Performance Modeling of PBFT Consensus Process for Permissioned Blockchain Network (Hyperledger Fabric)

Harish Sukhwani[1], José M. Martínez[1], Xiaolin Chang[2], Kishor S. Trivedi[1], and Andy Rindos[3]

[1]Department of Electrical & Computer Engineering, Duke University, Durham, USA
[2]School of Computer and Information Technology, Beijing Jiaotong University, Beijing
[3]IBM Corporation

***Abstract*—While Blockchain network brings tremendous benefits, there are concerns whether their performance would match up with the mainstream IT systems. This paper aims to investigate whether the consensus process using Practical Byzantine Fault Tolerance (PBFT) could be a performance bottleneck for networks with a large number of peers. We model the PBFT consensus process using Stochastic Reward Nets (SRN) to compute the mean time to complete consensus for networks up to 100 peers. We create a blockchain network using IBM Bluemix service, running a production-grade IoT application and use the data to parameterize and validate our models. We also conduct sensitivity analysis over a variety of system parameters and examine the performance of larger networks.**

***Index Terms*—blockchain; hyperledger fabric; PBFT; performance modeling; Stochastic reward nets**

I. Introduction & System Description

Reasonable success of the Bitcoin crypto-currency [1] has led to an increasing interest in the technical community for using the underlying decentralized ledger of transactions (called Blockchain) to solve other interesting problems. The blockchain is an encrypted, distributed database/transaction system where all the peers share information in a decentralized, trusted and secure manner. In our work, we focus on the v0.6 of Hyperledger Fabric [2], which is an open-source implementation of distributed ledger platform for running smart contracts in a modular architecture [3]. Thus Bitcoin could be an application running on the fabric.

In a public blockchain network such as Bitcoin, any one can join the network, which induces the risk of Sybil attack. Bitcoin resolves this by making it computationally expensive for a peer to propose a new block of transactions using an approach called proof-of-work (PoW). Although crypto-currency is an interesting application, its performance is worrisome, with time to confirmation of transaction of ten minutes or more, achieving a maximum throughput of 7 transactions per second [4]. In a private blockchain network, all the participants are whitelisted and bounded by strict contractual obligations to behave "correctly", and hence more efficient consensus protocols such as Practical Byzantine Fault Tolerance (PBFT) [5] can be used [6]. PBFT works on the assumption that less than one-third of the peers are faulty (f), which means that the network should consist of at least $n = 3f + 1$ peers to tolerate f faulty peers [5]. Thus $f = \lfloor (n-1)/3 \rfloor$. The network requires $2f + 1$ peers to agree on the block of transactions.

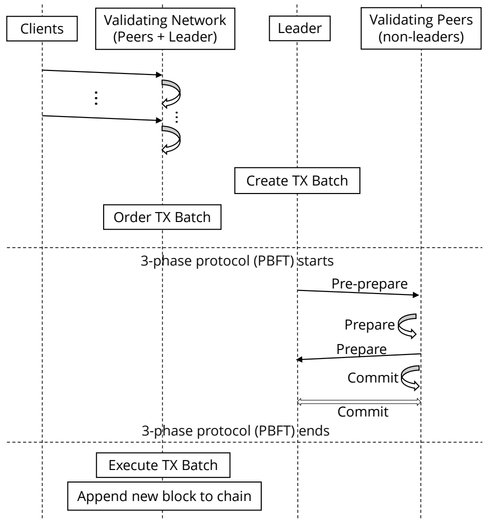
Clients
Validating Network (Peers + Leader)
Leader
Validating Peers (non-leaders)
Create TX Batch
Order TX Batch
3-phase protocol (PBFT) starts
Pre-prepare
Prepare
Prepare
Commit
Commit
3-phase protocol (PBFT) ends
Execute TX Batch
Append new block to chain

Fig. 1. Sequence diagram of transactions on the Blockchain network

Figure 1 shows a high-level view of a permissioned blockchain network. Each participating institution is a Validating Peer (VP), one of which is elected to be the leader. The clients make transaction requests to their respective institution's VP, which validates the transaction and broadcasts it to other VPs. After a few seconds (defined as *batch timeout*) or after a set number of pending transactions (defined as *batch size*), the leader creates a block of the pending transactions, maintaining order by timestamp. Then it broadcasts this candidate block to other VPs to obtain a consensus on the block using PBFT. If $2f + 1$ peers agree, then each VP executes all the transactions and appends the block as the next block on their private ledger. Each block is hashed with the value of the previous block, creating a chain of blocks, and hence the name *blockchain*.

II. Performance model of PBFT consensus process

In our work, we model the "mean time to complete consensus" of the PBFT consensus process using Stochastic Reward Nets (SRN) [7], by capturing the three most time-consuming steps, viz. transmission time of consensus messages between peers (transitions with subscript *Tx*), time to process incoming consensus message (transitions with subscript *Ip*), and time to prepare consensus message for next stage (transitions with subscript *Op*). We make the following assumptions (many of these assumptions can be relaxed if needed):

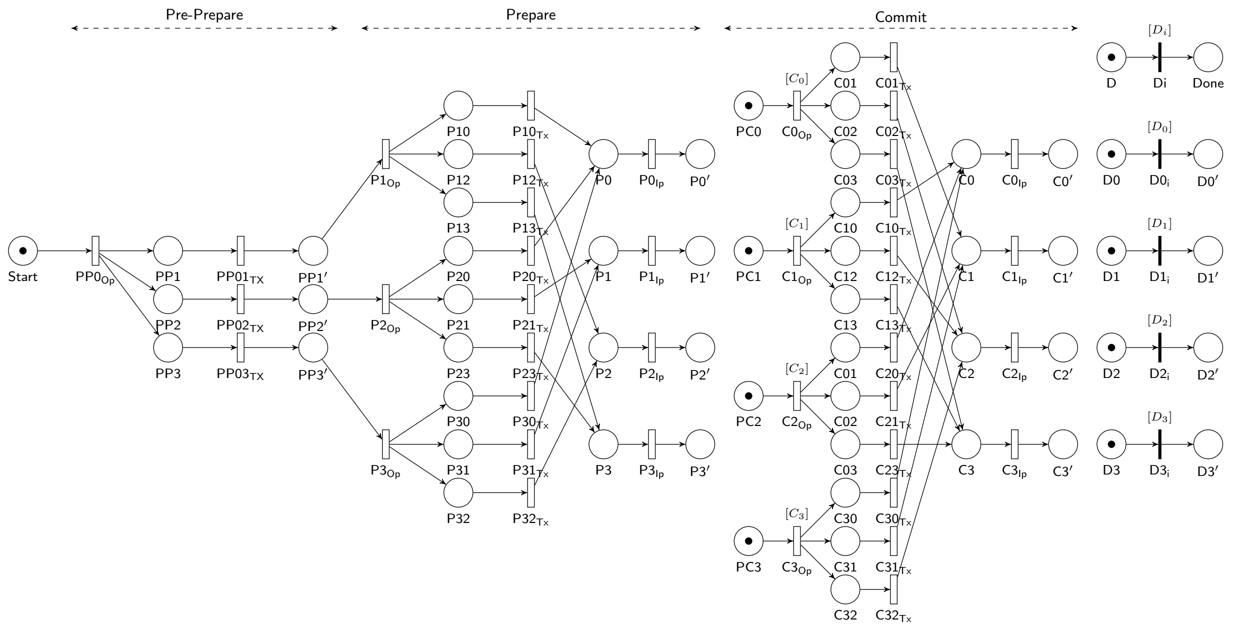

Fig. 2. SRN model for PBFT consensus process with four peers

1) A leader peer is already chosen before the block transaction starts, and it does not change during the execution of the three-phase protocol for a single block.
2) Rate of processing a message at each VP is the same.
3) Rate of message transmission between all pairs of peers is the same.
4) VPs do not fail at any time during the execution of the three-phase protocol for a single block.

In the SRN model in Figure 2, a token in place *Start* signifies that the leader is ready with the new proposed block, and a token in place *Done* signifies the completion of the consensus process. The leader is identified with number 0 and other VPs with the numbers 1, 2, and 3 respectively. The model is intuitive and easy to follow. The model is intuitive and easy to follow. Due to space limitations, we ignore the detailed explanation of the model.

TABLE I
GUARD FUNCTIONS FOR SRN MODEL IN FIGURE 2

Guard Name	Guard Function
[C$_0$]	If #P0$' \geq 2f$, return 1, else return 0
[C$_x$], $x \in (1, 2, 3)$	If #P$x' \geq 2f - 1$, return 1, else return 0
[D$_x$], $x \in (0, 1, 2, 3)$	If #C$x' \geq 2f$, return 1, else return 0
[D$_\mathrm{i}$]	If $\sum_{y \in (0,1,2,3)}$ #D$y' = 3f + 1$, return 1, else return 0

III. ANALYSIS RESULTS

We setup the blockchain network with $n = 4$ and $f = 1$ using the IBM's Bluemix service, running an IoT application [8] developed by IBM Watson team. For a system running over few weeks with around 800 blocks committed per hour, we analyze the logs for 50 randomly chosen blocks. We fit the datasets with Exponential, Weibull, Gamma, Hypoexponential (2-stage, 3-stage), LogNormal and Pareto distributions using Maximum Likelihood Estimate (MLE) technique. We evaluate the goodness-of-fit using Kolmogorov-Smirnov (KS) statistic at 5% level for significance and select the distribution with the lowest Akaike information criteria (AIC). The best-fit models are as follows: Weibull (shape = 2.092, scale = 0.8468) for transitions with subscript *Tx*, Weibull (shape = 1.561, scale = 0.124) for transitions with subscript *Ip*, Weibull (shape = 1.509, scale = 0.2575) for transitions with subscript *Op* in pre-prepare and prepare phase, 2-stage Hypoexponential (λ_1 = 22.050, λ_2 = 267.97) for transitions with subscript *Op* in commit phase. Since the firing time corresponding to transitions in our model have non-exponential distribution, we use SPNP tool in simulator mode only (as opposed to computing analytical-numerical solution). For our model, the total time to deposit a token in *Done* place corresponds to the "time to complete consensus" for a block. We conduct 5000 runs and consider the average total time value (along with confidence interval) as our result. Due to the presence of outliers in the empirical results, we compare the estimated mean time to consensus from our model (3.0815 ms) with the median empirical results (3.314 ms). With a relative error of about 7%, we find the results comparable and our model validated.

A. Sensitivity Analysis

Assuming all the peers are equidistant from each other, we increase the time to transmit (*Tx*) the message keeping the other parameters constant, and find that the mean transmission time increases with a slope of 3.0539, which makes sense since messages are transmitted in three phases of the consensus process. In the same manner, if we increase time to prepare consensus message *Op* in preprepare and prepare phase, the

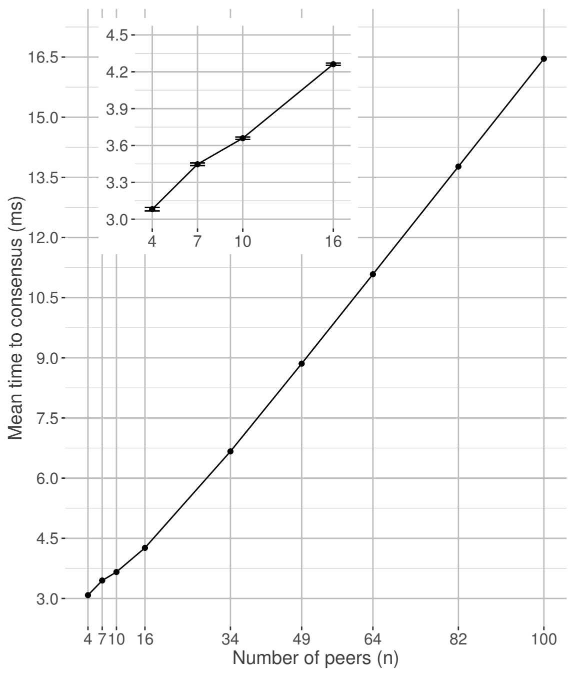

Fig. 3. Mean time to consensus for large number of peers (mean Tx = 0.75ms)

mean time to consensus increases with a slope of 1.89; and for the time to process incoming message *Ip*, the mean time to consensus increases with a slope of 3.309. Thus, a slowdown in handling incoming prepare and commit messages can have a greater impact on the mean time to consensus than the slowdown in preparing message for the new phase.

B. Large number of peers

We generate models for larger values of n, where $n = 4, 7, 10$ for $f = 1, 2, 3$, respectively, up to $n = 100$. We evaluate the model using the parameters from our experimental validation. In the analysis up to 10 peers (inset of Figure 3), we find that the time to consensus increases with n, however, the slope decreases at $n = 7$. Since $2f + 1$ consensus messages are required by each peer in the prepare and the commit phases, the proportion of peers required for consensus decreases from three out of four peers (75%) for $n = 4$ to five out of seven peers (71.42%) for $n = 7$, and so on, asymptotically reaching $\frac{2}{3}$ (66.667%). However, the slope starts increasing again after $n = 10$. This happens due to increasing queuing delays for messages in the prepare and commit phases. The slope continues to increase slightly as n increases. Eventually, the mean time to consensus for $n = 100$ is 5.34 times that for $n = 4$.

In our Bluemix setup, the peers are co-located in the same rack. Let us consider a realistic scenario where peers are located in separate regions of the same data centers or different data centers, and hence the mean time to transmit messages will be much larger. Let us assume that the mean time to transmit messages between all pairs of peers is 5 ms. As shown in Figure 4, we see a similar pattern for the mean time to consensus; however, the percentage increase in the mean time

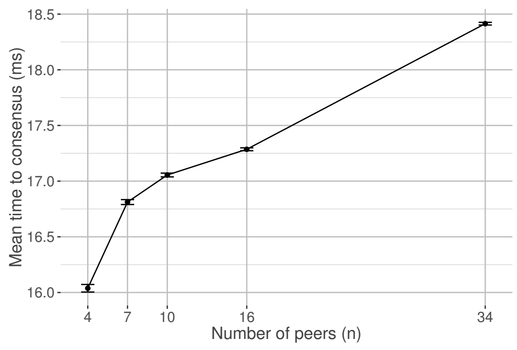

Fig. 4. Mean time to consensus for large number of peers (mean Tx = 5ms)

to consensus for $n = 34$ compared to $n = 4$ is 15%, compared to 116% increase when mean time to transit is 0.75 ms (Figure 3). This percentage increase continues to decrease for networks with even larger mean transmission delays. Thus if the transmission delays are an order of magnitude or two greater than the time to process or queue the message, the mean time to consensus does not increase significantly as n increases.

Thus, our model can estimate performance metrics as a function of various system configurations and parameters and provide early feedback about potential performance bottlenecks. In our ongoing work, we continue to validate the model for a larger number of peers and a wider range of PBFT parameter and system configurations. We are also analyzing other performance aspects, such as block execution.

ACKNOWLEDGMENTS

This research was also supported in part by IBM under a faculty award and an IBM student fellowship. The authors would like to thank Konstantinos Christidis for helping us face-validate our model, Kim Letkeman for running the IoT application on IBM Bluemix setup and related discussions, Mihir Shah and Dale Avery for facilitating the data collection, Nan Wang for support with the SPNP package.

REFERENCES

[1] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," https://bitcoin.org/bitcoin.pdf.
[2] "Hyperledger Fabric v0.6," http://web.archive.org/web/20160924231627/ http://hyperledger-fabric.readthedocs.io/en/latest/protocol-spec.
[3] C. Cachin, "Architecture of the Hyperledger Blockchain Fabric," https://www.zurich.ibm.com/dccl/papers/cachin_dccl.pdf.
[4] K. Croman, C. Decker, I. Eyal, A. E. Gencer, A. Juels, A. Kosba, A. Miller, P. Saxena, E. Shi, E. Gün Sirer, D. Song, and R. Wattenhofer, *On Scaling Decentralized Blockchains*. Springer Berlin Heidelberg, 2016, pp. 106–125.
[5] M. Castro and B. Liskov, "Practical Byzantine Fault Tolerance and Proactive Recovery," *ACM Trans. Comput. Syst.*, vol. 20, no. 4, pp. 398–461, Nov. 2002.
[6] T. Swanson, "Consensus-as-a-Service: a brief report on the emergence of permissioned, distributed ledger systems," http://www.ofnumbers.com/wp-content/uploads/2015/04/Permissioned-distributed-ledgers.pdf, 2015.
[7] J. K. Muppala, G. Ciardo, and K. S. Trivedi, "Stochastic Reward Nets for Reliability Prediction," in *Communications in Reliability, Maintainability and Serviceability*, 1994, pp. 9–20.
[8] "IBM Watson IoT Track and Trace contract - GitHub," https://github.com/ibm-watson-iot/blockchain-samples/.