Online Appendix: More Examples for PoPCorn Consensus

Anonymous Authors

1 Examples roadmap

In this online Appendix, we are going to show more examples for PoPC leader election in terms of penalizing and rewarding servers. We present the penalization in leader election in Section 2 and discuss the rewarding for correctly behaved servers in Section 3.

2 Penalizing leader election

First, let us recall the penalization in PoPC leader election. Every server that initiates an election campaign is penalized by limen increment according to the following equation.

$$limen^{(n+1)} = (newEpoch - epoch) \times limen^{(n)} + 1 \qquad (1)$$

For example, in Figure 1, suppose S_i initiates an election campaign after replicating $\mathcal{TB}^{(n)}$ where n is the *id* of this $\mathcal{TB}$. Before the initiation, S_i's limen and epoch are p and q, respectively. Since correct servers increment their limens by one when initiating new election campaigns, per Equation 1, S_i updates its epoch to q+1 and increases its limen to p+1. Then, if S_i does not own any credit, S_i needs to perform hash computation through PREPARE-ELECTION until the result prefixes p+1 consecutive and identical bytes, e.g., r in Table 1.

$\mathcal{TB}$:"popcorn"	n:078d27487e7ff030d9df5e5d2cba5758	l=5	i=1656
r:eeeeec9664000d3dcddcedf87df046685a9079a2378d9608234e1e31d3e0e7b3			

Table 1: Applying SHA-256, algorithm PREPARE-ELECTION repeatedly combines $\mathcal{TB}$ and a randomly generated nonce (n) and hashes the combination until the result (r) meets the requirement of $limen$=5.

Generally, facing with a higher limen results in more iterations for obtaining a qualified result, which requires more computations. To collect votes from all of the other servers, a server must comply with these requirements since a nominee needs $n-f$ votes from the system. Election blocks ($\mathcal{EB}$) record information such as limens and epochs for leaders, so every server can keep track of the information for validating requests from nominees. Consequently, servers are penalized indiscriminately for initiating leader election campaigns. If a server repeatedly initiates election campaigns to launch takeover attacks, then the server is penalized repeatedly with incremented limens.

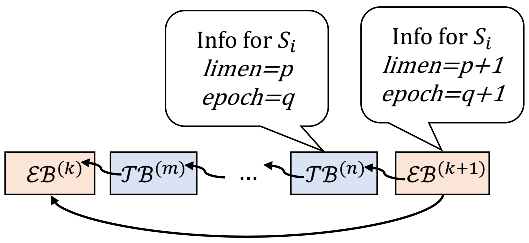

Figure 1: Penalization of leader election.

3 Rewarding correctly-behaved servers

Besides the examples provided in Section 3.6.3, we show more examples to help the understanding of the reward mechanism. The reward for servers is limen deductions associated with credit. Recall that *credits* is the number of unconsumed transaction blocks, and Λ is the set of a server's all the past recorded limens.

$$\delta = \begin{cases} 0 & \text{if } \dfrac{|\mathbb{E}|}{|\mathbb{T}|} > \alpha \\ \max\{0, \dfrac{limen - \mu_\Lambda}{\sigma_\Lambda} \times \dfrac{credit}{|\mathbb{T}|}\} & \text{otherwise} \end{cases} \qquad (2)$$

We set α=1 in the paper, and in Equation 2, if the number of election blocks ($|\mathbb{E}|$) is more than the number of transaction blocks ($|\mathbb{T}|$), no limen deductions apply. Because when $\frac{|\mathbb{E}|}{|\mathbb{T}|}>1$, a PoPCorn system is experiencing frequent leader rotations, which is an undesired situation that sabotages system availability. Thus, PoPCorn discourages leader election by only penalizing servers.

On the other hand, limen deduction may apply when the system is more stable than the expected α. In this case, the deduction is the multiplication of the normalization of limens and the ratio of *credit* to $|\mathbb{T}|$.

$$\because \frac{limen - \mu_\Lambda}{\sigma_\Lambda} \leqslant limen \ , \ \frac{credit}{|\mathbb{T}|} < 1$$
$$\therefore 0 \leqslant \delta < limen$$

The reason for applying normalization to *limen* is to judge servers' past behavior, where Λ includes all the limens recorded in election blocks. Normally, correct servers do not have a sharp increase in limens because correct servers follow the protocol of PoPC leader election and initiate leader election campaigns based on timeouts. However, Byzantine servers have to fortify the leader position to undermine system availability so that they need to launch takeover attacks whenever they are not the leader; limens for Byzantine servers have a sharper increase, resulting in a higher μ_Λ and σ_Λ. Therefore, in this case, the normalized limens of Byzantine servers are smaller than that of correct servers, thereby reducing the reward, limen deductions, for Byzantine servers.

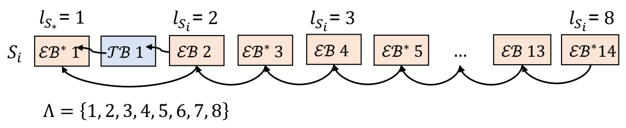

Figure 2: Limen deduction for Byzantine servers.

3.1 Rewards to Byzantine servers

Figure 2 shows an example that S_i has launched 8 takeover attacks. A Byzantine server, S_i, initiates election campaigns whenever S_i is not the leader. $\mathcal{EB}*$ is an election block produced by correct servers, and $\mathcal{EB}$ is produced by S_i where l_{S_i} represents the limen of S_i. After launching the 8-th attack, S_i cannot afford the computation and decide to apply limen deduction.

$$\Lambda = \{1,2,3,4,5,6,7,8\}.$$

Thus, μ_Λ=4.5, σ_Λ=2.29, and the normalized limen is 1.52. To obtain a limen deduction, i.e., δ=1, the ratio of *credit* over $|\mathbb{T}|$ should be at least 0.87, which means S_i needs to stop launching attacks and accumulates credits by behaving as correct servers until the ratio reaches to 0.87. After that, S_i is able to launch the 9 th takeover attack with *limen*=7, and the 10 th attack with *limen*=8 (S_i is penalized again). At this time, Λ includes two new values:

$$\Lambda = \{1,2,3,4,5,6,7,8,7,8\}.$$

Therefore, μ_Λ=5.1, σ_Λ=2.39, and the normalized limen is 1.22. To obtain a limen deduction, i.e., δ=1, the ratio should be at least 0.89. Then, after accumulating credits, again, S_i launches two attacks (the 11 th attack with *limen*=7 and the 12 th attack with *limen*=8), and two new limens are recorded to Λ:

$$\Lambda = \{1,2,3,4,5,6,7,8,7,8,7,8\}.$$

Hence, μ_Λ=5.5, σ_Λ=2.36, and the normalized limen is 1.06; the ratio should be at least 0.94. Repeatedly, S_i performs the 13 th and the 14 th attacks, and the Λ contains more limens:

$$\Lambda = \{1,2,3,4,5,6,7,8,7,8,7,8,7,8\}.$$

Then, μ_Λ=5.79, σ_Λ=2.30, and the normalized limen is 0.96. In this case, no matter how many credits S_i obtains, no limen deduction can apply because the result after the normalization is less than 1. If the computation resources of S_i can only support the hash computation with requirements of *limen*=8. Then, S_i exhaust its computation ability, vanishing into performing computations in the next takeover attack, and the PoPCorn system continues to provide stable services in the presence of the Byzantine server.

3.2 Rewards to correct servers

The increase of limens of correct servers is smoother compared with that of Byzantine servers. Usually, correct servers do not face with high limens because correct servers do not intentionally initiate new election campaigns. In our evaluation in a PoPCorn system consisting of 16 servers, we did not observe any correct servers reaching to a limen of 8. In order to show the comparison, we misconfigured a correct server, $S_\Diamond$, with a shorter *commitTimeout* so that $S_\Diamond$ is more likely to firstly detect leader failures and initiate election campaigns. The recorded limens of $S_\Diamond$ is shown below when the Byzantine servers takes over the leader position from $S_\Diamond$ in the 14 th attack.

$$\Lambda = \{1,2,3,4,5,4,3,4,3,4,5,4,5\}$$

To calculate the limen deduction, μ_Λ=3.62, σ_Λ=1.15, and the normalized limen is 1.21; the ratio should be 0.83. Note that $S_\Diamond$'s current limen is only 5 compared with the Byzantine server. The reason is that when *limen*>5, the computation takes much longer than the maximum *commitTimeout* of correct servers so that the other correct servers that have smaller limens obtain qualified results before $S_\Diamond$ does, though $S_\Diamond$ has a misconfigured timer.

To conclude, Byzantine and correct servers are both penalized for initiating election campaigns. In order to attack PoPCorn systems, Byzantine servers have to perform increasingly costlier computation to keep the leader position. However, correct servers amortize the cost and are benefited from the limen deduction mechanism. Ultimately, Byzantine servers exhaust their computation abilities, thereby vanishing into performing computations to becoming future leaders.