Analysis and Characterization of Consensus Correctness in Ripple (XRP) Networks

***Abstract*—Ripple network or the XRP network is one of the most versatile blockchain platforms used worldwide for payment systems, healthcare applications etc. The abstract protocol called XRP ledger consensus protocol, XRPL in short (called Ripple Protocol consensus algorithm (RPCA by the initial designers) is based on Byzantine fault-tolerant (BFT) agreement protocol but does not use the standard models or implementation but utilizes collectively-trusted sub-networks within a large network. Consensus is achieved by maintaining a certain level of "trust" for the sub-networks and a certain minimal connectivity throughout the network so that the network can be robust in the face of Byzantine failures. For each server in the XRP network, there is specified a sub-network or validator network called the Unique Node List (UNL) consisting of a subset of the servers of the whole network. To be robust against Byzantine failures, XRPL enforces 80% quorum and a certain overlap of nodes across the UNLs. The overlap was initially specified to be 20% and was later enhanced to be greater than 90% to satisfy conditions of safety and liveness. However, even with such an enhancement, safety and liveness issues have been pointed out by researchers.**

In this paper, we characterize, the XRP Ledger Consensus protocol (abbreviated XRPL) for consensus correctness using a notion of similarity metric of rand-index used for cluster analysis of networks. We establish that XRPL with 80% quorum and UNLs satisfying 50% similarity, is robust against 20% failures. We further establish that with 80% quorum and 50% similarity, XRPL preserves safety and transaction liveness.

Blockchain, POW, POS, Consensus, Correctness, Forking

I. Introduction

The XRP network is a blockchain-based distributed payment system that enables users to transfer funds seamlessly around the world. One of the main challenges for any of the electronic cash system has been the avoidance of double spending. Transactions in an XRP rely on a consensus protocol to validate account balances and prevent double spending. The initial protocol designed [1] was referred to as RPCA (Ripple Protocol Consensus Algorithm) and subsequent refined consensus protocols go under the general name of XRP Ledger Consensus Protocol (XRPL for short). These consensus protocols basically use Byzantine fault tolerant agreement protocol over collectively trusted sub-networks. Note that the Ripple network does not run with a proof-of-work (POW) system like Bitcoin or a proof-of-stake (POS) system like Ethereum but achieves a good scalability and performance; this is one of the main reasons for Ripple to have been widely adopted for payment systems worldwide. The crux of payment systems lies with consensus correctness and is briefed below:

1) Consensus:
   (C1): Every non-faulty node makes a decision in finite time.
   (C2): All non-faulty nodes reach the same decision value
   (C3): 0 and 1 are both possible values for all nonfaulty nodes. (This removes the trivial solution in which all nodes decide 0 or 1 regardless of the information they have been presented.
2) No-Forking: There is no possibility of two blocks being attached to the ledger at the same time (that is ledger extension is deterministic) and thus, assuring that there is no possibility of double spending.
3) Finality or Liveness: Eventual happening of the transaction requested.

Other properties based on fairness etc., will not be looked at in this paper.

There has been a number of explorations in the literature [1], [2], [3], [4], [5], [6] on various aspects of Ripple protocol like safety, forking, liveness etc.

A brief account of the results in terms of consensus correctness is briefed in the following. The original white paper by Schwartz et al.[1] describe the UNL model, and claim that with 80% quorum requirement for consensus, it is necessary to have 20% common nodes across UNLs to avoid forking. The work by Armknecht et al.[2] show that the overlap of 20% is not sufficient for reaching consensus and every pair of nodes needs an overlap of at least 40%. The exploration in [3] shows that the minimum overlap of 90% of the UNLs is needed for realizing consensus and establishes that liveness is violated even with 99% overlap of UNLs even without faulty nodes. Security analysis has been made in [4] starting from abstracting the consensus protocol from the code and has shown issues of safety and liveness of XRPL. [6] looks atsome of the issues using a formal approach and [5] explores graph theoretic properties of the network.

Study of several works briefed above indicates that just a structural overlap of node may not be enough and the pattern of shared communications in the underlying sub-networks of the UNLs play a vital role.

In this paper, we characterize consensus correctness in terms of a "similarity measure" of UNLs that captures the structure of overlap and the communications that should be shared across UNLs. Our similarity is based upon the metric *rand-index (RI)* that has been used for cluster-analysis [7], [8]; recently it has also been used for analysing byzantine failures[9] but not for consensus correctness. We establish that XRPL with 80% quorum maintains consensus correctness with 50% similarity and further preserves safety and transaction liveness. While several authors had analysed the XRPL protocol for security and correctness, this is the first time, a characterization

of consensus correctness has been established using a easy-to-understand metric rand-index.

Rest of the paper is organized as follows: after introduction in section II, the XRP protocols are reviewed in section II. Section III is entirely devoted to consensus correctness of XRPL; it further describes a similarity measure for UNLs and establishes the robustness of XRPL under 80% consensus with 80% similarity; relationship between overlap of nodes in and the similarity measure based on the metric of rand-index for UNLs is also described. In section IV-B, it is shown that 50% similarity is optimal for 80% consensus and establishes XRPL preserves safety and liveness with 50% with 80% quorum. The paper concludes with section V.

II. XRPL: An Overview

The XRP network has one of the highest cryptocurrency market capitalizations and is aimed at fast global cross-border payments, asset exchange and settlement. The main challenges of XRP ledger is to prevent double spending and assure network-wide consensus on the state of user accounts and balances. The consensus protocol referred to as, XRP Ledger Consensus Protocol (abbreviated XRPL), was called Ripple Consensus Protocol Algorithm (RPCA) in the initial design phase [1], is generally considered to be based a Byzantine fault-tolerant agreement protocol. It uses a Byzantine fault tolerant agreement protocol over collectively trusted sub-networks. Note that the XRP network does not run with a proof-of-work (POW) system like Bitcoin or a proof-of-stake (POS) system like Ethereum but guarantees consistency with only a partial agreement on who participates, allowing a decentralized open network. Thus, XRPL does not depend on mining but uses a voting process based on validator nodes of each of the servers. It is through such a voting process, it but achieves a good scalability and performance. This is one of the main reasons for XRP to have been widely adopted for payment systems worldwide.

The main function of XRPL is only to make the network reach agreement on sets of transactions and not on the content or outcome of those transactions. That is, it provides a common global order for the transactions submitted by clients to all participating nodes in spite of faulty or malicious (Byzantine) nodes.

For a good understanding of evolution of the correctness of consensus algorithms, we present:

1) firstly, the initial designed protocol called Ripple Protocol Consensus Algorithm (RPCA) [1], followed by
2) the XRPL protocol that handles the merging of forks that arise due to the underlying asynchrony of the XRP network; note that the algorithm remains deterministic.

Before going, XRPL (or RPCA) protocol components and the terminology used are briefed below.

Ripple Protocol Components and Terminology: It consists of [1]:

1) *Server*: A server is any entity running the XRP Server software (as opposed to the XRP Client software which only lets a user send and receive funds), which participates in the consensus process.
2) *Ledger*: It is a record of the amount of currency in each user's account and represents the "ground truth" of the network. The ledger is repeatedly updated with transactions that successfully pass through the consensus process.
3) *Last-Closed Ledger*: IT is the most recent ledger that has been ratified by the consensus process and thus represents the current state of the network.
4) *Open Ledger*: It is the current operating status of a node (each node maintains its own open ledger). Transactions initiated by end users of a given server are applied to the open ledger of that server, but transactions are not considered final until they have passed through the consensus process, at which point the open ledger becomes the last-closed ledger.
5) *Unique Node List (UNL)*: Each server, s, maintains a unique node list, which is a set of other servers that s queries when determining consensus. Only the votes of the other members of the UNL of s are considered when determining consensus. Thus, the UNL represents a subset of the network which when taken collectively, is "trusted" by s to not collude in an attempt to defraud the network. Note that this definition of "trust" does not require that each individual member of the UNL be trusted.

   For any server node P_i, UNL_i denotes its' UNL list, $n_i = |UNL_i|$, a parameter called *quorum*, q_i denotes the number of nodes in UNL_i from whom P_i should get a concurrence before committing to a decision.
6) *Proposer*: Any server can broadcast transactions to be included in the consensus process, and every server attempts to include every valid transaction when a new consensus round starts. During the consensus process; however, only proposals from servers on the UNL of a server s are considered by s for taking a decision.

A. Ripple Protocol Consensus Algorithm (RPCA)

The broad steps of of RPCA [1] given below:

1) Initially, each server takes all valid transactions seen prior to the consensus round that have not already been applied and makes them public in the form of "candidate set". It includes new transactions initiated by end-users of the server and the transactions held over from previous consensus process.
2) Each server then amalgamates the candidate sets of all servers on its UNL, and votes on the veracity of all transactions.
3) Transactions that receive more than a minimum percentage of "yes" votes are passed on to the next round, transactions that do not receive enough votes will either be discarded or included in the candidate set for the beginning of the consensus process on the next ledger.
4) Final round of consensus requires a minimum percentage of 80% of a server's UNL agreeing on a transaction

(this is treated as quorum). All transactions that meet this requirement are applied to the ledger, closing it - new last-closed ledger.

B. Characteristic Properties of RPCA Consensus correctness Argued in [1]

Some of the salient properties that have been argued in [1] are given below:

1) Assuming RPCA is correct, it must be the case that if 80% of the UNL of a server agrees with the transaction, the transaction will be approved. Thus, for an UNL of n nodes in the network, the consensus protocol will maintain correctness so long as $f \leq n - 1/5$ where f is the number of Byzantine failures.
2) Since the UNL for each server can be different, agreement across servers is not inherently guaranteed by the correctness proof. For instance[1], if there are no restrictions on the membership of the UNL, and the size of the UNL is not larger than $0.2 * n_{total}$ where n_{total} is the number of nodes in the entire network, then a fork is possible. An simple example of two cliques connected without any common members can achieve independent decisions leading to *forks*. If the connectivity of the two cliques surpasses $0.2 * n_{total}$, then a fork is no longer possible, as disagreement between the cliques would prevent consensus from being reached at the 80% agreement threshold that is required. It is further argued that a fork becomes much more difficult to achieve, due to the greater entanglement of the UNLs of all nodes.
   - It must be noted that there are no assumptions about the nature of intersecting nodes like Byzantine or honest.

   Thus, according to [1], RPCA achieves correctness and agreement as long as 80% quorum and the intersection of UNLs surpasses $0.2 * n_{total}$.

Implicit Assumptions on the Network in RPCA [1]:
It is assumed that

1) The network is peer-to-peer.
2) For every node P_i and $P_j \in UNL_i$, there is a reliable authenticated channel for P_i to receive messages from P_j.
3) Byzantine accountability: all nodes including Byzantine ones cannot send different messages to different nodes. This was assumed such behavior in a peer-to-peer network as it could be identified and corrected by honest nodes. However, due to asynchrony and the possibility of honest nodes being temporarily partitioned. This assumption has been dropped in later versions. Note that the counterexample to safety shown in [4] cannot arise if that assumption is kept. We shall discuss these aspects during our discussion on consensus correctness.

C. XRP Ledger Consensus Protocol (XRPL)

XRPL is a refined version of RPCA, taking into account the possible way the short-term forks arising due to network asynchrony are merged with progress of time. The broad steps of XRPL [3], [4] consists of the following three phases:

1) **Deliberation**: Deliberation is the component of Ripple consensus in which nodes attempt to agree on the set of transactions to apply towards ledgers they validate. Clients submit transactions to one or more nodes in the network, who in turn broadcast the transaction to the rest of the network. Each node maintains a set of these pending transactions that have not been included in a ledger. Starting from this set, a node iteratively proposes new transaction sets based on the support of individual transactions among the sets proposed by nodes in its UNL as highlighted in the RPCA steps.
2) **Validation**: Once a quorum of validations from the UNL to its server for the same ledger is reached, that ledger and its ancestors are deemed fully validated and its state is authoritative and irrevocable.
3) **Preferred Ledger**: In times of asynchrony, network difficulty, or Byzantine failure, nodes may not initially validate the same ledger for a given sequence number. In order to make forward progress and fully validate later ledgers, nodes use the ledger ancestry of trusted validations to resume deliberating on the network's preferred ledger

It is easy to see that the first two steps capture the major steps of RPCA. A similarity can also be seen to the proof of stake finality gadget steps introduced introduced in [10]. The third one is the one which is the additional phase, that is introduced to merge the short-term forks that arise due to asynchrony; the rule is similar to that of the GHOST rule [11]; we will not go into further details, as the details are not very essential for the discussions; interested readers are referred to [3], [4].

III. Correctness of XRPL Protocol

While we have discussed some of claims on RPCA in section II-B, in the following, we discuss some of the issues of XRPL highlighted by several authors [1], [2], [3], [4].

As discussed in section II-B, [1] tries to arrive at conditions to avoid forking through overlap of nodes in the UNLs and in a sense qualitatively bringing out the connectivity requirements in the XRP network. As highlighted already, the protocols abstracted from the code do not satisfy the requirements of safety and liveness [4].

As mere overlap of UNLs is not able to characterize correctness, let us try to incorporate the connections (communications) of nodes in UNLs that overlap, in some abstract way. In this paper, we shall look at similarity measure used for cluster analysis [7] like rand-index [8] and adapt it for consensus correctness.

As highlighted in [7], Cluster analysis or clustering is the task of grouping a set of objects in such a way that objects in the same group, called a cluster, are more similar, in a way, to each other than to those in other groups (clusters). The notion of a "cluster" cannot be precisely defined, which is one of the reasons why there are so many clustering algorithms like

connectivity models, centroid models, graph-based models etc. For instance, in graph-based models, a clique, that is, a subset of nodes in a graph such that every two nodes in the subset are connected by an edge can be considered as a prototypical form of cluster. Relaxations of the complete connectivity requirement (a fraction of the edges can be missing) are known as *quasi-cliques*. We use cluster and cliques interchangeably often.

Let us first glance at the overlap requirements in UNLs argued in [1] depicted in Figure 1, where the top figure shows two cliques connected by a bridge and bottom one connected by two bridges. As RPCA is based on 80% quorum, [1] is interested to capturing the majority in spite of failures. Hence, the possible impact of connections (communications) between nodes has not be given a seripus thought.

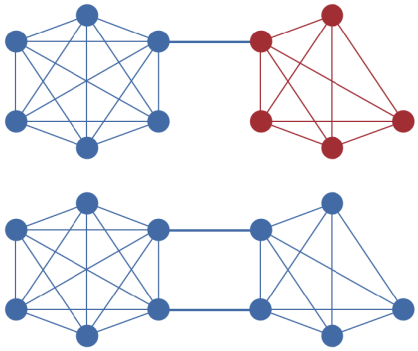

Fig. 1: Connectivity Requirements of two UNL Cliques [1]

To overcome the possibility of fork, [1] arrives at the condition of overlap of nodes in each of the UNLs as well the size of each of the UNLs. The overlap as argued in [1] is shown to be not true in general in [4]. These indicate that that just an overlap of nodes may not be good enough for the correctness.

Let us consider two UNLs, UNL_1 and UNL_2. If they are completely disjoint, then each of the UNL can take an independent decision (thus, easily could lead to forks in an obvious way) as illustrated already.

In the following, we shall adapt the notion of cluster highlighted above [7] that has both overlap and connections and arrive at the notion of *similarity measure* with respect to UNLs. Note that as in [1], *no assumptions are made on the nature of intersecting nodes like honest or Byzantine*. The similarity measure we propose, combines the number of nodes and communications among them and generalizes the concept of rand-index analysis recently envisaged for exploring the BFT properties [9].

Before going into similarity discussions, first let us understand the representation of UNLs as sets of clusters.

A. Representation of UNLs as Voting Clusters

A cluster is essentially a sub-network where each node Byzantine or honest cannot send contradictory messages to members in the same cluster. However, if it is connected through a common node to another cluster, a Byzantine node can send inconsistent or contradictory messages to its directly connected clusters.

A cluster of n nodes could be represented either as a clique of n nodes if every node is connected to every other node, or as a graph $G = (V, E)$ where E denotes the set of edges or communications possible in the network. A graph with n nodes can have a range of edges varying from 0 to $n(n-1)/2)$ edges.

A graph with n nodes can be represented by a union of clusters where member clusters share some of their nodes to denote connections between the clusters. Thus, *we can represent any network as a union of clusters.* For instance, the network shown in Figure 3 could be represented as {{(a,b),(b,c),(a,c)}, {c}, {(c,d), (c,e), (d,e)}}. Here, if c is a Byzantine node, it can send different messages to {a,b} and {d,e}. Similarly

- the top sub-graph in Figure 1,an UNL, can be represented by a set of clusters consisting of the leftmost cluster, the single bridge (one edge), and the rightmost cluster,
- the bottom UNL can be represented through a set of clusters consisting of the leftmost cluster, the two bridges and rightmost cluster.

A *Voting cluster* is a peer-to-peer network that converges on the transactions of its members by voting. For simplicity, we can consider transactions to be just values on which the members are voting. Let us explore the representation of UNLs as clusters. A sub-graph with (i) a single node is a cluster or a clique (one node and no edge), (ii) a single edge is a cluster with two nodes and one connection, (iii) a triangle is a cluster with three nodes and three connections, and (iv) a polygon with n nodes with n nodes and $n(n-1)/2$ edges is a cluster (clique). If the number edges is less than $n(n-1)/2$, we call them *quasi-cluster*.

In a cluster or a quasi cluster, a node even if it is Byzantine cannot send conflicting or contradictory messages to any other member in the same cluster. This will become clear through an example of 3-node cluster shown in Figure [**?**].

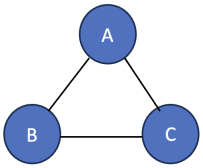

Fig. 2: A Three Node Cluster

Suppose, node A, sends message m, to B and a message m', to C at the same phase or time, B and C in the same cluster, then B and C recognize the byzantine nature of A right away. In the case of two-node cluster, A,B, as cryptography is unbreakable, neither of the nodes can say a different message that the one sent, has been sent by its immediate neighbour; in some sense, A or B will not non-repudiate the messages received between them . However,a Byzantine node in one

cluster can send one message to member(s) in one cluster and another one to member(s) in the other. For example, The UNL shown in Figure can be represented as { {(1,n+1), ... , (n,n+1)},{(n+1)}, {(n+1, n+2), ..., (n+1,2n+1)}. The node labelled $n+1$ could send one message to its left cluster and another one to its right cluster.

B. Rand Index: A Brief Background

Definition 1: Rand Index is a similarity measurement of two data clusters [8]. By *Rand Index* (RI), we can calculate the number of agreements and disagreements in terms of node pairs from the same or different clusters.

Consider the network with N number of nodes, and there are 2 different partitions or clusters of the network, say $C' = (C'_1, C'_2, \ldots, C'_r)$ and $C'' = (C''_1, C''_2, \ldots, C''_s)$. Here the meaning of partition/clusters is that of non-empty disjoint subsets of the network such that their union equals N. Observe that the set of all unordered pairs of the network having $\binom{N}{2}$ pairs is the disjoint union of the following defined sets:
$A = \{$pairs that are in the same clusters under C' and $C''\}$
$B = \{$pairs that are in different clusters under C' and $C''\}$
$C = \{$pairs that are in the same cluster under C' but different in cluster C'' $\}$
$D = \{$pairs that are in different cluster under C' but in the same cluster under $C''\}$
The *Rand Index* measure of similarity of clusters C' and C'' is given by

$$RI = \frac{|A|+|B|}{|A|+|B|+|C|+|D|} = \frac{|A|+|B|}{\binom{N}{2}} = \frac{2(|A|+|B|)}{N(N-1)} \quad (1)$$

Here $|A|+|B|$ can be considered as the number of agreements between C' and C'', and $|C|+|D|$ as the number of disagreements between C' and C''. Notice that as the denominator $|A|+|B|+|C|+|D|$ is the total number of unordered pairs of nodes, so it can be written as $\binom{N}{2}$.

The Rand index has a value between 0 and 1, with 0 indicating that the two data clusters do not agree on any pair of points and 1 indicating that the data clusters are exactly the same.

C. Rand Index Similarity Adaptation for Ripple Analysis

UNLs themselves can be treated as a cluster of clusters of the XRP network. UNLs in Ripple consensus can be treated as two different sets of clusters; the members are essentially cliques or quasi-cliques similar to that described in Figure 1; the top UNL can be described by {left blue-cluster, center-blue-edge, right-red cluster}; similarly the bottom UNL is given by {left-blue-cluster, top blue edge, bottom blue edge, rightmost blue cluster}.

Categories A and B given in the definition of Rand Index above can be interpreted as follows:

1) Category A corresponds to saying as they are in the same clique of cluster, communications can be trusted.
2) Category B corresponds to saying that as they are together either in the same cluster or different clusters together; hence communications can be trusted.

Thus, using A and B, the agreement analysis of any two UNLs provides agreement of the behaviour of the underlying nodes (honest or byzantine) while C and D provide for possible byzantine behaviour by the underlying UNL nodes.

D. Similarity Analysis of UNLs

As highlighted already, it is the categories A and B given in definition of Rand Index that matters for similarity. As UNL is selected from among the total number of nodes in the network, there will be possibly some overlap. Thus, in any UNL, we can identify two components: common members across UNLs and some that are not-common Let us analyse the similarity of UNL clusters based on the common members. For simplicity, to begin with, consider two simple UNLs with two not-common members that share a single common member, two common members and three common members shown in Fig 3, Fig 4 and Fig 5, four common members etc. The similarity or RI of these UNLs are given below:

1) Fig. 3, depicts two clusters corresponding two UNLs, each having two disjoint members, and c is the common member; cluster is abstracted assuming the common member can communicate with all other members in the respective UNLs. The two clusters corresponding to the two UNLs are C'= {(a,b),(b,c),(a,c)} and C" = {(c,d), (c,e), (d,e)}.
   - It is easy to observe that Rand Index (RI) is 0.
   - If c becomes byzantine naturally, it easily follows there will be no consensus.
2) Fig. 4, depicts two clusters corresponding to two UNLs, each having two disjoint members, and c, d are the two common members; cluster is abstracted assuming the common members can communicate with all other members in the respective UNLs. The two clusters corresponding to the two UNLs are C'={(a,b), (a,c1), (a,c2), (b,c1), (b,c2), (c1,c2)} and C"= {(c1,d), (c1,e), (c2,d), (c2,e), (d,e), (c1,c2)}.
   - (c1,c2) is the only member corresponding to clause A and B in Definition 1, between C' and C". Thus, RI is $1 \div \binom{N}{2}$. With N being 6, RI will be 1/15. Again here, consensus need not be there due to the common members becoming byzantine.
3) In Fig 5, on the same lines, we get C'={(a,b), (a,c1), (a,c2), (a,c3), (b,c1), (b,c2), (b,c3), (c1,c2), (c1,c3), (c2,c3)} and C"= {(c1,d), (c1,e), (c2,d), (c2,e), (c3,d), (c3,e), (d,e), (c1,c2), (c1,c3), (c2,c3)}.
   - Here, members corresponding to clauses A and B in Definition 1 are (c1,c2), (c1,c3), (c2,c3).
   - RI = $3 \div \binom{N}{2}$. With N being 7, RI will 3/21 =1/7.
4) Continuing in this manner for say till m common members, RI for that network will be $\binom{m}{2} \div \binom{n}{2}$.

Note: Note that if the non-common members are increased the quorum gets effected (of course the similarity of the UNL also gets affected and hence, there is no need to treat the impact on non-common members explicitly).

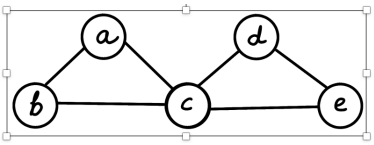

Fig. 3: Two UNLs sharing One Common Member

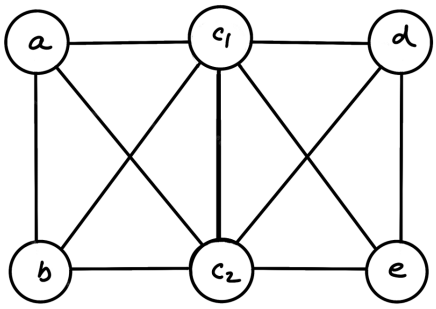

Fig. 4: Two UNLs sharing Two Common Members

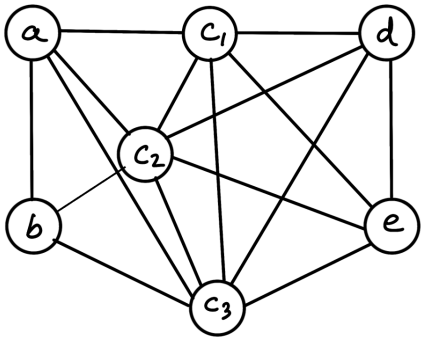

Fig. 5: Two UNLs sharing Three Common Members

Arbitrarily, let us enforce 80% similarity and see whether it preserves consensus correctness. Later we shall discuss how much similarity is indeed needed and discuss the optimal similarity.

E. How much failures can be tolerated with 80% similarity of UNLs with 80% quorum?

Lemma 1: With 80% quorum, RPCA maintains correctness for each transaction with its UNL in an isolated way, as long as $f \leq (n-1)/5$ where f is the number of Byzantine failures and n is the number of nodes in the XRP network.

Proof: Follows from [1].

Lemma 2: With 80% quorum, RPCA will not be able to guarantee that there will be no contradictory transactions across distinct UNLs and its servers, even if the overlap of the UNLs is 80%.

Proof: Consider the structure shown in Figure 6 taken from [4]. Consider the two UNLs L=$\{$1,2,3,4,5$\}$ and L' = $\{$3,4,5,6,7$\}$. It is easy to say they have 60% overlap. As highlighted in [4], node 4 is Byzantine while the others are trusted ones. As node 4 can give contradictory votes to L and L', it can be seen that both of them can get 80% quorum leading to contradictory transactions in L and L'. Hence, the lemma.

Lemma 3: If two UNLs are at least 80% similar then it implies the overlap is at least 89.44%.

Proof: From the RI expression (4) in III-D, we have, *similarity measure* of the two UNLs, is of the form

$$\text{RI} = \binom{m}{2} \div \binom{n}{2} = m(m-1)/n(n-1) \approx m^2/n^2 = (m/n)^2$$

Thus, 80% similarity of UNLs corresponds to saying $(m/n)^2 \geq 80\%$. Hence, m/n is $> 89.44\%$ which can be interpreted as: overlap of the two UNLs is more than 89.44%.

Corollary 1: if L and L' are more than 80% similar, introducing non-common-members to L or L' will lead to reducing below 80% similarity.

Proof: Follows by definition.

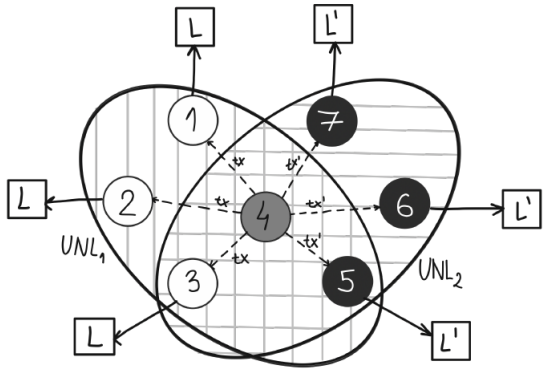

Fig. 6: Safety Violation Example from [4]

Lemma 4: If two UNLs are at least 80% similar, then the UNLs cannot have two contradictory concurrence of transactions assuming at most 20% failures (Byzantine)

Proof: Let L and L' be the two UNLs. Assume the contradictory: L concurs Tx and L' concurs TX' where Tx and Tx' are conflicting transactions. Assuming 20% failure, Tx or TX' would have to be selected by the remaining 69.44% (89.44-20–ignoring for simplicity ceiling functions required to realize integers) of the UNL members that are behaving correctly. As majority forms the basis of selection, it follows that some of the 19.44% processors have to show Byzantine behaviour if both Tx and Tx' have to be selected - contradicting the hypothesis that only 20% can fail.

IV. How much Similarity is essential?

First let us look at existing hints of formation of UNLs as highlighted by the designers.

A. UNL Formation Hints

In [1], it is argued that UNLs is not created randomly in Ripple, but done using the notion fault tolerance to avoid the formation of a nefarious cartels. The relation expressed in [1] in this context is: If p_c is the probability the node will decide

to collude and join a nefarious cartel, then the probability of correctness, p^* is given by

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

p^* represents the likelihood that the size of the nefarious cartel will remain below the maximal threshold of Byzantine failures, for a given p_c. As binomial distribution, followed, values of p_c greater than 20% will result in expected cartels of size greater than 20% of the network - thus short circuiting the consensus process.

Further, the authors in [12] explore the use of tools available in Ripple to arrive at network health monitoring measures. The authors suggest that a large UNL overlap is needed only with more than 20% of malicious nodes in the network.

B. Consensus Correctness Characterization via Similarity

Given that 80% quorum is geared into RPCA and XRPL in general, let us see how much similarity must be enforced on UNLs to preserve safety. Note that the purpose of UNLs based consensus is to enhance the performance and thus, reduction of similarity will lead to higher performance.

In the previous section, we have shown that 80% similarity overcomes conflicting transactions. The question would be, what would be its lower bound.

Lemma 5: If UNLs are at least 50% similar, then the overlap would be $\sqrt{50} = 71.44\%\lceil 71.44\% \rceil = 72\%$ overlap (keeping in view that the number nodes and links have to be integers).

Proof: Follows trivially from the earlier proof given for lemma 3 as we get majority even assuming 20% Byzantine failure occurs in an UNL.

Lemma 6: If two UNLs are at least 50% similar, then the UNLs cannot have two contradictory concurrence of transactions assuming at most 20% failures (Byzantine).

Proof: From Lemma 5, it can be seen that anything larger than 50% similarity corresponds to greater than 71% overlap. Now from the proof of Lemma 4, the result follows.

Reducing similarity below 50% will lead to scenarios wherein majority voting (greater than 50%) cannot be guaranteed; this is modulo integer ceilings of the members. Hence, 50% similarity is optimal.

Theorem 2: RPCA or XRPL will be safe with 80% quorum and a similarity of at least 50% across UNLs.

Proof: Follows from Lemma 5.

Corollary 2: The RI or the similarity of the network shown in Figure 6 is less than 50% with respect to the underlying XRP network of those nodes and thus, does not guarantee non-conflicting transactions.

Proof: Communications in L are given by { (1,4), (2,4),(3,4), (4,5)} and those of L' are given by { (3,4),(4,5),(4,6),(4,7)} and hence RI = $2 \div \binom{7}{2} = 2 \div 21 < 50\%$. From Lemma 4, it follows that it does not guarantee non-conflicting transactions.

Lemma 7: The network can get stuck with XRPL even with more than 90% Overlap[3] and without a node being Byzantine.

Proof: Using Figure 7, the authors of [3] illustrate an example of two UNLs even 90% overlap do not converge on consensus. There are two UNLs, X (in red) = {P1,P2, . . . ,P101} and Y (in blue) = P2,P3,...,P102. Peers 1 51 use X and peers 52 102 use Y . There are two ledgers, L and L. The nodes listening to X all validate a descendant of L, while the nodes listening to Y all validate a descendant of L. Since $51 > 0.5|X|$ nodes in X validate a descendant of L. Thus, according to the preferred branch protocol all, the nodes listening to X cannot switch branch to L. Similarly, since $51 > 0.5|Y|$ nodes in Y all validate a descendant of L, the nodes listening to Y cannot switch branch to L. The network cannot ever rejoin without manual intervention.

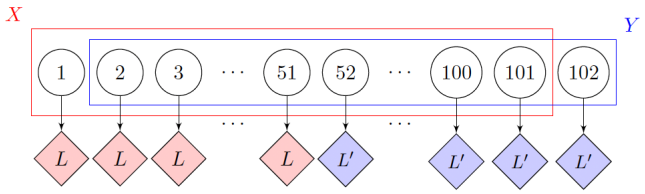

Fig. 7: Stuck-in-Net, 99% Overlap & No Byzantine Node [3]

C. Assessing Loss of communication due to failures in XRP network

In the XRP network, due to failures, communication links between peer-to-peer nodes also fail. Let us try to have at least a qualitative understanding of the quantum of connections that can get lost due to Byzantine failures. Recalling that the similarity of a UNL of n nodes is of the order of $(n)(n-1)/2$ or is proportional to n^2. If we assume 20% failures and assuming that the corresponding connections also fail, the remaining working connections will be proportional to $(16/25) * n^2$ and connections lost will be proportional to $(1/25) * n^2$. A simple estimate is tabulated below assuming clique connection:

n	16/25 * n^2	1/25 * n^2
10	64	4
50	1600	100
75	3600	225

Fig. 8: Density of Working Connections vs Loss of Connections Due to Failure

AS discussed already, 80% similarity will have an overlap of nearly 90% overlap of nodes in UNLs, 70% similarity will have an overlap of 83% overlap of nodes in UNLs, 50% overlap will have an overlap of 71% overlap of nodes in UNLs. Assuming n is reasonably large, the cluster or quasi-cluster will have quite a dense set of communications compared to the original sub-network or the UNL; that is, Byzantine node failure will only remove a small number of connections (communications) and thus, clusters or quasi-clusters will not become disjoint or remain connected. As one can see from

the table that the order of the number of communications lost will be small (very small at lower values of n).

D. Similarity vs Stuck-Free Networks and Liveness

Lemma 8: If the UNLs shown in Figure 7 are 50% similar such a partition cannot take place.

Proof: Let UNL1 = {C11, ... C1m} and UNL2 = {C21, ... C2m}. Assuming 20% byzantine at the worst, there will be overlap of more than 50% honest nodes. As the density of edges will have to maintained to keep up the assumed 50% similarity, ignoring 20% of the byzantine nodes, there will be enough connections (cf.Figure 8) in the network that allows exchange of values without leading to disjoint partitions that arises if one had considered only overlap of nodes.

Lemma 9: There are UNLs with just one Byzantine node leading to liveness violation in spite of 80% quorum.

Proof: The authors of [4] illustrate through an example shown in Figure 9 that depicts a system with 2n correct nodes and one single Byzantine node. All nodes are assumed to trust each other, i.e., there is one common UNL containing all 2n + 1 nodes. Based on Figure, the byzantine node suggests Tx to 1-n and TX' to n+2 to 2n+1. This obviously leads to situation where neither nodes from {1,...,n} nor {n+1, ... 2n+1} switch from Tx to Tx' or Tx' to Tx respectively as per the preferred ledger rule.

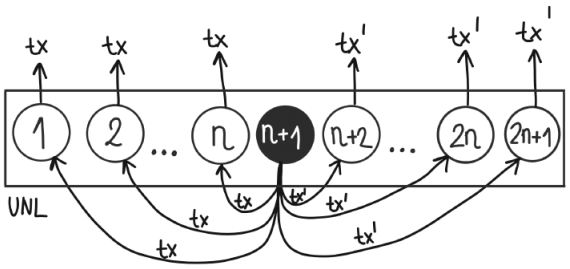

Fig. 9: Single Node Byzantine Creating Liveness Violation [4]

Lemma 10: The UNL shown in Figure 9 has RI = 0 with respect to the XRP peer-to-peer network, assuming the arrows shown are the only possible communications.

Proof: The UNL shown in can be represented as { {(1,n+1), ... , (n,n+1)},{(n+1)}, {(n+1, n+2), ..., (n+1,2n+1)}. It easily follows from the definition of RI, that RI of the network is 0.

Lemma 11: If the UNL is at least 50% similar, then there cannot be liveness violation with at most 20% failures (or byzantine).

Proof: Let the UNL be represented by {C1, C2, ...cm}. A similarity of 50% implies more than $\lceil 71.41\% \rceil$ overlap pairwise. Hence, even if 20% failures occur there will be other nodes that will vote on seeing the other value (cf. Figure 8). As the algorithm is deterministic, one of the Tx will go through and there will be no liveness violation.

V. CONCLUSIONS

In this paper, we have described a similarity measure based on rand index for UNLs, and shown that XRPL maintains consensus correctness with 80% consensus and 50% similarity. The similarity measure binds the UNLs tightly that enables it to maintain consensus correctness. We have also related the similarity measure with overlap of UNLS and shown that the XRPL preserves safety and maintains transaction liveness with 50% similarity under 20% Byzantine failures. If the system must be further failure tolerant of say up to the limit of 1/3 failure of the total number of nodes in the XRP network, the similarity must be enhanced. We have shown how through the enforcement of 50% similarity, XRPL preserves safety and liveness requirements. While several authors had analysed the XRPL protocol for security and correctness, this is the first time, a characterization of consensus correctness has been established using a easy-to-understand metric rand-index. The notion of similarity and correctness established here, perhaps indicates as to how the XRP network has been working robustly in practice.

Whenever, one looks at UNL based consensus, one wonders as to how the UNLs can be formed to enhance performance keeping the correctness invariant. While in section IV-A, we discussed informal hints, we opine that the notion of similarity shall provide a basis on which UNLs can be effectively designed. So far, even with overlap notions, there has been no clear indication of how much performance gain is obtained either in theory or practice through the use of UNL based consensus. It would be interesting to evaluate how much performance can be gained through a rigorous design of UNLs both theoretically and practically.

REFERENCES

[1] D. Schwartz, N. Youngs, and A. Britto, "The ripple protocol consensus algorithm," 2014.

[2] F. Armknecht, G. Karame, A. Mandal, F. Youssef, and E. Zenner, "Ripple: Overview and outlook," vol. 9229, 08 2015.

[3] B. Chase and E. MacBrough, "Analysis of the xrp ledger consensus protocol," 2018.

[4] I. Amores-Sesar, C. Cachin, and J. Mićić, "Security Analysis of Ripple Consensus," in *24th International Conference on Principles of Distributed Systems (OPODIS 2020)*, vol. 184, 2021, pp. 10:1–10:16.

[5] V. Tumas, S. Rivera, D. Magoni, and R. State, "Topology analysis of the xrp ledger," in *Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing*. New York, NY, USA: Association for Computing Machinery, 2023, p. 1277–1284.

[6] L. Mauri, S. Cimato, and E. Damiani, "A formal approach for the analysis of the xrp ledger consensus protocol," in *Proc. 6th, ICISSP 2020, Valletta, Malta*. SCITEPRESS, 25-27 February 2020, pp. 52–53.

[7] Wikipedia, "Cluster analysis," 2023. [Online]. Available: https://en.wikipedia.org/wiki/Cluster_analysis,note=

[8] ——, "Rand index," 2022, [Online; accessed 30-Dec-2022]. [Online]. Available: https://en.wikipedia.org/wiki/Rand_index

[9] S. Roy and R. K. Shyamasundar, "An analysis of hybrid consensus in blockchain protocols for correctness and progress," in *37th Annual IFIP WG 11.3 Conference, DBSec 2023, Sophia-Antipolis, France, July 19–21, 2023*. Springer-Verlag, 2023, p. 404–412.

[10] V. Buterin and V. Griffith, "Casper the friendly finality gadget," https://arxiv.org/abs/1710.09437,October2017,.

[11] Y. Sompolinsky and A. Zohar, "Secure high-rate transaction processing in bitcoin," pp. 507–527, 2015.

[12] K. Christodoulou, E. Iosif, A. Inglezakis, and M. Themistocleous, "Consensus crash testing: Exploring ripple's decentralization degree in adversarial environments."