Page de couverture



IUT Nancy Charlemagne

Université de Lorraine

2 ter boulevard Charlemagne

BP 55227 54052

Nancy Cedex

Département informatique

Implémentation du protocole de membership SWIM à un logiciel d’édition collaborative de texte

Rapport de stage de DUT informatique

Entreprise : LORIA

Equipe COAST

Mendez-Porcel Tom

Tuteur : Matthieu Nicolas

Année universitaire 2019-2020

Remerciements

Table des matières

[Introduction 6](#_Toc42088775)

[Contexte 7](#_Toc42088776)

[MUTE 7](#_Toc42088777)

[Problème 7](#_Toc42088778)

[Etat de l’art 9](#_Toc42088779)

[Systèmes distribués (CRDT) 9](#_Toc42088780)

[Le protocole SWIM 9](#_Toc42088781)

[Prototype 10](#_Toc42088782)

[Première version 10](#_Toc42088783)

[Deuxième version 10](#_Toc42088784)

[Intégration dans MUTE 11](#_Toc42088785)

[Conclusion 12](#_Toc42088786)

[Bibliographie 13](#_Toc42088787)

# Introduction

Présentation LORIA et COAST

Explication TRES vague du sujet

Annonce du plan

# Contexte

## MUTE

MUTE (Multi-Users Text Editing) est une application web qui, comme son nom l’indique, permet à plusieurs utilisateurs d’éditer un même document texte depuis un navigateur. Pour cela, tous les utilisateurs sont connectés en pair à pair, MUTE rentre donc dans le domaine des “systèmes distribués”. Nous reviendrons plus en profondeur dans la prochaine section sur les notions de pair à pair et de systèmes distribués, mais nous allons tout d’abord exhiber le problème qui justifie mon stage.

## Problème

Pour introduire ce problème, on considère la situation suivante :

* 2 clients connectés (en pair à pair) communiquent

Client 2

Client 1

* Cependant, chaque client connait uniquement son état (ici ok en vert ou ko en rouge) donc voici le point de vue du client 1

?

Client 2

Client 1

* Ce qui est schématisé, c’est que le client 1 sait uniquement qu’il envoie des messages au client 2 (sans savoir s’il les reçoit) et qu’il en reçoit du client 2.

Donc si le client 1 ne reçoit plus de messages du client 2, plusieurs cas sont possibles :

Client 2

Pas de réponse

Client 1

Le message envoyé par le client 1 n’a pas été reçu par le client 2 ou inversement

Client 2

Client 1

Ou

Le client 2 est ko

Client 1

Client 2

Client 1

Le client 2 a répondu, mais sa réponse a été retardée et le client 1 ne l’a pas encore reçu

Client 2

Cette situation met en évidence plusieurs choses :

1. Un client ok peut très bien être vus par un autre comme ko
2. Un client qui ne répond à un message n’est pas forcément ko

Cependant, dans la majorité des logiciels, on ne peut pas se permettre de considérer un client ko alors qui ne l’est pas et il est important de détecter les clients ko pour ne plus les contacter inutilement (et MUTE ne fait pas exception).

Pour cela, on utilise des protocoles appelés protocoles de membership

Un protocole de membership est justement un protocole qui va permettre de suivre l’état des différents clients qui font partie du réseau (quel qu’il soit). L’objectif du stage est justement d’implémenter le protocole de membership SWIM (sur lequel ont reviendras dans la prochaine partie) à MUTE.

Ce stage à donc été séparé en 3 parties : une partie de documentation pour appréhender les concepts, une partie de développement d’un prototype et enfin une partie d’intégration dans MUTE.

# Etat de l’art

Durant la première semaine de stage, je me suis uniquement documenté sur les domaines que j’allais être amené à appréhender pour mon stage.

Pour cela, mon tuteur avait préparé plusieurs documents sur lesquels je pouvais me baser (cf bibliographie)

Nous allons donc dans un premier temps évoquer les bases à connaitre pour rentrer dans le sujet.

## Systèmes distribués

La notion de système distribué est généralement associée à la réalisation d’une tâche par plusieurs machines (ou « node » en anglais), cette tâche peut-aussi bien être de stocker des données, de réaliser un calcul ou de faire fonctionner un logiciel.

## Conflits et CRDT

## Protocole de membership

Comme évoqué dans la partie précédente, un protocole de membership est un protocole qui permet de suivre l’état des clients d’un réseau dans le cadre des systèmes distribués.

Pour bien comprendre, prenons l’exemple d’un protocole très simple, le « heartbeat protocol » : chaque machine va envoyer à chaque période T un message à toutes les autres uniquement pour les notifier que cette machine est toujours ok.

Ce protocole permet donc bien de suivre l’état de toutes les machines du réseau :

* Les machines qui m’ont envoyé un message depuis T sont ok toutes les autres sont ko

Cependant, ce protocole comporte plusieurs problèmes :

1. Comme évoqué plus tard, une machine qui n’arrive pas à m’envoyer de message n’est pas forcément ko (connexion retardée, routage lent)
2. Tous les clients n’auront pas forcément la même liste de machines connectées
3. L’augmentation du nombre de messages sur le réseau augmente de manière quadratique
4. …

Transition

## Le protocole SWIM

Intro…

SWIM va donc permettre de suivre l’état des machines du réseau, mais aussi de répandre les informations concernant l’arrivée ou le départ d’une machine (le départ d’une machine étant légèrement différent de son échec)

Nous allons maintenant expliquer le fonctionnement de SWIM en détaillant les étapes d’une période :

Si B répond à A (ack direct), B est ok et A attends la prochaine période.

Toutes les machines choisissent une autre machine à qui elles envoient un message (ping). Soit A la machine qui cherche à joindre B.

Si B ne répond pas à A, A choisi *n* autres machines qui sont désignées pour pinger la machine initiale (ping-req). Soit C et D 2 machines désignées par A pour joindre B

Si ni C, ni D ne peuvent joindre B, la procédure se termine avec B ko

Si C ou D parvient à joindre B (ack indirect), la procédure se termine avec B ok.

A l’issu de cette période, certaines machines peuvent être déclarées ko. Plutôt que de les exclure directement du réseau, SWIM propose un état intermédiaire ou la machine est « suspecte ». Si une machine suspecte ne parvient pas à montrer qu’elle est ok, elle vient par être déclarée défaillante et est exclue du réseau.

Pour faire parvenir à toutes les machines les informations relatives au statut des clients dans le réseau, SWIM réutilise les messages ping, ping-req et ack auxquels il rajoute les informations Alive, pour une machine ok, Suspect pour une machine suspecte et Confirm pour une machine ko.

Cette réutilisation (le terme exact est « piggyback ») permet de ne pas générer de messages superflu et de garder un nombre bien déterminés 0de messages circulant sur ce réseau.

# Prototype

Sous-intro (techno utilisées…)

## Première version

Plutôt axé fonctionnalité

## Deuxième version

Plutôt axé bonne conception

# Intégration dans MUTE

?

# Conclusion

Objectifs atteint au moment de rendre le rapport et pronostic sur la fin du stage

Difficultés rencontrées

# Bibliographie