Simulating Resource Consumption in Three Blockchain Consensus Algorithms

John M. Medellin*, Ph.D., C.P.A. and Mitchell A. Thornton, Ph.D., P.E.
Research Associate Professor and C.E. Green Chair in Engineering
Darwin Deason Institute for Cyber Security, Southern Methodist University
Dallas, Texas 75275, USA
johnmedellin@verizon.net, mitch@lyle.smu.edu

Abstract— ***The Blockchain design pattern has become a very useful tool since it was first made famous by the Nakamoto paper in 2007/2008. This technology has many uses in private trust situations that go beyond support of crypto-currency exchanges.***

A key part of Blockchain is the ability for participants to come together in consensus and approve a particular block to be added to the chain. Our focus in this document is to present the results of resource consumption for the Byzantine Generals Problem, the RAFT Algorithm (proposed by researchers at Stanford University) and our own variation of RAFT; D-RAFT.

This paper executes simulations on the above algorithms and scales them to determine the resource consumption characteristics of each. In addition, we also scale the specific trust-election algorithm of RAFT and our own D-RAFT by varying the length of time before another election has to occur. Our results are presented and discussed in relation to each test case.

We believe the contribution of this work is to demonstrate the resource consumption of each under very controlled situations and to provide a potential enhancement to RAFT which may reduce even further the consumption of resources in that algorithm.

Keywords— Blockchain Scalability, Consensus Algorithms, Byzantine Generals Problem, RAFT, D-RAFT.

I. INTRODUCTION

Scarcely a decade has passed since the famous Nakamoto article [10] that gave exposure to Blockchain as a commercial design pattern; the high potential for disintermediating financial institutions from monetary transactions. This particular paper has inspired incalculable number of business models has created value.

Blockchain patterns rely heavily on two key factors, a) a Consensus algorithm which dictates when a particular block of data is officially committed to posterity (usually because most agreed it does [9]) and b) Encryption of the data block by taking parts of the previous Blockchain and executing a particular hashing algorithm usually based on discrete mathematics and number theory [5]. With this approach, we mathematically guarantee that the transaction remains without tamper via cumulative cryptography.

A first decision in implementation is whether the focus is to provide a vehicle for anyone who is legitimate to join and transact without disclosing specific identity credentials (a public blockchain). A private blockchain in contrast provides the ability for a known group of individuals to store transactions in the ledger and be assured of their sequence and tamper-proof capabilities at a higher degree of personal trust [13]. In a bitcoin public blockchain example, a person can purchase an entitlement of crypto-currency (a "bitcoin" or more) to join that particular network and begin gaining or transferring value with actors they may not even know or trust as long as the transaction is valid under certain rules. The way of achieving consensus in that case is through a "proof of work". In that case, the block data (value exchange transaction) is deemed complete and committed to posterity through encryption when over 51% of the computational resource in the network has used it to encrypt their downstream transactions [10]. This consensus method guarantees that most of the network has agreed and used the transaction in its own encryption and also has accounted for the computational resource (thus negating the potential cyber-attack by a rogue actor; in theory, they would not have the computational ability to invalidate it since they have less resource at their disposal to attack).

As mentioned above, the objectives of a private blockchain are to guarantee the identity and encryption of a transaction (or event) in a closed network of actors who are known to each other. The encryption algorithm can be fairly similar to the public blockchain (in our experimental code we have modeled a SHA-256 workload according to the model in Stallings [12]). The consensus algorithms can vary and most of these tend to be certain variations of the Byzantine Generals Problem described by Lamport, Shostak and Pease [7]. One particular variation of this algorithm is "RAFT" proposed by Ongaro and Ousterhout [11]. This concept seems to deliver most of the benefits in the Byzantine Generals Problem but under greater understandability and significant reduction in computational/time resources. We will explore the details of those two models later in this document.

In our analysis, we first present the related work of the two algorithms mentioned above in greater detail and

provide one potential variation to RAFT (we call it "D-RAFT"). Next, we create a series of experiments which simulate the impact on resources (CPU and time-duration) of each algorithm by increasing the volume of blocks that are being processed. In a second set of experiments we vary the time period a trust-election is valid for between RAFT and D-RAFT in order to test the impact of that parameter.

This is an initial exploratory study into the nature of resource consumption of the above but under very controlled conditions. Further details on the limitations of this research are discussed towards the end of the document.

II. RELATED WORK

As mentioned above, a significant amount of literature and time has been spent on consensus algorithms for blockchain [9][13]. Most of them have been modeled as efficiency improvements to the Byzantine Generals Problem while preserving the basic assumptions of majority vote to commit a transaction to a final tamper-proof ledger. Next, we will review Byzantine and RAFT.

A. The Byzantine Generals Problem.

The Byzantine Generals Problem is a challenge that is explained as follows.

1. There are a number of Generals that are leading the Byzantine Empire's army in a siege of a particular city.
2. They must all attack together at a given time in order to conquer the city.
3. They communicate through messengers that delivery information through verbal channels only.
4. A minority of them are traitors that will try to undermine the others by providing false information.
5. At some point in time, all the loyal Generals will agree on a time and when the majority of them agree the time for attack is set.

There are many potential ways to model the above sequence and many scientists have done so in the past. Our particular model is explained later in the experiment design section of this document.

B. The RAFT Consensus Model.

RAFT is the creation of Ongaro and Ousterhout [11] of Stanford University and has been designed with most of the benefits afforded by the majority-vote concept in the Byzantine Generals Problem. The consensus model can be explained as follows.

1. There are a number of nodes in the network that can carry out computations and will participate in placing their blocks on the blockchain.
2. Time is partitioned into epochs (our term; meaning intervals of time) and each epoch's duration is chosen at random.
3. Before an epoch begins, each of the nodes votes for a node (presumably through a mathematically sound way of selecting who to vote for), the one node with majority of votes wins.
4. If there is a tie then another election begins immediately (please note that the original document has 5 nodes participating to minimize chances of a tied-vote).
5. The winner issues the epoch's duration and a secret that is used by all nodes in encryption of transactions during the epoch.
6. As an epoch ends, another trust-election begins and so the process continues.

Researchers in this area have built models around this algorithm and provided initial results of the efficiency over the Byzantine Generals Problem with the differences in resource usage being higher than a 50% reduction (the results can be different in between runs since there is a randomization aspect to the model).

C. Nakamoto Proof-of-Work.

As mentioned above, the Nakamoto Proof-of-Work model for consensus relies on knowledge of the network computational resources and deems a block as finalized and committed when 51% of these resources have used the block in their encryption (note that the encryption of each transaction contains elements from previous blocks in that computation). This algorithm can be explained as follows.

1. Participants purchase for value an entitlement that allows them to transact with others in the network.
2. When a transaction is needing to be appended to the other blocks, that transaction's node re-computes the cryptography of previous transactions that are not yet committed and derives an encryption of its own.
3. If the assumptions of the blockchain encryption are still valid (e.g. another node has not placed their transaction while the encryption computations were being executed) then the node reports the transaction to the rest of the nodes as a valid addition and reports the computation power it has spent doing work. This number accounts for a percentage of the total computation power (which is known by all).
4. More nodes add their encrypted transaction blocks and when a transaction block yields an accumulated load of 51% or more since the original transaction block was appended this will cause the original one to be deemed finalized and it will be committed to permanency.

The above sequence is what is deployed by the bitcoin crypto-currency. Other algorithms exist as well that rely on

other assumptions (such as Proof-of-Stake [13] which uses the amount of value that the particular node is entitled instead of the computation power, once the accumulated stake percentages exceed 51% it will get committed).

III. Experiment Design

We designed our experiments to simulate the execution of consensus and encryption of the algorithms referenced above. We also created a variation on RAFT ("D-RAFT") which can be explained as follows.

1. Before an epoch begins, the prior elected node executes a randomization algorithm to select one primary node and one backup node from the known nodes.
2. Another randomization algorithm designates the amount of time the new epoch's duration will be.
3. The out-going node notifies the primary and backup nodes of their status and broadcasts to the network the new epoch's duration.
4. The primary (or backup if down) node issues the secret for encryption to the network and it is used for the duration of that epoch.
5. When the epoch is over, the out-going node executes the first step again (it will designate lead nodes again, which is where the "D" in D-RAFT comes from, it is for "Designated" by the out-going node).

Next, we will describe the infrastructure setting for our experiments and the Model Code Architecture for each one of the three algorithms.

A. *Experiment Infrastructure*

The hardware and system software infrastructures for the experiments is described below.

Hardware Infrastructure

The hardware infrastructure used consisted of:

- Apple MacBook Pro
- 3.1 GHz Intel Core i7
- 8 GB 1867 MHz DDR3

System Software

- OS: Mac OSX Sierra v. 10.12.6
- GCC Compiler (bundled in OS)
- POSIX Threads

B. *Experiment Model Architecture*

Versions of each model were coded in the C language using two POSIX Multi-Threaded design paradigms [2]:

- Mutexes were used to isolate critical regions where votes, epoch duration and block appending operations were involved (interrupts generated under this design).
- Thread-specific data was employed for encryption of potential transactions (no interrupt generated).
- Time duration was considered in cycles rather than actual internal time clock for synchronization in all models. See [3] for discussion on clock synchronization issues.

We next describe the application architecture of each model using diagrams similar to those proposed by Booch in RUP [1] and Larman in UML [6]. In addition, we further advise that the code is available through Github [4].

Byzantine Generals Problem Architecture

The Byzantine pseudocode is as follows (please refer to the Activity diagram immediatelly after):

<begin>

<receive transaction and encrypt>

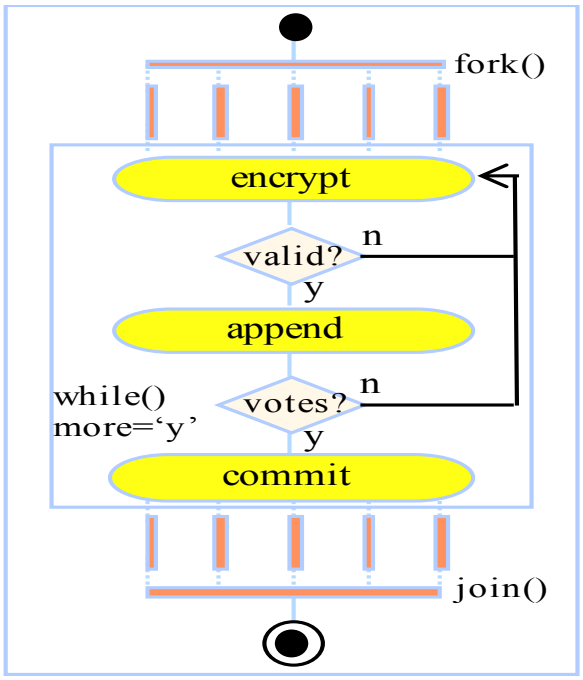
<if transaction ok for next, append & increment vote count>

<else recompute>

<if votes=majority, commit transaction>

<loop to begin>

Figure 1: Byzantine Activity Diagram



RAFT Architecture

The RAFT pseudocode is as follows (please refer to the Activity diagram immediatelly after):

<begin>

<vote>

<if majority; designate node, issue secret & epoch duration>

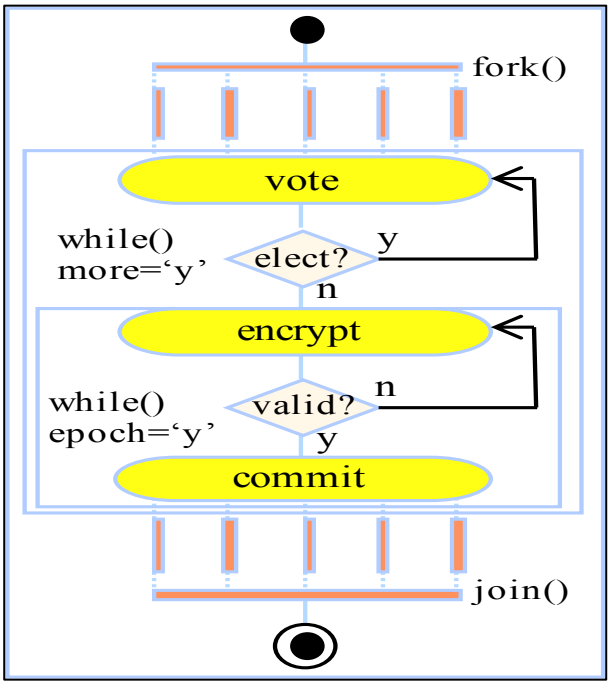
<encrypt>

<if transaction ok for next, append >

<else recompute>

<if epoch over, loop to begin>

Figure 2: RAFT Activity Diagram



D-RAFT Architecture

The D-RAFT pseudocode is as follows (please refer to the Activity diagram immediatelly after):

<begin>

<randomize, designate primary and backup nodes, notify>

<primary: issue secret & epoch duration>

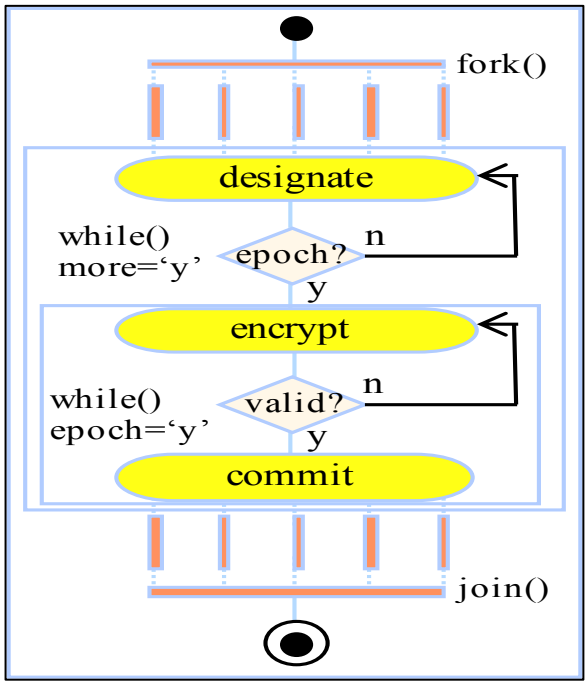
<encrypt>

<if transaction ok for next, append >

<else recompute>

<if epoch over, loop to begin>

Figure 3: D-RAFT Activity Diagram



IV. Experiment Results

Table 1 reports the CPU % and Mac OSX time used by each of the three algorithms described above for processing 100K, 200K and 300K blocks. Table 2 reports the CPU and Mac OSX time used by RAFT and D-RAFT for varying the epoch from 50, 500 and 5,000 cycles.

Table 1: Resource Consumption; Volume Scaled

Volume	CPU%	Time
100K Blocks		
Byzantine	48%	0:00.91
RAFT	12%	0:00.27
D-RAFT	1%	0:00.02
200K Blocks		
Byzantine	53%	0:02.23
RAFT	27%	0:00.53
D-RAFT	3%	0:00.08
300K Blocks		
Byzantine	79%	0:03.51
RAFT	30%	0:00.78
D-RAFT	4%	0:00.09

The above results report the efficiency differences between the three algorithms. When using the Byzantine as a base, there is a significant incremental efficiency in RAFT and again more efficiency in D-RAFT.

Figure 4: Resource Consumption; Epoch Variations

(All Intances at 200K Blocks to be Added)

Epoch	CPU%	Time
50 Cycles		
RAFT	30%	0:00.58
D-RAFT	3%	0:00.08
500 Cycles		
RAFT	27%	0:00.53
D-RAFT	3%	0:00.08
5000 Cycles		
RAFT	25%	0:00.50
D-RAFT	3%	0:00.08

The table above was produced from varying the election time period (cycles) until 200,000 blocks are added to the chain. The RAFT algorithm will become more efficient both in CPU and time of execution as the election becomes less frequent. This is normal since the additional load of the election is executed less frequently. A key strategy of protection is to vary parameters frequently and using hard to predict frequencies. Infrequent elections could constitute a vulnerability since it is easier for an attacker to predict values if more stability is provided in most systems under attack.

V. RESULTS DISCUSSION & FUTURE PLANS

This is an initial discussion on resource usage characteristics of the three algorithms presented. Additional work is needed on the D-RAFT algorithm in the quality attributes and potential attack vectors of this model.

A. Results Discussion

Our volume scaling experiment indicates that the RAFT and D-RAFT algorithms are significantly more efficient in CPU and execution time. This is perhaps because of the removal of the majority vote from each transaction and replacement with the trust election component. This feature is the creation of Ongaro and Ousterhout [11] and they have obtained similar results. The D-RAFT algorithm replaces the election algorithm with a designation algorithm that requires tasks of the prior leader (executed through discrete mathematics/probability analysis) but removes the dialogue with other members of the network. We believe this removal is what makes this algorithm more efficient on both components (CPU and execution time).

The epoch duration component experiment reflects the efficiencies/inefficiencies by holding elections less/more frequently. We again believe that the mathematics component in the D-RAFT designation rather than voting is what might make this algorithm more efficient.

As expressed in various sections of this document, this is an initial variation on the RAFT algorithm and it has not been vetted for potential attacks. For example, what happens if two nodes are down in the network? or what if an impostor executes a man-in-the-middle (MITM) attack? These attack vectors (or others) have not yet been vetted for quality protection characteristics at this time.

B. Future Plans

The D-RAFT algorithm will continue to be evolved through additional experimentation and field trials to determine the validity and resillience of attack. The authors invite additional commentary be sent to us at the email addresses mentioned above.

REFERENCES

[1] G. Booch; "Object-Oriented Analysis and Design with Applications" c. Addison Wesley Longman, Inc. 1994, Reading, Massachusets.

[2] D. Butenhof; "Programming with POSIX® Threads, The Addison-Wesley Professional Computing Series" c. Addison-Wesley 1997, Boston, Massachusets.

[3] N. Ferguson, B. Schneier, T. Kohno; "Cryptography Enginering, Design Principles and Practical Applications" c. Wiley Publishing, Inc 2010, Indianapolis, Indiana.

[4] www.github.com

[5] R. Johnsonbaugh; "Discrete Mathematics 8th edition" c. 2018 Pearson Education, Inc., New York, New York.

[6] C. Larman; "Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development (3rd Edition)" c. Pearson Education, Inc. 2005, Upper River, New Jersey.

[7] L. Lamport, R. Shostak, M. Pease; "The Byzantine Generals Problem" ACM Transactions on Programming Languages and Systems, Vol. 4, No. 3, July 1982, p. 382-401.

[8] X. Liang, T. Wu; "Exploration and Practice of Inter-bank Application Based on Blockchain" The 12th International Conference on Computer Science & Education (ICCSE 2017) p. 219-224.

[9] D. Mingxiao, M. Xiaofeng, Z. Zhe, W. Xiangwei, C. Qinjun; "A Review on Consensus Algorithm of Blockchain" 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC) p. 2567 - 2572

[10] S. Nakamoto; "Bitcoin: A Peer-to-Peer Electronic Cash System" www.bitcoin.org.

[11] D. Ongaro and J. Ousterhout; "In Search of an Understandable Consensus Algorithm" Proceedings ATC'14 USENIX Annual Technical Conference (2014), USENIX.

[12] W. Stallings; "Cryptography and Network Security, Principles and Practice 7th edition"c. Pearson Education Limited 2018, London, United Kingdom.

[13] X. Xu, I. Weber, M. Staples, L. Zhu et. al.; "A Taxonomy of Blockchain-Based Systems for Architecture Design" Proceedings 2017 IEEE International Conference on Software Architecture, p. 243-252.