Jouez à Faire Consensus Avec MITTEN (démonstration)

Çagdas Bozman, Mohamed Iguernlala, Michael Laporte, Maxime Levillain, Alain Mebsout, Sylvain Conchon

Jouez à Faire Consensus Avec Mitten

Çagdas Bozman[1], Mohamed Iguernlala[1], Michael Laporte[1], Maxime Levillain[1], Alain Mebsout[1], and Sylvain Conchon[2]

Functori(1) & Nomadic Labs(2), Paris, France

Résumé

Cet article présente Mitten, un outil pour décrire finement et jouer des scénarios sur une implémentation de Tenderbake — le prochain protocole de consensus *à la pBFT* de la blockchain Tezos. Mitten est paramétrable pour filtrer et examiner les messages selon des scénarios particuliers écrits dans un DSL construit sur OCaml. Grâce à Mitten, nous avons pu écrire et simuler des scénarios subtils pour reproduire des comportements difficilement atteignables en temps normal. Nous avons également pu simuler des situations permettant d'exhiber des bugs dans l'implémentation en cours et de tester des correctifs proposés.

1 Introduction

La conception d'algorithmes de consensus est une tâche très ardue et complexe. Il convient de prendre en considération de nombreux paramètres pour en assurer la correction et la vivacité, comme le ratio d'acteurs byzantins tolérés, la désynchronisation des participants, les délais et la perte de messages sur le réseau, *etc.* Qu'ils permettent une finalité probabiliste ou déterministe, les algorithmes de consensus sont au cœur de la technologie des chaînes de blocs (blockchains). Par exemple, les algorithmes de consensus de type pBFT, qui permettent d'avoir une finalité déterministe et immédiate, sont au centre des blockchains basées sur le protocole Tendermint [5].

Pour tester Tenderbake [1], le prochain protocole de Tezos [4], nous avons développé Mitten ; un outil pour décrire finement et jouer des scénarios sur une implémentation dudit protocole, car recréer les conditions et les progressions qui amènent à une configuration voulue est particulièrement difficile, ce qui rend ces protocoles de consensus ardus à tester. Mitten est à son cœur un proxy *intercepteur* pour un réseau pair-à-pair, qui peut être paramétré pour filtrer et examiner les messages selon un scénario particulier. Les scénarios sont écrits dans un DSL construit sur OCaml, ce qui apporte une grande souplesse à l'outil. Grâce à Mitten, nous avons pu écrire et simuler des scénarios subtils pour reproduire des comportements difficilement atteignables en temps normal. Nous avons également pu simuler des situations permettant d'exhiber des bugs dans l'implémentation en cours et de tester des correctifs ou améliorations proposés.

2 Description et architecture

Ce travail fait suite à l'implémentation en TLA+ [3] du protocole Tenderbake. Il est possible de vérifier certaines propriétés et de mettre en évidence certains comportements du protocole en utilisant le model checker TLC sur le modèle TLA+. Mitten permet dans un second temps de s'assurer que ces comportements sont réalisables par l'implémentation. Mitten est conçu comme un proxy intercepteur entre différents nœuds d'un réseau pair-à-pair blockchain. Il se connecte au travers de multiples *sockets TCP* à plusieurs nœuds Tezos, configurés pour ne communiquer qu'avec le proxy. Mitten est ainsi capable de voir passer *tous* les messages échangés sur le réseau, et d'agir en conséquence (faire suivre le message, jeter le message, retarder le message, et même le modifier). Cette logique d'action est implémentée par la machine interne de Mitten (voir Figure 1), qui peut, elle-même, être paramétrée par un scénario.

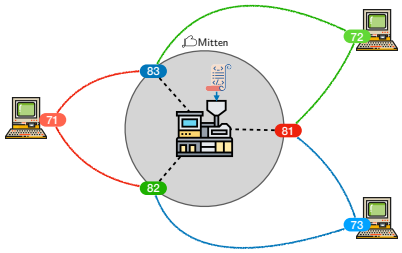

FIGURE 1 – Architecture du proxy Mitten

$$
\begin{aligned}
\langle const \rangle &::= \texttt{N ".*"} \mid [0-9]^{+} \mid \texttt{vh[a-zA-Z]}^{52} \\
\langle v_i \rangle &::= \texttt{\_} \mid \texttt{".*"} \mid \langle const \rangle \\
\langle istep \rangle &::= \texttt{propose } \langle v_b \rangle\ \langle v_l \rangle\ \langle v_r \rangle\ \langle v_p \rangle\ \langle v_d \rangle \\
&\quad \mid \texttt{preendorse } \langle v_b \rangle\ \langle v_l \rangle\ \langle v_r \rangle\ \langle v_p \rangle\ \langle v_d \rangle \\
&\quad \mid \texttt{endorse } \langle v_b \rangle\ \langle v_l \rangle\ \langle v_r \rangle\ \langle v_p \rangle\ \langle v_d \rangle \\
&\quad \mid \langle step \rangle\ \texttt{[\&\& | || | ->?]}\ \langle step \rangle \\
&\quad \mid \texttt{\~!}\langle step \rangle \\
&\quad \mid \texttt{seq[}\langle step \rangle^{\star}_{;}\texttt{]} \\
\langle step \rangle &::= \{pre\}\langle istep \rangle\{post\}
\end{aligned}
$$

FIGURE 2 – Langage de scénarios

Les scénarios pour MITTEN sont écrits dans un langage qui décrit quels messages doivent être transmis (voir Figure 2). Une étape élémentaire du scénario représenté par `propose b1 (level 3) (round 0) __ b2` laisse passer un message de (*i.e.* signé par le validateur du nœud) `b1`, qui est un "propose" au niveau 3 et round 0, avec n'importe quel payload hash au nœud `b2`. Le langage permet également de composer ces étapes élémentaires, comme des séquences, conjonctions, disjonctions, mais aussi des boucles d'attente–transmissions. Ce langage est plongé dans OCaml, et permet également de décorer les étapes avec des assertions (sous la forme de *pre* ou *post* actions/vérifications). Bénéficier d'un langage de programmation comme OCaml pour l'écriture d'assertions apporte une grande souplesse.

3 Exemple

Cet extrait de scénario avec deux nœuds `a` et `b` vérifie qu'un validateur qui a observé un *quorum de préendorsements* (PQ) (mais pas de quorum d'endorsements) propose à nouveau au round suivant avec le *même payload hash*. On voit que `a` doit proposer au niveau 3, round 0 avec un payload hash `p1`, que les validateurs s'échangent les préendorsements normalement, que `b` *endorse* (donc il a vu un PQ sur `p1`) et qu'il doit proposer au round 1 avec le même `p1`. MITTEN fait du filtrage par motif (ie. étapes du scénario) sur les messages. La variable `p1` est donc liée à une valeur concrète après le *propose* au round 0.

```
...
let scenario = seq [
  [ preendorse __ (level 3) __ __ [__];
    endorse __ (level 3) __ __ [__];
  ] <>?
  propose a (level 3) (round 0) p1 [a;b]
    ~post:[Exec (Helpers.inject_dummy n_b)];
  preendorse a (level 3) (round 0) __ [a;b] ;
  preendorse b (level 3) (round 0) __ [a;b] ;
  endorse b (level 3) (round 0) __ [a;b];
  propose b (level 3) (round 1) p1 [a;b];
]
let () = run_scenario { code; timeout = None;
    nodes; parameters; constraints = [] }
```

4 Conclusion et travaux futurs

Bien que MITTEN ait été initialement conçu pour Tezos, son architecture modulaire le rend adaptable à d'autres usages. En effet, la partie proxy paramétrable peut être utilisée sur n'importe quel réseau (modulo quelques modifications), ceci afin d'observer, tester et contrôler les échanges. Par exemple, des acteurs byzantins pourraient être simulés par MITTEN, comme dans l'approche de Twins [2]. Il serait aussi pertinent d'observer les échanges de messages en temps réel et de vérifier qu'ils représentent bien des traces possibles de l'automate spécifiant le protocole. Un second objectif serait le développement d'un DSL de plus haut niveau pour décrire les scénarios. Enfin il serait intéressant de rejouer les traces fournies par un model checker.

Références

[1] Lăcrămioara Astefănoaei, Pierre Chambart, Antonella Del Pozzo, Thibault Rieutord, Sara Tucci Piergiovanni, and Eugen Zalinescu. Tenderbake - A Solution to Dynamic Repeated Consensus for Blockchains. In *Fourth International Symposium on Foundations and Applications of Blockchain*, 2021.

[2] Shehar Bano, Alberto Sonnino, Andrey Chursin, Dmitri Perelman, and Dahlia Malkhi. Twins : White-Glove Approach for BFT Testing. *arXiv preprint arXiv :2004.10617*, 2020.

[3] Sylvain Conchon, Alexandrina Korneva, Çagdas Bozman, Mohamed Iguernlala, and Alain Mebsout. Formally Documenting Tenderbake. In *3rd Workshop on Formal Methods for Blockchains (FMBC 2021)*, 2021.

[4] LM Goodman. Tezos—a self-amending crypto-ledger White paper. *URL : https ://www.tezos.com/static/papers/white paper.pdf*, 2014.

[5] Jae Kwon and Ethan Buchman. Cosmos whitepaper, 2019.