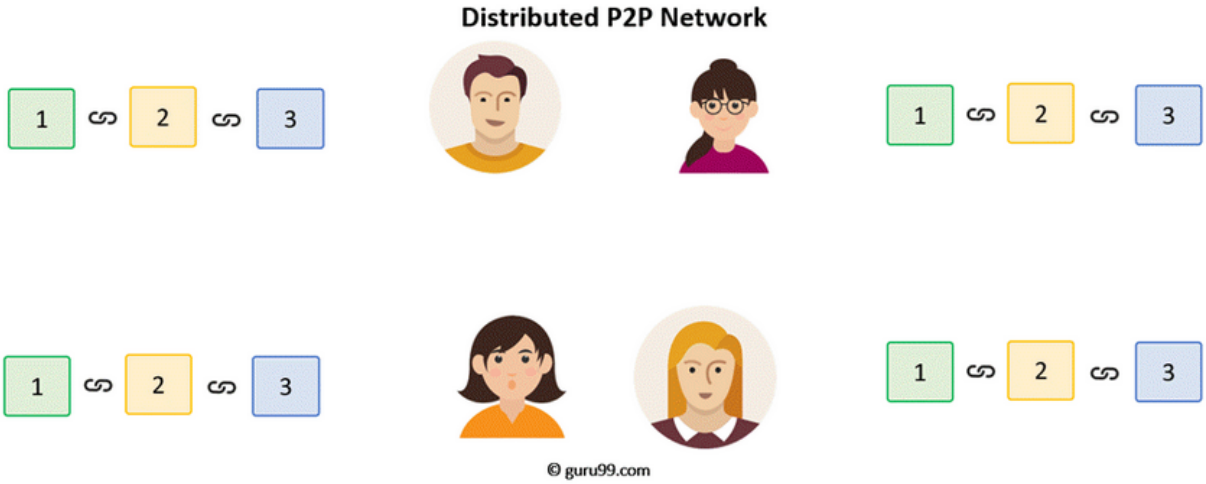
Blockchain

Une chaîne de *blocks*, stockée sur un noeud de réseau (réparti)

source : www.guru99.com/blockchain-tutorial.html

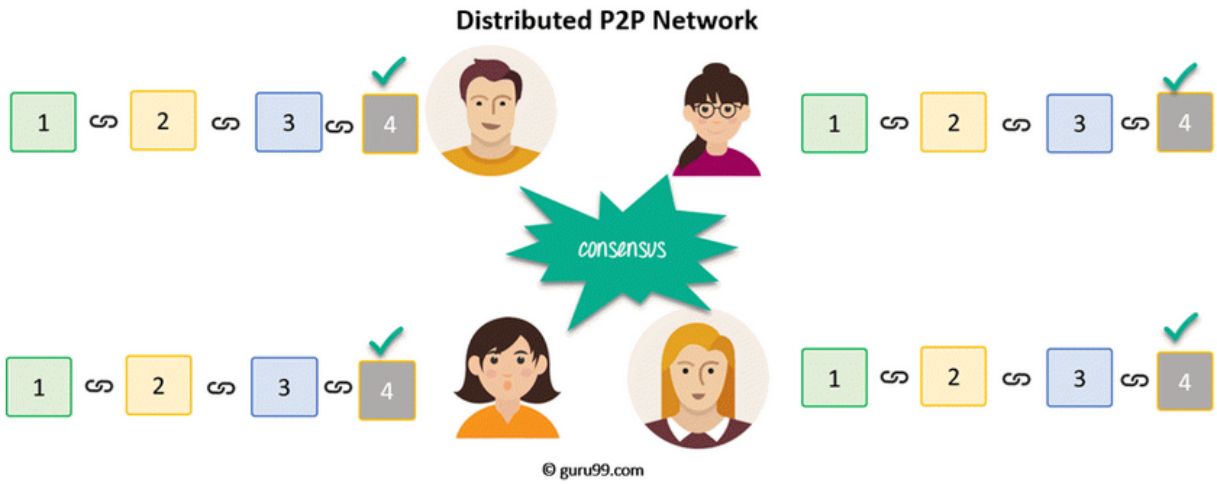
Application avec le protocole blockchain

Etat du réseau et des données réparties (blockchain de données).



Application avec le protocole blockchain

Ajout de donnée/block (mise à jour de la chaine sur tout le réseau)



à condition que par consensus le block soit jugé valide.

Conclusion

Difficultés de modélisation et construction des applications réparties

- Nommage et accès aux applications
- Architecture des applications
- Sécurité et authentification
- Disponibilité de l'application
- Tolérance aux pannes
- Passage à grande échelle

Outils, techniques, méthodes, ...

- Modèle Client/Serveur
- socket, RPC, RMI, CORBA
- Modèles à événements
- Middleware (localisation, communication, transactions, sécurité, ...)
- Web-Services
- ...

Références

- *The Byzantine Generals Problem, Leslie Lamport, Robert Shostak, Marshall Pease; ACM Transactions on Programming Languages and Systems; Vol. 4, No. 3, juillet 1982*
- Distribued Systems pinciple and paradigms, Andrew Tanenbaum, Marteen Van steen
- Cours ”Les pannes Byzantines”, Johanne Cohen, LORIA/CNRS, Nancy
- Tutoriel blockchain `https://www.guru99.com/blockchain-tutorial.html`