Ripple Labs Byzantine Consensus Algorithm

Konzer A. Cason

Student: University of Michigan- Mathematics

Author Note

Written under Dr. Cam McLeman – MTH 200: Proofs and Structures

Abstract

Cryptocurrency developers at Ripple Labs break down the Ripple (XRP) consensus protocol. The basic concept of a decentralized accounting system is introduced. Basics of consensus algorithms are introduced on past foundations before diving into the specifics of the Ripple Protocol Consensus Algorithm (RPCA). Specifics of agreement, consensus and utility are described as the core network principles. A deeper dive into the Byzantine Generals problem is addressed with bounds introduced for both consensus and agreement.

Keywords: Cryptocurrency, Protocol, Server, Node, Set, Probability, Byzantine Generals, Ripple

Ripple Labs Byzantine Consensus Algorithm

Ripple Labs set their sights on the next big cryptocurrency following the blossom and general adoption of well-known Bitcoin. In their whitepaper on the Ripple Protocol Consensus Algorithm (RPCA), Schwartz, Youngs, and Britto dive into the foundations they built upon and the specifics of their network. Following an introduction to consensus systems, a formal framework is introduced for such systems. The consensus algorithm is running on a network of servers, entities running the ripple server software, which create a node based graphical network of transaction verifiers. Each confirmed transaction is submitted to a community ledger, documenting transfers of XRP, the ripple coin, on the ripple network. Two main ledgers are recorded on the network. The last closed ledger is the last know verified ledger available to all nodes, a system log of historic transactions verified by a consensus of nodes. The open ledger is the current, real-time, log of transactions yet to be, or currently being verified. Each server, or node, references a unique node list (UNL) of other nodes in the network it must double check each transaction against. A server itself becomes a proposer by broadcasting its own verifications to the public, for those nodes who's UNL includes it to check against.

Within a network, two types of entities are defined. Non-faulty nodes, or good actors, are those servers contributing to the network while following community guidelines and only confirming verified transactions. Faulty nodes, or bad actors, are those servers with intention to manipulate the network, looking to double spend, spend one currency twice simultaneously to double its value, or introduce malignant transactions that remove XRP tokens from one account and deposit into another, without consent from both parties. The consensus problem is now set

with three conditions: 1 – Each good actor decides in a finite time. 2 – All good actors approve/deny the same sets of transactions. 3 – Each consensus reached is binary, either approved or denied.

The basis of any consensus algorithm relies on three main metrics, correctness, agreement, and utility. The correctness metric introduces the Byzantine Generals problem. This problem is defined as follows, each node is to report approval/denial for each transaction, any node may be a good/bad actor, how can we be certain a correct consensus is reached even in the presence of bad actors? The caveat is that there is a limit on the allowed bad actors, in any case a network with a majority bad actors could delude the good actors. The agreement metric begins to introduce some graph theory in order to assure each node will have an identical consensus for each transaction, no matter the proposers they verify against on their UNL. In general, a minimum connectivity is required in order to assure there is enough cross-referencing taking place such that it will scale the whole network, even in the presence of bad actors. The last metric, utility, addresses the usefulness of the network. One common way to measure this is convergence. As it is required good actors verify transactions and submit their proposal in finite time, a network with a majority of good actors will be sure to converge. This metric is the intrinsic measure of transaction speed, or the time it takes for XRP sent from one wallet to be spendable in the next. Each of these metrics have room for optimization based on stipulations what defining the RCPA.

The RPCA Correctness algorithm assumes that the percentage of bad actors is under, or equal to, twenty percentage of the total nodes within the network and is as follows:

$$p^* = \sum_{i=0}^{\lceil (\frac{n-1}{5}) \rceil} \binom{n}{i} p_c^i (1-p_c)^{n-i}$$

Within the paper the justification of this formula is very vague and left up to the author to interpret. p^* represents the probability of correctness, or the probability that there will be no more that 20% failures within the network. p_c represents the probability that a node is a bad actor and will thus at some point collude with other bad actors in attempt to fool the network. Form here it becomes clear that $(1 - p_c)$ is the probability of any selected node being a good actor. The important concept is the recognition that inside the sum is a binomial distribution given for a selected number, i, of bad actors. For example, in a network of 10 nodes, the probability of 1 bad actor, given a 10% chance of bad actors is, $\binom{10}{1}(0.1)^1(0.9)^9 = 0.3874 = 38.74\%$. This formula is stating that there are 10 unique ways to select 1 node from a 10 node network, such that there is 1 node a bad actor, with a probability given by $p_c = 0.1$, and 9 good actors, with a probability given by $(1 - p_c) = 0.9$. For a single instance the chance of it existing can be given by the product of the probability each individual node is as we define. Because there are 10 unique instance, we sum all 10 to get the total likelihood there is exactly 1 bad actor, on any such node, in a network of 10 nodes. Looking now at the external sum, $\left\lceil (\frac{n-1}{5}) \right\rceil$ represents the 20% limit on bad actors for any network with node count n. p^* is now well defined as the probability of any unique selection of bad actors within a network, ranging from 0-20% of the total network having bad actors. It becomes clear that this is the probability of correctness and it is the probability the total number of bad actors is under the defined limit required for consensus. Looking back at the previous example, $p^* = 0.9298 = 92.98\%$. When reviewing this equation, one unclear factor is how the RPCA arrived at the 20% metric. A question for Ripple Labs, what are the factors influencing the 20% bound on correctness within the algorithm, and why is the bound for the RPCA tighter than the other algorithms mentioned in the whitepaper?

The Agreement algorithm introduced some new graph theory to my mathematical language dictionary and is as follows:

$$|UNL_i \cap UNL_j| \geq \frac{1}{5}\max(|UNL_i|, |UNL_j|) \forall i, j$$

This equation is a limit on the connectivity of the nodes within the network. The left-hand side references the number of edges, and the right-hand side reference 20% the size of the larger UNL. The requirement of this stipulation within the network is to prevent a fork within the network. Such an instance would result in 2 conflicting open ledgers on the network, violating the condition that all good actors reach the same conclusions, given correctness is imposed. The intuitive reasoning for this is that given the number of edges between two UNLs is less that 20% of the larger UNL cardinality, it is possible that the select edge nodes are all bad actors. Referencing Figure 1, this becomes more obvious from a graphical representation. Ripple Labs includes the idea that allowing for indirect edges, the edge constraint can be tightened as the network becomes more entangled. To expand on this idea, the previous example is modified to allow this in Figure 2. As the edge constraint is no longer valid, I will solve for both the left- and right-hand side while dropping the requirement. From the image, the top graph displays 3 edges, and the bottom 4. Additionally, the left UNL has 6 nodes, and the right 5. From this we can derive that $4 \geq C(6)$. Here C represents the new edge constraint. To keep consistent, it is also clear that $3 < C(6)$, as the top set of UNLs should still return faulty. Solving the system of inequalities, we have $\frac{2}{3} \geq C > \frac{1}{2}$. As mentioned above, this does tighten the connectivity requirement as any possible value for C is greater than the previous 20% requirement.

In conclusion, Ripple Labs produced this whitepaper to inform the audience of the RPCA. They made sure to highlight the major factors that backed the validity of their monetary

transfer system. The audience of this paper consists of a plethora of individuals. The main reader would be the cryptocurrency enthusiast with a strong knowledge of the crypto space. Some other readers include cryptography enthusiasts, potential investors, and independent auditors. The main point being conveyed by the author is that Ripple offers a new consensus network that is both quicker and less energy intensive than the others which allow for a higher number of bad actors. The desired outcome of the authors is to gain support and investors for the project.

References

Schwartz, D., Youngs., N Britto, A. (2014). The Ripple Consensus Algorithm. *Ripple Labs Inc*.

Figures

Figure 1: UNL Connectivity

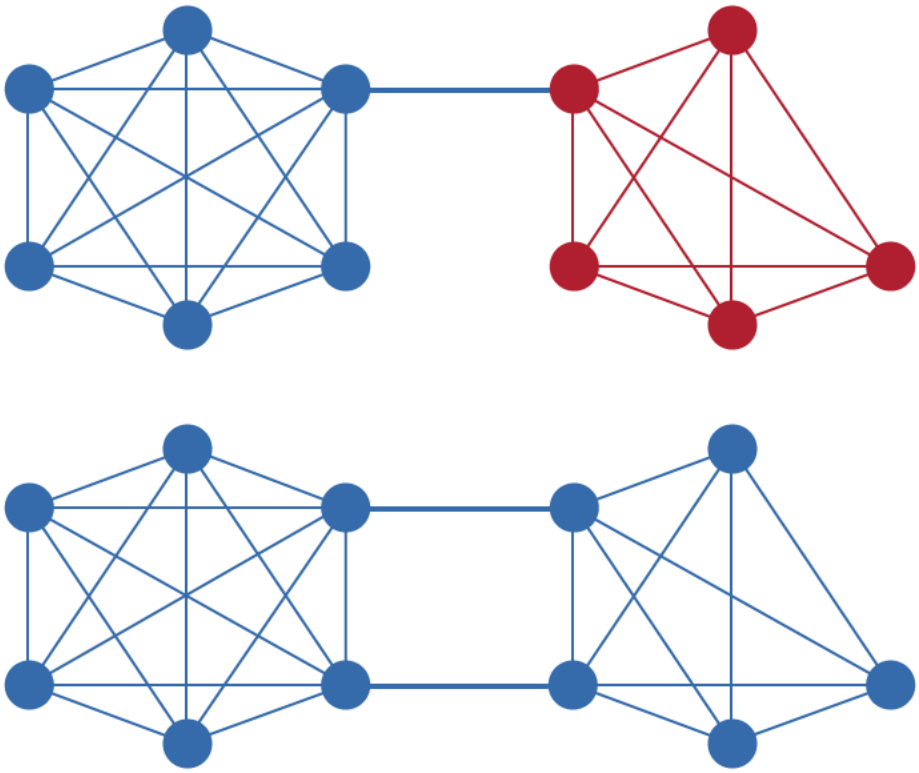

Figure 1. An example of the connectivity required to prevent a fork between two UNL cliques. The top network does not satisfy the connectivity requirement, the bottom network does. (Schwartz, 2014).

Figure 2: UNL Connectivity including Indirect edges.

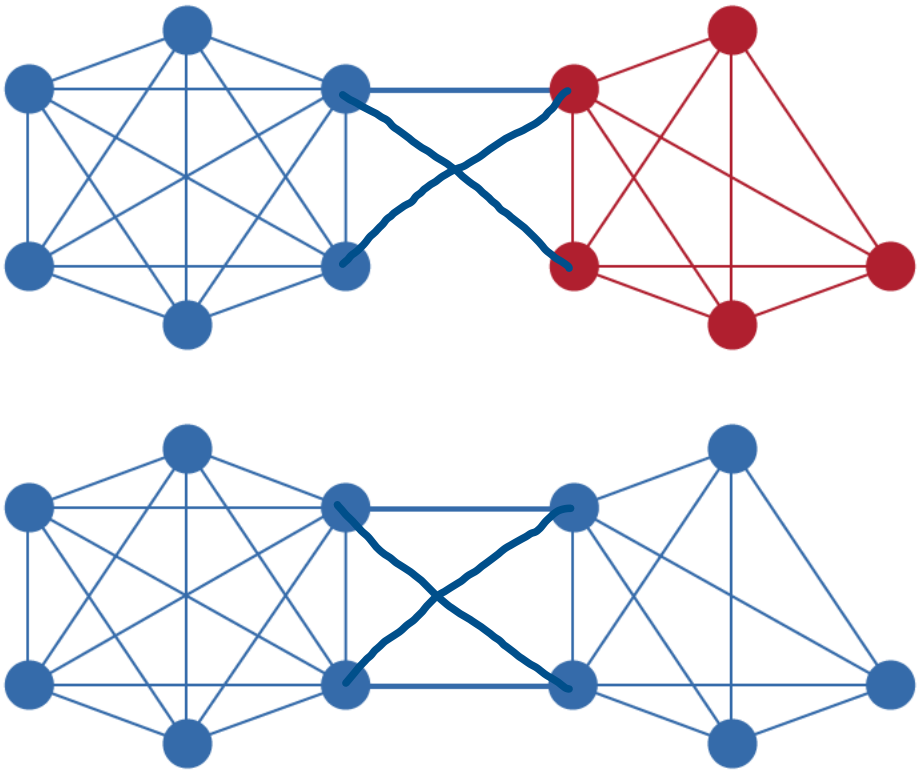

Figure 2: A n example of the connectivity required to prevent a fork between two UNL cliques, when allowing for indirect edges. The top network does not satisfy the connectivity requirement, the bottom network does.