The 33rd Annual Conference of the Japanese Society for Artificial Intelligence, 2019

Cooperation Model for Improving Scalability of the Multi-Blockchains System

Liu Keyang Ohsawa Yukio Teruaki Hayashi

Department of System Innovation, Graduate School of Engineering, The University of Tokyo

Scalability is an open question of the blockchain. Ongoing solutions, like Sharding and Side-chain, try to solve it within an independent blockchain system. We propose a cooperation model by constructing a system of multiple blockchains. In this model, secure cross chain operations can help to handle more requests. The gossip channel can help to refresh the states of other blockchains. Through manage interactions among blockchain systems, this model can limit their misbehaviors and improve scalability.

1. INTRODUCTION

Blockchain, a solution to decentralized systems, can solve problems in fields like finance[Eyal 2017], supply chain[Abeyratne 2016], crowdsourcing [Li 2018]. Currently, blockchain can provide two functions. First, a blockchain can work as a securely distributed ledger[Ren 2018]. Second, a blockchain can provide a reliable distributed calculating platform. By using smart contracts[Underwood 2016], all participants can execute functions correctly and give the same output.

Generally, a decentralized system is more robust and trusted than a centralized system. However, the scalability is its weakness. Scalability problem is the long latencies or superfluous messages caused by growing participants. Usually, it is a result of the consensus algorithm[Karame 2016]. Many solutions try to solve this problem within a blockchain system. Sidechain shifts some assets into a sidechain to realize faster responses[Back 2014]. Sharding technology tries to split participants into several shardings for parallel processing[Luu 2016]. All these works sacrifice security or consistency for the efficient responses.

This work tries to solve the scalability problem through cooperative problem-solving. In this model, each blockchain system is an independent agent. The contributions of this work are 1. A protocol for delivery versus payment(DVP) problem. 2. The framework of Blockchain's cooperation model.

2. RELATED WORKS

Sharding is an exciting idea that split blockchain into several shardings so they can handle requests simultaneously. This idea shares some similarity with multiple blockchains system. Elastico[Luu 2016] and Rapid chain[Zamani 2018] are some great implementations of this solution. In these systems, the randomness of each sharding limits the possibility of collusion and planned attacks. However, they give up security or consistency to some degree. To solve this problem, we propose a framework to weaken secure assumptions of each blockchain. By considering possible attack happens, this work focus on limiting the effect of attacks. Hence, our model allows more sacrificing of security on individual blockchain while providing better services.

Chen et al[Chen 2017] and Kan et al[Kan 2018] had finished some works about the communication between different blockchains. They simulate Internet stack and TCP protocol to create the Inter blockchain communication protocol. Although these methods are functional, they are also fragile to malicious attacks. Besides, they ignored the achievements of the consensus algorithm which is very useful in DVP problem. This work will consider the case that some blockchains are controlled or created by attackers. We can prove that these attacks cannot affect other blockchains.

3. PROBLEM MODEL

This part will clarify assumptions and notations of this model. First, $N = 1, 2, ...n$ represents the set of agents. Each agent i is a distributed network that maintains one blockchain $B_i[]$ with all terminated blocks list linearly. The participants of each agent run the consensus algorithm to maintain the blockchain and provide their services to users. Terminated block means at least $f_i > 0.5$ participants have confirmed and stored the block. f_i is the parameter of each agent's consensus algorithm.

Second, all block contents two parts: header and body. A header contains at least the hash of the previous Block, metadata of the body, and signatures of the creator. For convenience, all blockchains' contents, like transactions or Key-Value pair, is unified under an abstracted class – log. Each agent should support two operations: $verify$ and $check_i$. $verify(log, h)$ will return the validity of one log before the h^{th} block $B_i[h]$ according to the rule of the blockchain. Taking a Bitcoin's transaction as an example, input should be a subset of unspent transaction output (UTXO), and the sum of inputs should be larger than the sum of outputs. $check_i(log)$ returns the position of one log in B_i. It will return -1 when it does not exist. Hence, the

Contact: Liu Keyang, University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656 Japan, Department of Systems Innovation, School of Engineering,The University of Tokyo, Bldg.No.8. 507 , 070-4336-1780, stephenkobylky@gmail.com

following property held:

- a.(Validity) $\forall h > 0$ and $log \in B_i[h]$, $verify_i(log, T) = True$ for $T < h$ and $verify_i(log, T) = False$ for $T > h$.
- b.(Agreement) If $check_i(log) == True$, log can be accessed from at least f_i part of participants in agent i.

Third, we assume adding one legal log into a blockchain consume resources for all agents. Under this condition, cross chain operation becomes a DVP problem. Off-line cost's DVP problem is the job of exchanges. This work focus on the online DVP problem between two agents. Cooperation model will guarantee the payment's validity stick to the delivery of goods.

Last, we allow an agent itself can work improperly or attack some other agents. Since an agent can work independently, agents need some mechanism to control the effect of attacks. One naive way is creating a higher layer blockchain among different agents so it can be byzantine fault tolerate. This work uses another lightweight method. We create an externality of each cross-chain operation without affecting other agents. However, these externalities can prove the existence of misbehavior and punish the agent by detaching it from the network.

4. COOPERATION MODEL

This section includes the detail of the cooperation model. To begin with, we defined two extended functions for each participant of agents in our model.

4.1 Extended operation

Define a condition log $clog1 = log1||log2||j||h1||h2$ represents $log1$ is a cross chain operation related to $log2$ in agent j. The expiration height for a condition log is $h1$ and $h2$ in agent i and j. All participants of agents maintain a waiting list(WL) for cross chain log and condition log. WL supports a function $condcheck()$. When $clog1$ and $log1$ are stored together in WL, $condcheck(log1) = clog$ and $condcheck(clog1) = log1$. In other cases, it returns the existence of input log in WL.

Next, we need to define the extended function $checkEX_i(log1)$. Let $checkEX_i(log1)$=-1 if $condcheck(log1) = True$. If $condcheck(log1) = clog1$, $checkEX_i(log1) = check_i(clog1)$. In other cases,$checkEX_i(log1) = check_i(log1)$. $checkEX_i(clog1) = check_i(clog1)$

Then, we define function $verifyEx(log, h)$ in algorithm 1.

4.2 Workflow

By using previous notations, the operations to WL are following:

- When the new terminated block contains a condition log $clog1$, all participants add $clog1$ and $log1$ into their WL.

Algorithm 1 verifyEx

Input: log1 or clog1, h
Output: True or False

```
if Input is normal log then
  if condcheck(log1)! = clog1 then
    return verify(log1, h)
  else
    return checkEX_i(clog1) ≥ 0 & checkEX_i(log_1) < 0 & checkEX_j(clog2) ≥ 0 & verify(log1, h)
  end if
else if condcheck(clog)! = False then
  return False
else
  return h < h_1 & verify(log1, h)
end if
```

- Expire: When $B_i[h1]$ is terminated $clog1$ is removed from WL. When$checkEX_j(log2) \geq 0$ and $checkEX_i(log1) \geq 0$ $log1$ and $clog1$ is removed.

the workflow of cross chain operations are following:

- 1. Register: A user submits $clog1$ to one participant. Participants check $verifyEX_i(clog1, h_t)$ where h_t is the current height of blockchain B_i. If it returns True, commit $clog1$ to next block.
- 2. Condition-commit: If the newest terminated Blocks contain $clog1$, all participants add $clog1$ and $log1$ to their WL.
- 3. Pre-commit: User submits $log1$ to one participant.The participant checks $verifyEX_i(log1, h)$. If it is true, commit $log1$ to next block.
- 4. Commit: When the height of blockchain reaches $h1$, participant check $checkEX_i(log1)$ and $checkEX_j(log2)$ to determine whether to expire $clog1$.

The workflow of a success cross chain operation looks like Figure 1.

Till now, we have clarified main steps of a cross chain operation. The final step is to broadcast the latest view, like the hash of last terminated block header, of both agents. One agent can use a gossip channel to notify other agents of updating. This gossip is not necessary to be received or confirmed. However, an agent can reveal a fork by identifying an unmatched view of one agent.

4.3 Communication rule

In the cooperation model, each blockchain is an agent to act. Hence, verifying the status of one agent demands sufficient supports from its participants. Due to the property of agreement, secure connection with one agent i anchors to the parameter f_i. For a given possibility p , the required confirmations m_i should satisfied $(1 - f_i)^{m_i} < p$. Hence, $m_i \geq \frac{p}{\ln(1-f_i)}$.

A secure communication requires enough participants of agent i asks for m_j confirmations from agent j independently. The communication cost is $O(m_i * m_j)$ per time.

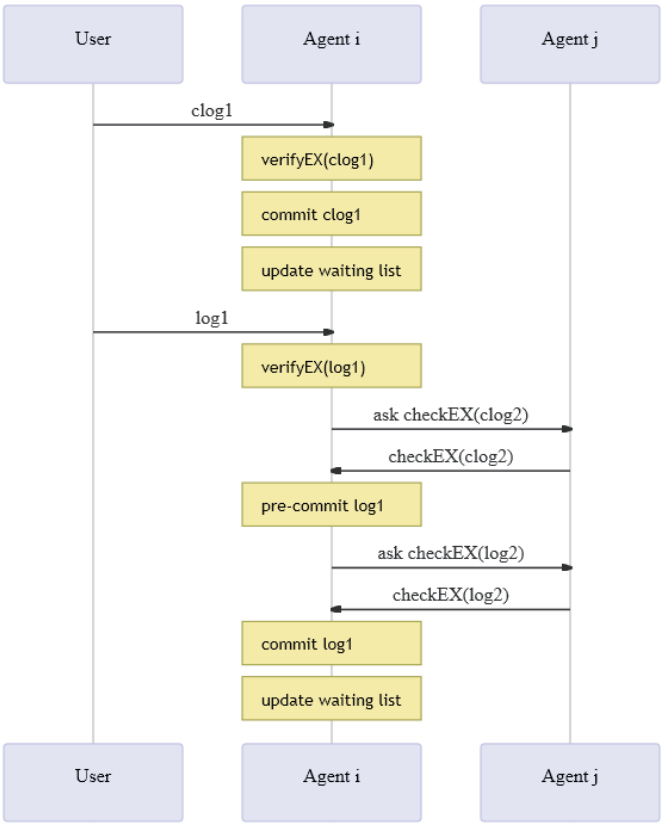

Figure 1: The work flow of one cross chain operation.

An efficient way is to select some proxies to do this communicate. Once selected proxies confirmed $checkEX_j()$, they can spread the gossip information among agent i. Under this method, the cost is $O(m_i)$. For preventing collusion, participants should randomly select proxies.

5. ANALYSIS AND EXPERIMENT

5.1 Function analysis

To view other possible situations of this protocol, we can consider the states of WL. WL can allow three statues:1. Null,2. $clog1$ and $log1$, 3. $log1$. Table 1 shows the result of VerifyEX and CheckEX, where the condition is the 3rd line in Algorithm 1. When a user submits $clog1$, only case 1 and case 3 can continue. Case 3 means new condition log is a supplement to an expired one which jumps to the final stage. When a user submits $log1$ at case 3, participants can detect that it is a rejected log. As a result, on one can admit a existed cross chain operation $log1$ without satisfying a condition $clog1$. This is the cost of agreement in a blockchain.

Assumes log' is conflicted with $log1$, which indicates the terminated blockchain can only contain one of them. The condition for committing log', $checkEX_i(log1)$ should be less than 0. Since termination requires admissions of at least half of all participants, they can exclude the possibility of conflicts within the blockchain. Once a terminated block contains log', it is impossible to activate $log1$ in case 3 any more. The reason is $clog1$ is also conflicted with log'.

Func \ WL	Null	Clog&log	log
VerifyEX(clog)	Verify(Log)	False	Verify(Log)
CheckEX(clog)	check(clog)	check(clog)	check(clog)
VerifyEX(log)	Verify(Log)	condition	Verify(Log)
CheckEX(log)	check(log)	check(clog)	-1

Table 1: Result of VerifyEX(ignore h) and CheckEX

Hence, by using the property of Blockchain, this protocol can solve the DVP problem of cross chain operations.

5.2 Security analysis

This part provides some brief proofs of security. Generally, there are two types of attacks: agent's misbehaviors and communication attacks. During the protocol, the only information needed about other agent is function $checkEX$ which affect by WL and terminated blocks. Hence, Adjusting can detect the counterfeit WL and B_i. Once attacker Eve controls the agent j, j can reply to other agents arbitrary and support any cross-chain operations. After completing a cross chain operation, Eve can create a fork to repeal the existed log. In this case, agents that received the previous view of j will anchor to the elder branch and reject new branch inherit the identity of j. The network will wait for j recovering the former branch and continue its services. Here, the gossip channel creates an externality of an agent so that it cannot change its termination within the network. The higher rate to verify gossip, the higher termination the model can propose.

Besides, there are some network attacks like Sybil attacks and DDOS attack among agents. Sybil attack means the attacker create several agents in the model to arrange attacks. However, these bot agents need to spend enough resources to convince users of other agents for one round attack. Hence, this attack is not profitable if users can manage their risk. Another way is isolating one agent like Man in the Middle(MITM) attack to block gossips. This attack is very costly when the target is a distributed network. A fixed and reliable channels can also resolve this attack. In a word, the resilience against manufactured identities depends on the value of each agent. Lastly, DDOS attacks also worth considering. Attackers can attack waiting list by creating many useless conditional logs. The solution can be charging an additional fee for registering cross chain operation. Indeed, cross chain operations require extra payment for stronger termination and complex procedures. The most significant problem relates to the gossip channel where redundancy informs can block useful gossip. Hence, the gossip channel needs some rule for filtering. Agents can require a signature for each message, limit frequency of source agents, create some periodical routes.

5.3 Experiment

This work intended to improve the scalability of one agent through cross chain operation. The experiment evaluated the average latency and gossip burden for different rates of cross chain operations in the worst case. Latency is evaluated by the number of blocks for solving same amount

of requests. Gossip represents the number of message sent in gossip channel when discard rate is 0.5. The result(Figure 2) shows a linear growth of latency with inter log rate and stable average gossip message related to the number of agents. The increasing of latency relates to the size of condition logs. In the experiment, we assume condition log spending double space compare to other logs.

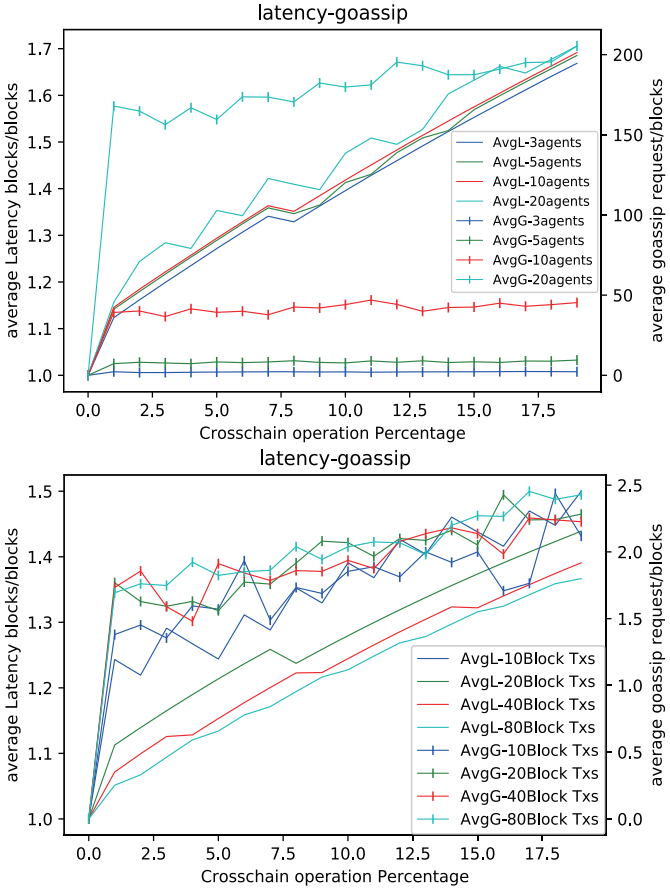

Figure 2: Average latency and gossip according to cross operation rate when transactions terminate immediately. The first picture shows the number of gossips is square to the system's scale. The second picture shows the system do not accumulate gossips or latency as time goes by.

6. CONCLUSION

This work proposes a cooperation model for improving scalability. Under the cooperation model, cross chain operation can create externality of each agent and spread it through gossip channel. The termination of one agent can partly rely on other agents and agent can pay more attention to achieve agreements. On the other hand, cross chain operation extends the ability of one agent and allow a more flexible exchange between different digital assets. We can still optimize this work in many ways. Condition log and gossip message can be compressed to reduce latency and burden of gossip channel. A gossip checking protocol can detect forks faster. The future works will focus on the consensus design under the cooperation model and optimization of gossip channel and related protocols.

7. ACKNOWLEDGMENT

This work was funded by JSPS KAKENHI, JP16H01836, JP16K12428, and industrial collaborators.

References

[Eyal 2017] Eyal I. Blockchain technology: Transforming libertarian cryptocurrency dreams to finance and banking realities[J]. Computer, 2017, 50(9): 38-49.

[Abeyratne 2016] Abeyratne S A, Monfared R P. Blockchain ready manufacturing supply chain using distributed ledger[J]. 2016.

[Li 2018] Li M, Weng J, Yang A, et al. Crowdbc: A blockchain-based decentralized framework for crowdsourcing[J]. IEEE Transactions on Parallel and Distributed Systems, 2018.

[Karame 2016] Karame G. On the security and scalability of bitcoin's blockchain[C]//Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2016: 1861-1862.

[Back 2014] Back A, Corallo M, Dashjr L, et al. Enabling blockchain innovations with pegged sidechains[J]. URL: http://www. opensciencereview. com/papers/123/enablingblockchain-innovations-with-pegged-sidechains, 2014.

[Luu 2016] Luu L, Narayanan V, Zheng C, et al. A secure sharding protocol for open blockchains[C]//Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2016: 17-30

[Ren 2018] Ren Z, Cong K, Aerts T, et al. A scale-out blockchain for value transfer with spontaneous sharding[C]//2018 Crypto Valley Conference on Blockchain Technology (CVCBT). IEEE, 2018: 1-10.

[Chen 2017] CHEN Z, Zhuo Y U, DUAN Z, et al. Inter-Blockchain Communication[J]. DEStech Transactions on Computer Science and Engineering, 2017 (cst).

[Kan 2018] Kan L, Wei Y, Muhammad A H, et al. A Multiple Blockchains Architecture on Inter-Blockchain Communication[C]//2018 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C). IEEE, 2018: 139-145.

[Zamani 2018] Zamani M, Movahedi M, Raykova M. RapidChain: scaling blockchain via full sharding[C]//Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2018: 931-948.

[Underwood 2016] Underwood S. Blockchain beyond bitcoin[J]. Communications of the ACM, 2016, 59(11): 15-17.