Distributed Consensus

Simulator for Sleepy Consensus Protocol

SJTU 2017 Cornell Summer Workshop

Instructor

Elaine Shi

Cornell University

Our Group:

- **Framework Team Members**

Junxiang Huang	841450297@qq.com
Yifei Pu	pkq2006@gmail.com

- **Honest Node Team Members**

Tiancheng Xie	wjxtcsgx@hotmail.com
Jiaheng Zhang	ZHANGJIAHENG@sjtu.edu.cn
Xiaotian You	youxiaotian@hotmail.com
Shuyang Tang	tangshuyang25@163.com
Chengyao Li	cyli2014@sjtu.edu.cn

- **Adversary Members**

Qingrong Chen	chenqingrong@sjtu.edu.cn
Ruisheng Cao	211314@sjtu.edu.cn
Shiquan Zhang	zsq007@sjtu.edu.cn
Haochen Huang	hhc98598@189.cn

- **Integrators**

Feiyang Qiu	st.yeah@gmail.com
Lingkun Kong	klk316980786@sjtu.edu.cn
Lanqing Liu	sunnysunny@sjtu.edu.cn
Jialu Li	790359064@qq.com

- **Where to find our project?**

https://github.com/initc3/sleepysim	SleepySim

Table of Contents

1 Introduction

Consensus protocols are at the core of distributed computing and also provide a foundational building protocol for multi-party cryptographic protocols. In the paper about *sleepy consensus* protocol [1], Pass and Shi propose a consensus protocol for realizing a linearly ordered log abstraction – often referred to as state machine replication or linearizability in the distributed systems literature. They name it as sleepy consensus protocol, which respects two important resiliency properties, i.e., consistency and liveness. And in sleepy consensus model, players can be either online (alert) or offline (asleep), and their online status may change at any point during the protocol.

Algorithm 1 presents how sleepy consensus protocol works. The protocol takes a parameter p as input, where p denotes the probability each node is elected leader in a single time step. All nodes that just spawned will invoke the init entry point. During initialization, a node generates a signature key pair and registers the public key with the public-key infrastructure F_{CA}.

Algorithm 1 Sleepy Consensus Protocol

```
If On Initialisation:
 1: Let $(pk, sk) := \sum.\text{gen}()$
 2: Register $pk$ with $F_{CA}$
 3: Let $chain := genesis$
If On Received chain':
 4: Assert $|chain'| > |chain|$ and $chain'$ is valid w.r.t. eligible and the current time $t$
 5: $chain := chain'$ and gossip $chain$
Every Time Step:
 6: Receive input transactions(txs)
 7: Let $t$ be the current time
 8: if $\text{eligible}^t(P)$ where $P$ is the current node's party identifier then
 9:     Let $\sigma := \sum.\text{sign}(sk, chain[1].h, \text{txs}, t), h' := d(chain[1].h, \text{txs}, t, P, \sigma)$
10:     Let $B := (chain[1].h, \text{txs}, t, P, \sigma, h')$, let $chain := chain||B$ and gossip $chain$
11: end if
12: Output $extract(chain)$ to $Z$ where $extract()$ is the function outputs an ordered list
    containing the txs extracted from each block in $chain$
Subroutine eligible^t(P):
13: if $H(P, t) < D_p$ and $P$ is a valid party of this protocol. then
14:     return 1
15: else
16:     return 0
17: end if
```

Now, the sleepy protocol proceeds very much like a proof-of-work blockchain, except that instead of solving computational puzzles, in this protocol a node can extend the chain at time t iff it is elected leader at time t. To extend the chain with a block, a leader of time t simply signs a tuple containing the previous blocks hash, the nodes own party identifier, the current time t, as well as a set of transactions to be confirmed. Leader election can be achieved through a public hash function H that is modeled as a random oracle. The difficulty parameter

D_p is defined such that the hash outcome is less than D_p with probability p. For simplicity, here we describe the scheme with a random oracle H – however as we explain in this section, H can be removed and replaced with a pseurdorandom function and a common reference string.

In this document, we build a simulator for monitoring the real-world performance of sleepy consensus protocol by constructing a framework which implements Algorithm 1, as well as imitating behaviors of honest players and corrupted/adversarial players in the meanwhile. After analyzing the simulating results, **we know ... add text here**

This document is organized as follows. In Section 2, we introduce the framework of simulator. In Section 3, we present how honest players work while simulating. And in Section 4, we imitate the adversarial players' behavior and attack the sleepy consensus protocol by several algorithms. We give the analysis of simulating results in Section 5. Finally, we draw conclusions in Section 6.

2 The Framework of Simulator

In this section, we will illustrate the construction of the framework of our simulator, which includes **add text here**

2.1 Controller

The LLNCS class is an extension of the standard LaTeX "article" document class. Therefore you may use all "article" commands for the body of your contribution to prepare your manuscript. LLNCS class is invoked by replacing "article" by "llncs" in the first line of your document:

```
\documentclass{llncs}
%
\begin{document}
  <Your contribution>
\end{document}
```

3 The Imitation of Honest Players

3.1 Algorithm for Honest Players

In our program, the honest nodes use the algorithm of sleepy consensus. To be specific, There are some steps for the algorithm.

– First, the nodes will elect a leader using the hash function of identity and current time. If
$$H(identity, currenttime) < D$$
, where D denotes difficulty, then $Node[identity]$ will be elected to be a leader.

– Second, the leader can sign the block using the hash value of previous block, transactions and time, then broadcasts.

$$Block = sign(sk, block, Trans, time)$$

– When one honest node receive a new chain, if the time in block is strictly increasing and the time in the blocks is not in the future, it will update its chain with the new chain.

3.2 The process to simulate sleepy consensus

There are several steps to simulate the algorithm of sleepy consensus in our program.

– First of all, in every round, the controller will let every node to run.

– Second, the honest node will ask network controller for messages.

– Third, the network controller will send related message to every node.

– Then every honest node will ask the controller whether it has been elected a leader.

– Next, if one honest node is not elected to be a leader, it will do nothing. However, if it is a leader, it will sign one new block using its secret key with the hash value of previous block, transactions and time stamp. Then it will broadcast it.

– Finally, whatever one honest node has done, it will give some feedback to controller and network controller.

4 The Imitation of Adversarial Players

The previous section introduces the implementation of the honest players' behavior under the sleepy consensus protocol in simulator. This section, in contrast, will present the imitatation of the adversarial players' behavior, which aims to hinder the normal functioning of sleepy consensus protocol under current framework structure. And we design four different attacking algorithms to try to break the consensus between different nodes, i.e., players in the distributed system. What should be noticed is that the adversaries cannot betray the rules established by the framework, while they can only control the network message transportation, i.e., intercepting and delaying the message from honest players. Also, an adversary can manipulate several corrupted nodes in the system, by which adversary impose damage to the system under sleepy consensus protocol.

To simulate the attacks, we assume there is an an adversary lurking within the framework, who owns competence to intercept all messages coming from honest nodes and decide which to delay in the transportation. Also, the adversary is able to access useful information from these message to fork blocks right behind the private chain he captures. Based on above setting, we design four attacking methods for adversary to smash the consensus holded by system, which are illustrated in following subsections.

4.1 Naïve Adversary Attack

4.2 Selfish Adversary Attack

4.3 Stubborn Adversary Attack

4.4 Selfish Eclipse Attack

5 The Analysis of Simulating Results

Here is the analysis of Simulating Results.

6 Conclusion

In conclusion....

References

1. Pass, Rafael, and Elaine Shi. *The sleepy model of consensus.* Cryptology ePrint Archive, Report 2016/918, 2016. http://eprint. iacr. org/2016/918, 2016.