To Squelch or not to Squelch: Enabling Improved Message Dissemination on the XRP Ledger

Lucian Trestioreanu*, Flaviene Scheidt*, Wazen Shbair*, Jerome Francois*, Damien Magoni†, and Radu State*
* University of Luxembourg, SnT, 29, Avenue J.F Kennedy, L-1855 Luxembourg
Email:{lucian.trestioreanu, flaviene.scheidt, wazen.shbair, jerome.francois, radu.state}@uni.lu
† University of Bordeaux, LaBRI - CNRS, 351 Cours de la Liberation, F-33405 Talence, France
Email: damien.magoni@u-bordeaux.fr

***Abstract*—With the large increase in the adoption of blockchain technologies, their underlying peer-to-peer networks must also scale with the demand. In this context, previous works highlighted the importance of ensuring efficient and resilient communication for the underlying consensus and replication mechanisms. However, they were mainly focused on mainstream Distributed Ledger Technologies like Bitcoin or Ethereum, and generally, Proof-of-Work-based designs.**

In this paper, the problem is investigated in the context of consensus-validation based blockchains, like the XRP Ledger. The latter relies on a Federated Byzantine Agreement (FBA) consensus mechanism which is proven to have a good scalability in regards to the transaction throughput. However, it is known that the size of the network constituting the XRP Ledger is limited, and significant increases would be challenging to achieve. The main reason is the simple flooding mechanism used to disseminate the messages related to the consensus protocol, which creates many duplicates in the network. Squelching is a recent solution proposed for limiting this duplication, however, it was never evaluated quantitatively in real-life scenarios involving the XRPL production network. In this paper, our aim is to assess this mechanism using a real-life controllable testbed and the XRPL production network, to assess its benefit and compare it to alternative solutions relying on Named Data Networking and on a gossip-based approach.

***Index Terms*—Performance, Efficiency, XRP Ledger, Overlay, Networks, communication, blockchain, named data networking**

I. Introduction

The Distributed Ledger Technology (DLT) is relatively new and still evolving. Its development has been fostered by a diverse and enthusiastic community that sometimes forgot about lessons from the past related to efficiency, resilience, and security of communications.

As a result, many blockchains rely on the flooding mechanism as a straightforward solution for addressing the one-to-many and many-to-many communication specifics of DLT. This undoubtedly leads to limitations in blockchain scalability.

More specifically in the context of blockchains, scalability generally concerns either being able to add more nodes to the network (*Node*-wise scalability), or the ability of the blockchain to process more transactions per second (TPS) (*Throughput*-wise scalability).

Throughput-wise scalability can be achieved on-chain (changing code, e.g. sharding), off-chain (payment channels, sidechains), or otherwise - for example, specific mechanisms like the XRP Ledger (XRPL) consensus. Unlike Proof-of-Work (PoW) and Proof-of-Stake-based blockchains which rely on computational power or stakes respectively, XRPL is a Byzantine Fault Tolerant (BFT)-based blockchain which uses votes of groups of trusted validators to achieve consensus.

Generally, PoW blockchains can easily increase their number of nodes, but face challenges to increase their throughput. In contrast, Byzantine Fault Tolerant (BFT)-based blockchains offer higher throughput but can not easily scale with respect to the number of nodes [1]. For instance, XRPL can achieve 1500 TPS (with 3000 TPS expected soon [2], [3]), but relies on a peer-to-peer (P2P) flooding mechanism leading to a multitude of redundant messages disseminated through the network. With a network size of around 1000 nodes, this already has a negative impact on scalability [4].

Previous works highlighted how the latency and bandwidth of the underlying communications (from physical channels to protocols) can limit the scalability of blockchain networks [5], [6]. In this perspective, the community effort was mainly focused on mainstream blockchains like Ethereum (ETH) [7], [8] or Bitcoin (BTC), both being Proof-of-Work type (PoW); recently, ETH switched to *Proof-of-Stake*. For example, projects like *Fibre*[1], *Falcon* [9] or *bloXroute* [10], aimed to improve BTC transaction rate by speeding up block propagation. *Node*-wise scalability can be achieved, for example, by efficient message transmission options, but was not well explored in the case of consensus-validation based blockchains [1].

Therefore, this paper focuses on the *Node*-wise scalability of BFT-based blockchains taking XRPL as an illustrative example of our analysis.

To mitigate the overhead of message flooding, *Squelching* is a recently proposed dissemination protocol, that decreases the number of messages on the XRPL P2P network by reducing the number of duplicates. The main principle resides in a careful selection of peers from which to receive messages, rather than receiving messages from all possible peers.

In this paper, our objective is to evaluate to what extent *Squelching* can improve the performance of intra-ledger communication on XRPL, (*i.e.* decrease the total number of *validation* and *proposal*-related messages), thus reducing

[1]https://bitcoinfibre.org/, valid in October 2023

the computational overhead of XRPL nodes, and ultimately improving the XRPL *Node*-wise scalability.

For evaluation purposes, two types of experiments are performed: (1) baseline experiments on the actual XRPL P2P network to measure the impact of the flooding mechanism on the computational overhead of the node (without Squelching) and (2) experiments in a fully controlled distributed environment to evaluate the benefit of Squelching. Together, these two experiments allow us to assess how Squelching can improve the P2P connectivity.

This paper contributes to filling a research gap regarding the efficiency of the underlying communications which support the consensus protocols of BFT-based blockchains like the XRP Ledger. Our contributions are four-fold:

1) Highlight how the current flooding mechanism of XRPL is a factor contributing to limit its *Node*-wise scalability.
2) Define a measurement method to evaluate the impact of Squelching in regards to a baseline version.
3) Apply the aforementioned method to perform a quantitative assessment of Squelching.
4) Discuss the results in the context of other proposed alternatives, namely XRP-NDN [11] and GossipSub [12].

The structure of the paper is as follows: Section II introduces background on XRPL and refines the problem. Section III presents *Squelching*. The evaluation method and obtained results are shown in Section IV. In Section V, alternative solutions are discussed while Section VI provides a broader overview of related work. Section VII concludes the paper.

II. Problem Refinement

A. Background on the XRP Ledger

Leslie Lamport previously showed that, in a synchronous environment, a consensus can be achieved if at most n out of $3n + 1$ parties involved are dishonest [13]. The *Practical Byzantine Fault Tolerance* [14] algorithm makes the latter practicable in an asynchronous environment such as the Internet, paving the path to consensus-validation based DLTs.

The XRPL specifically implements a variant of the BFT consensus called Federated Byzantine Agreement (FBA) Consensus [15], which is named the *XRP Ledger Consensus Protocol* (XRP LCP). It improves transaction (TX) throughput while maintaining security against Byzantine failures.

Although the baseline algorithm supposes all nodes to agree on the list of participants and to process all transactions to reach a consensus, FBA introduces the concept of quorum slices: a node only needs to trust a subset of the other nodes, to take its own decision about the TX to be validated. Because nodes can decide themselves which other nodes to trust, multiple quorum slices can form in the network. For a healthy network, the quorum slices must overlap to some degree, as shown in Figure 1. This means some nodes are trusted in multiple quorum slices, thus ensuring the dissemination of information. The slices must reach a decision among themselves, and after this, a final decision can be taken.

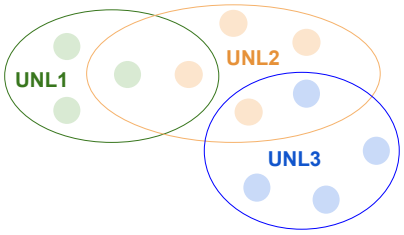


Fig. 1: FBA quorum slices.

In XRPL, there are two main types of nodes: *tracker* nodes in charge of processing transactions and *validator* nodes used for consensus voting. Each voter (validator) selects a Unique Node List (UNL) of *trusted* voters to collaborate with, meaning they are *trusted* not to collude among themselves to defraud the respective validator. Figure 2 illustrates the ledger creation on XRPL. A new ledger is created from the previous one by applying a new set of transactions. This process involves three main phases:

1. Transaction submission: New transactions can be submitted through flooding at any time, but for a certain ledger, there is a time window in which the new transactions can be accepted. The network waits for a certain time for new *transactions* to be included in the current ledger: during this time, new transactions submitted from the tracker nodes T1, T2, T3, are flooded as *transaction* messages through the XRPL network and will be received multiple times by the validators V1, V2, V3. New transactions that missed the time window are stored and will be processed in the next ledger. While validators can technically also submit transactions, this is a discouraged practice for security and performance reasons.

2. Consensus: During multiple consensus rounds, validators exchange *proposal* messages to agree on the transaction set to be included. Due to the flooding mechanism that is used, the network nodes produce and propagate duplicate messages.

3. Validation: Validators may produce inconsistent ledger versions for the same ledger index. Therefore, in this phase, validators exchange *validation* messages to agree on the next ledger to be created from all candidates for the given index.

B. XRPL topology

XRPL uses a flooding mechanism for the dissemination of transaction, proposal, and validation messages. It is an effective solution to explore every path and as a result, to reach every node. It is however inefficient because it also forwards a significant number of duplicates, but this also depends on the underlying topology.

As a preliminary study, we scanned and analyzed XRPL's structure as of 2021 with Nem [16]. Results are summarised in Table I. The network consisted of 892 nodes of which 152 were identified as *validators*. The average distance between a pair of nodes was 2.37. The *topology diameter* was 5. Between these nodes, a giant bi-connected component of 867 nodes was identified, together with 25 smaller ones. The giant component was composed of 9172 edges and the mean degree of the nodes

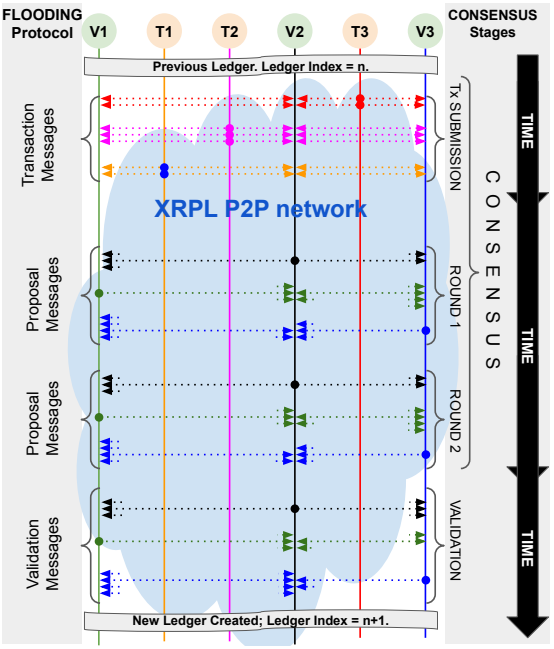


Fig. 2: The Ledger building process on XRPL.

was 21, while the maximum degree was 296, with multiple similarly high-degree nodes, which means that some nodes act as hubs. The *mean distance* was 2.34, the *median distance* 2.37, with a *diameter* of 4. An in-depth analysis of XRPL can also be found in [17].

Thus, on the one hand, the structure of the XRPL network is dense, highly connected, and with a low diameter. On the other hand, it is known that a dense network is more resilient to node and edge failures but its communication performance is then affected by the high number of duplicates generated if a flooding protocol is used as the number of messages is directly linked to the number of edges [18]. It is worth noticing that each message including duplicates, will go through multiple processing steps including parsing, signature verification, etc.

Hence, the XRPL topology emphasizes the unfavorable impact of the flooding protocol used to relay many different message types involved in the voting-based consensus. In fact, the same type of message might even be sent multiple times to create a single ledger. For example, during the XRPL consensus, there are multiple consensus rounds to create a single ledger, which all involve repeatedly sending proposal messages. The XRPL network was composed of approximately 900 nodes, which is already a relatively high number for this type of blockchain, and it still continues to slowly increase over time.

C. Objective

Although XRPL is able to achieve a *throughput* of 1500 TPS, adding more nodes to the network can be challenging due to the inefficient message flooding used and the dense structure of its network as highlighted above. Therefore, it suffers from a lack of *Node*-wise scalability. Also, it was previously shown in [4] that the number of XRPL *proposal* and *validation* messages represents 72% of all messages, so it is worth optimizing them.

Squelching was recently proposed to enhance the scalability of XRPL by improving communication efficiency when disseminating these particular messages. However, no in-depth assessments were performed. In this paper, our goal is to benchmark this solution, evaluate its potential benefit and compare the achieved results with previous assessments of other solutions based on NDN [11] and Gossipsub [12].

III. Squelching

A. Overview

The *Squelching* protocol has been designed to optimize the relaying of messages within the XRPL network. Its primary goal is to reduce the bandwidth consumption, node CPU usage, and memory load, and therefore to reduce latency and impove *Node*-wise scalability.

Assuming a given validator, each node selects a subset of its peers to be the peers chosen to relay messages created and flooded by the given remote validator. In parallel, it sends a *squelch* message to the rest of its peers to *squelch* the connection for a given amount of time, i.e., stop relaying messages to it. Therefore, it is an active solution where a node assumes other compliant nodes to behave accordingly to the squelching requests they receive. By reducing the number of relaying connections (network edges) involved in the flooding process, and in addition by selecting lower latency connections, this protocol reduces the number of exchanged messages and so, the load on the network and on the processing hosts. As a result, performance is expected to increase on multiple facets (lower CPU and bandwidth usage, or lowered latency). We aim at quantitatively evaluating the achievable performance gain on the hosts in production.

B. Example

Figure 3 is an illustrative example of the squelching protocol phases [4] where a node alternates between phase 2 and 3 once the initialization is done (phase 1):

- Phase 1:
  1) Remote *validator V* creates and floods validation and proposal messages on the XRPL network, which reach *P1-P5* (either trackers or validators) via possibly different routes (dotted arrows).
  2) Nodes *P1-P5* are direct peers (next-hop connections) of node *N* and relay the messages from *validator V* to node *N* (black arrows).
  3) As a result, the node *N* receives each message five times. In this example, it determines that messages from nodes *P2*, *P3*, and *P4* arrive faster. The latency of the connections from these peers is thus lower. The node *N* selects these peers to relay the messages from the *validator V*, and sends *squelch* messages to peers *P1* and *P5* (dashed red arrows).
- Phase 2: *P1* and *P5* continue to receive messages from *validator V*, but only *P2*, *P3*, *P4* relay them to node *N*.

TABLE I: Topology analysis of the XRPL production network.

Metric	nodes	edges	validators	diameter	radius	avg_dist.	med_dist.	avg_deg.	max_deg.
Full XRPL	892	9197	158	5	3	2.37	2.4	20.62	297
Giant component	867	9172	-	4	3	2.34	2.37	21.15	296

- Phase 3:
  1) After some time, *P1* and *P5* un-squelch themselves and start relaying again messages from validator V to node *N*.
  2) Node *N* determines now that peers *P1*, *P2*, and *P4* are better candidates, and sends *squelch* messages to nodes *P3* and *P5*.

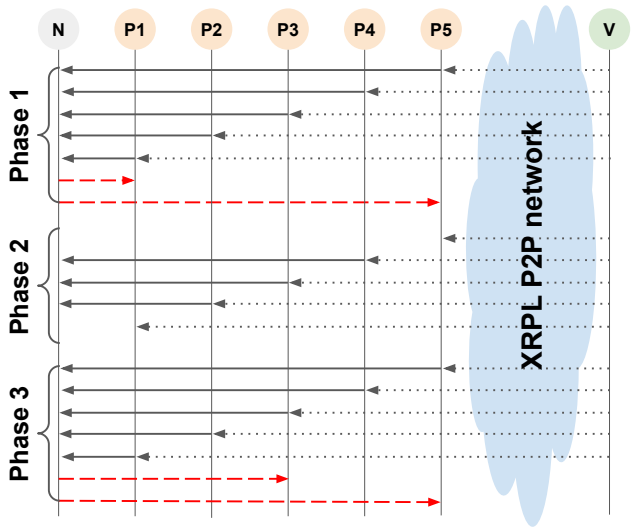


Fig. 3: Squelching sequence diagram.

C. Slot management

The *Squelching* protocol introduces the concept of *slots* and *squelches* to manage the relay of messages among XRP nodes.

For the sake of clarity in the following explanation, *uplinks* are the direct peers from which a given XRPL node receives messages, while *downlinks* are the direct peers to which this given node sends messages to. Thus, this given node is relaying messages from its *uplinks* to its *downlinks*.

Each node in the XRPL network (tracker or validator) creates a *slot* for each validator it chooses to relay messages for. For each slot (validator), a list of uplinks and downlinks is maintained. When an XRPL node (tracker or validator) receives a proposal or *validation* message from one of its directly connected peers (uplinks), it extracts the originating validator. The node can thus check the corresponding slot, and if enough copies have been already received, a *squelch* message is sent back to the peer that just relayed the respective message. This message requests the respective peer to stop forwarding further messages created by the respective originating validator.When receiving such a *squelch* message, a peer removes the sender of that message from the *downlinks* for the corresponding slot (*i.e.*, originating validator). Therefore, each node keeps track of its downlinks towards which the messages must be relayed for every unique validator.

Squelches have a time limit, allowing for network topology changes. After a certain period, the squelch expires, and the respective peer is eligible to be considered again by the node as a relay for the given originating validator.

For reliability purposes, if a node loses an *uplink*, it can replace it by sending *unsquelch* messages to other peers. This allows to maintain enough active connected peers for ensuring the consensus protocol security.

D. Implementation

The protocol introduces a new XRPL message type for instructing peers to *squelch* or *unsquelch* messages from specific validators. Feature negotiation is done via HTTP headers to determine whether a node supports the feature. So, the protocol can be implemented without the need for a network-wide upgrade. Obviously, if a node is not compatible, it cannot interpret the *squelch messages* and continues forwarding all messages. Hence, consensus can still be achieved. While the number of peers to squelch could be a configurable parameter, XRPL software sets this number to five.

IV. Evaluation

A. Method and metrics

In this section, the main objective is to measure the impact of squelching in comparison with a baseline implementation without squelching, i.e., using flooding.

After a set of initial experiments, we observed that mostly CPU usage is significantly impacted by the number of messages to handle. The first set of experiments is focused on the CPU usage by an XRPL node without squelching but with a varying number of peers, since the number of flooded messages received directly depends on the latter. So, on a node connected to the main XRPL network, we measured as a function of the number of peers: average CPU usage, the number of messages received, and number of messages sent.

From this first set of measurements, we are able to quantify the CPU overhead and messages to be processed due to an increased number of peers, and infer a regression model.

In the second set of measurements, squelching is applied and its impact is monitored in terms of number of messages sent and received. Thanks to the regression modeling, we will be then able to extrapolate the number of messages saved in the Mainnet XRPL network and ultimately the number of free slots for additional peers to connect to. In order to control the experimental parameters, we deploy a small-scale XRPL network on a controlled and configurable HPC facility.

The baseline used for the evaluation was an unmodified version of XRPL, which was compared with a modified version[2] implementing the Squelching mechanism:

- Baseline evaluation (using XRPL v1.6).
- Squelching evaluation (using XRPL v1.7).

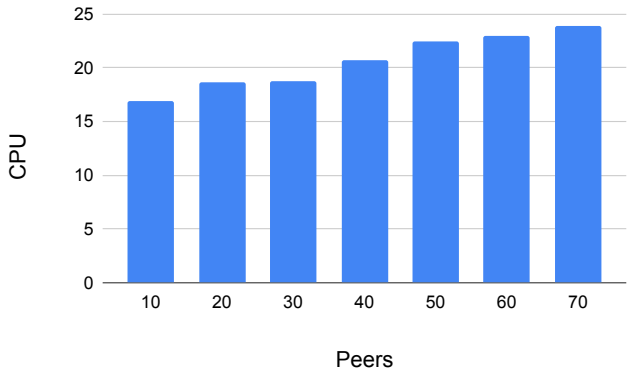


Fig. 4: Mean CPU usage according to the number of peers.

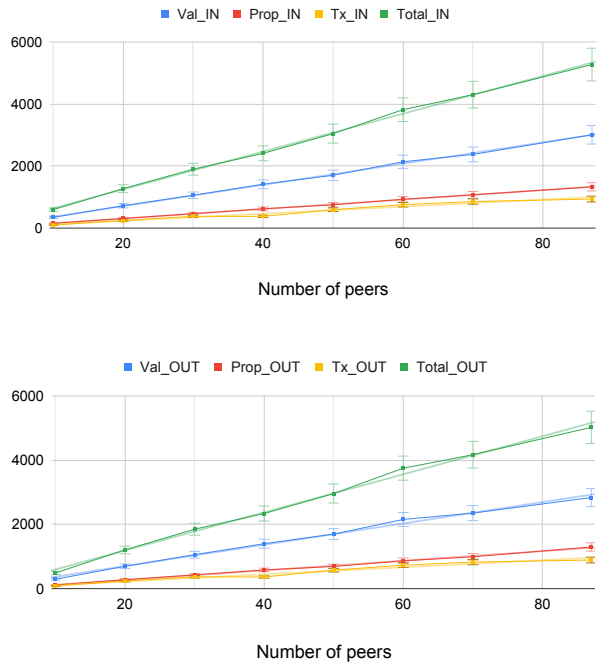


Fig. 5: Number of messages IN and OUT, per second and per type (Val: validation, Prop: proposal, Tx: transaction).

B. Baseline experiments

To perform the baseline experiments, a single node is connected to the actual XRPL network. Our node is run on an Ubuntu 22.04 with 16Gb RAM and 4 Intel Xeon E5-4650 v4 @2.2GHz. The statistics are gathered thanks to the RippledMon tool [3].

[2]https://github.com/XRPLF/rippled/pull/3412, valid in October 2023
[3]https://github.com/ripple/rippledmon, valid in October 2023

We configured the node to connect to an increasing number of peers over time, and to collect the average CPU usage alongside the number of messages sent and received. The results are given in Figures 4 and 5 respectively. We notice a linear dependency between the number of peers and the number of messages. As expected, the total number of messages received and sent increases with the number of peers, as the flooding mechanism is exploiting the new connections to send redundant messages. There are no significant differences between the direction of messages, since for each new message received, our node also sends duplicatas. Due to the BFT-based validation mechanism, the number of exchanged messages varies according to the type of messages with the *validation* messages being the most frequent. From these observations, Squelching could be fine-tuned to limit the flooding only for this type of message. This will still increase *Node*-wise scalability while enforcing a higher robustness for other types of messages.

Due to a higher number of messages to be processed (parsing, signature check, etc) when the number of peers increases, the CPU usage also increases gradually in Figure 4 with an increase of approximately 35% from 10 to 70 peers.

C. Performance model

To assess the *Node*-wise scalability, we need to model the relationships between the resources used in terms of CPU usage, messages processed, and the number of neighbors or peers of an XRPL node. This will allow us to quantify how many resources are used by a single peer.

Thanks to the linear dependency, a simple regression allows us to model the CPU use (cpu) with respect to the number of peers the node is connected to ($peers$):

$$\begin{aligned} cpu &= \beta_0 + \beta_1 peers \\ with\ \beta_0 &= 15.8754, \beta_1 = 0.1177 \end{aligned} \tag{1}$$

It is worth noticing that messages are also considered independently of their types. As a black-box measurement type, our approach cannot distinguish the CPU use by each of them. Similarly, the total number of messages ($msgs$) is linearly dependent of the number of peers ($peers$):

$$\begin{aligned} msgs &= \alpha_0 + \alpha_1 peers \\ with\ \alpha_0 &= -75.0943, \alpha_1 = 123.6365 \end{aligned} \tag{2}$$

Because of a good linearity of the underlying data, the regression models fit with a high coefficient of determination, $R^2 > 0.96$ for both equations (1) and (2).

D. Experimental setup for squelching

To assess Squelching, we deployed our own XRPL network to fully control the nodes. In order to perform the measurement in real-life conditions where nodes are spread among different geographical locations, a testbed hosted on Grid'5000 (G5K) was built. G5K[4] [19] is a large-scale HPC platform with interconnected sites in France and Luxembourg. It features

[4]https://www.grid5000.fr/w/Grid5000:Home, valid in October 2023

TABLE II: Squelch versus Flooding results.

FLOOD - average of total messages / second	297.633
SQUELCH - average of total messages / second	211.602
SQUELCH / FLOOD (%)	71.094
SQUELCHING saves over FLOOD (%)	28.905

a large amount of resources: 15000 cores, 800 compute nodes with bare-metal access, and 10Gbps Ethernet links.

To deploy the XRPL network on G5K and perform the evaluation, we used our previous work, the BlockZoom Tool [20] (available on Github[5]), offering a reproducible environment for experimentation with DLT and smart contracts.

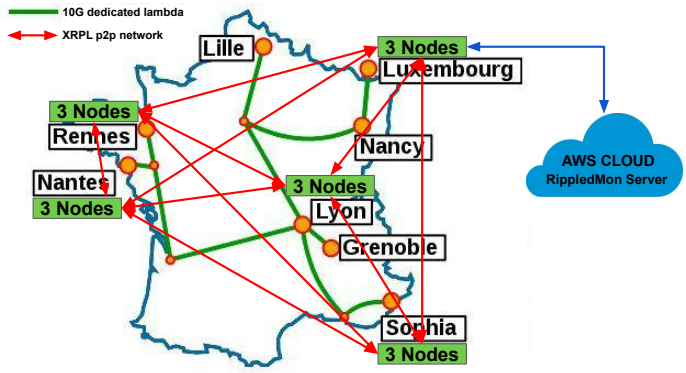


Fig. 6: XRP network monitoring testbed deployed on G5K.

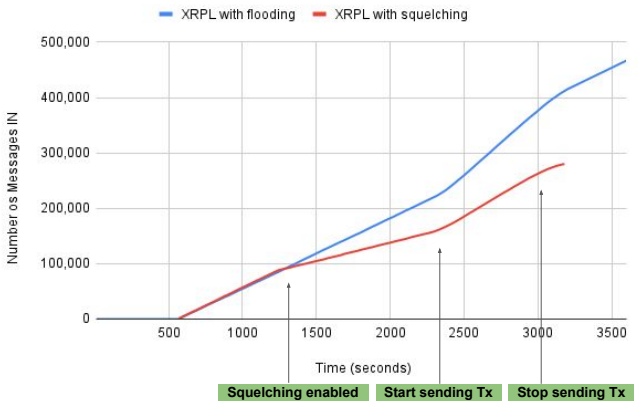
The testbed was composed of 15 XRPL nodes evenly spread over five *G5K* site locations, meaning three nodes per site, to simulate as much as possible the real-life conditions where XRPL nodes are spread over different locations, with different latencies. Because latencies impact message propagation performance, we prefer this setup instead of others with limited latency. For instance, while latencies can be added in other ways, e.g., using Mininet, this is still not realistic. Furthermore, Mininet also comes with its own limitations and challenges, for example, the requirements to run many XRPL nodes on the same machine. On the other hand, G5K is firstly a research-oriented platform providing monitoring tools to access precisely the performance of a running algorithm/software and hardware on the platform. Secondly, G5K supports experiment reproducibility which is crucial for scientific analysis.

The consensus quorum was set to six (in real life not all nodes on the production XRPL network are validators). The nodes boot with no transactions being generated until a predefined time. Once this period of time expires, a number of 1000 transactions per site was sent in parallel.

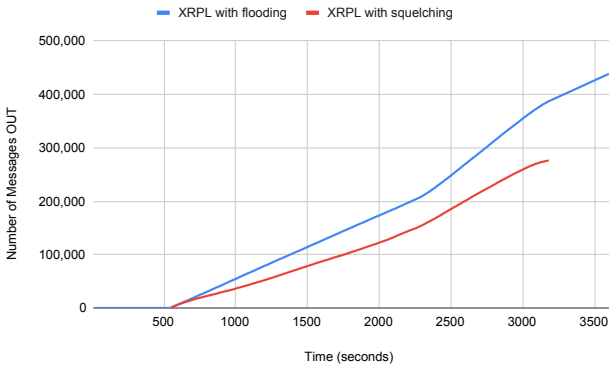
E. Squelching results

The comparative results are presented in Figure 7 and Table II. Figure 7(a) is annotated with the stages of the experiment for the sake of clarity, *i.e.* when squelching is enabled and when the node starts or stops sending *transaction* messages.

[5]https://github.com/wshbair/BlockZoom, valid in October 2023



(a) Total number of Messages IN



(b) Total number of Messages OUT

Fig. 7: Cumulative number of messages: unmodified XRPL *vs* XRPL with squelching.

As they are cumulative functions, the difference in the number of messages between flooding and squelching increases over time. The same observation applies to received messages at a slightly lower scale. When the transactions start to be sent, the number of messages logically increases and the observed differences are still valid.

From Table II we see that *Squelching* saves 28.9% messages per second on average over *Flooding*. It must be noted that on the G5K experiments, at 15 peers, the number of messages is much lower than its 15-peers equivalent on Mainnet (Figure 5). This is due to the much larger scale of Mainnet, which generates proportionally more messages. However, this is not an impediment, because what we are interested in is the last row in Table II, *i.e.*, the average savings offered by Squelching *vs* Flooding (%). The latter can be translated directly to the Mainnet experiment thanks to our regression models.

Table III describes how we computed the potential gain of an XRPL hub node with 200 peers on Mainnet if *Squelching* would be applied. The baseline measurements are reported as points #1 to #7, reminding that they were measured experimentally on Mainnet. Point #9 (black squares in Figure 8) is obtained by the linear regression models of equations (1) and

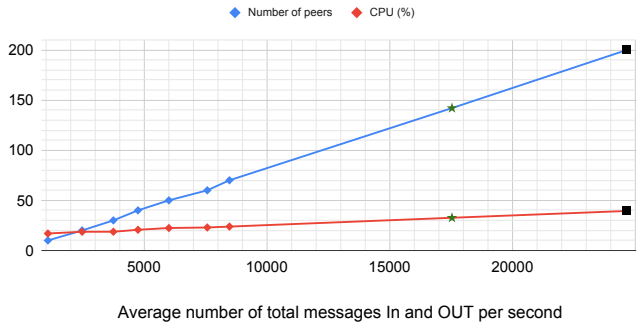


Fig. 8: Extrapolation of *number of peers* and *CPU(%)*.

TABLE III: Gains obtained through Squelching.

#Point	Total messages	Peers	CPU (%)	How obtained
1	1080.92	10	16.929	Experimental
2	2474.27	20	18.651	Experimental
3	3744.87	30	18.694	Experimental
4	4747.98	40	20.645	Experimental
5	6005.29	50	22.405	Experimental
6	7568.48	60	22.924	Experimental
7	8470.71	70	23.825	Experimental
8	17527	142	32.620	Regression
9	24652	200	39.407	Regression

(2), by rearranging the terms as follows:

$$\begin{aligned} CPU &= 0.1177 * Peers + 15.8754 \\ Peers &= 0.0081 * Messages + 0.6074 \end{aligned} \tag{3}$$

Then, knowing that squelching could reduce the amount of messages by 28.9%, we compute the number of peers and CPU for 28.9% less messages than Point #9, i.e., for 17527 messages. For this value, we compute the corresponding number of peers and CPU (Point #8, green stars in Figure 8) with our performance model (equations (1) and (2) again).

The result shows that a node connected to XRPL Mainnet network with 200 peers could save 17% CPU with *Squelching* and free up 58 peer slots. This would actually enhance the overall connectivity as a single node could have 29% additional peers to connect to.

V. Discussion and Future Work

While Squelching is one solution to improve the dissemination of messages, other alternatives exist for XRPL and are discussed hereafter.

A. Gossipsub

Gossipsub [7] is a publish-subscribe [21] protocol designed to enable efficient and scalable message dissemination in P2P networks. Concisely, the Gossipsub nodes engage in a gossip-type protocol by selecting a sub-set of peers to share messages with, which then propagate them again in a recursive manner. There can be different "topics of discussion" between the gossipping nodes, for which different sets of peers are used at a certain moment. The peer selection algorithm is complex and makes use of an extensive set of criteria and parameters.

Gossipsub was originally proposed for Ethereum and Filecoin but we previously proposed an adaption for XRPL named FlexiPipe [12]. To facilitate the discussion, we partially reproduce a relevant result from [12] in Figure 9, where this solution was compared to Squelching and to the original flooding mechanism. It shows that Gossipsub is able to properly disseminate the validations with less duplication. However, this evaluation does not consider the overhead of messages generated to maintain the Gossipsub overlay network.

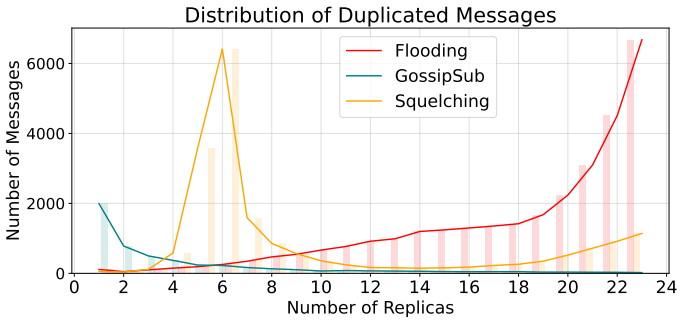


Fig. 9: Frequency of duplicated messages on XRPL with Flooding vs Squelching vs Gossipsub [12].

Also, compared to Squelching, Gossipsub relies on many parameters to be fine-tuned, and most of all, it assumes the deployment of a distinct overlay next to the native XRPL overlay network. Actually, the main advantage of Squelching is to directly work using connections of the XRPL overlay itself. Its deployment is thus simplified and does not introduce possible external threats or failures from external overlays.

B. XRP-NDN

Instead of sending data packets to specific locations (*e.g.* IP addresses), Named Data Networking (NDN) [22]–[24] allows the users to retrieve content by expressing what they want, similar to requesting a book by its title, by analogy. In brief, a user sends an interest packet with the data name. The forwarding is based on name prefixes instead of IP prefixes. The request is thus forwarded to reach one or multiple content providers or in-network caches with the expected content. The data is then sent on the backward paths. With such mechanisms, data is cached in a distributed manner among network nodes.

In [11] we showed how the blockchain consensus messaging can be ported to use NDN for message propagation by proposing multiple mapping models for the forwarding of messages, and investigated the advantages and disadvantages of each model according to the specifics of XRPL. Even if straightforward comparisons with squelching would not be fair due to the different nature of networks (*e.g.* caching mechanisms which imply a distributed overhead in the network), performances reported in [11] are good for an NDN-based solution. Again, this requires the creation and maintenance of an additional overlay network like for Gossipsub. Besides, the size of the NDN interest message payload is limited making the solution not viable for PoW-type blockchains.

Because in NDN each piece of data is named, a further exploitable advantage is the use of a single *validation* message per validator entering each XRPL node. This discrimination can be done by filtering messages at the NDN overlay level such that they do not reach the XRPL node application, which can even use different hardware to take more advantage from message filtering. The NDN solution leaves open the possibility to use a structured NDN overlay, or alternatively, an unstructured mesh mirroring the XRPL mesh. Naturally, a structured overlay could diminish the total number of messages on the NDN overlay at the expense of security and robustness, while an overlay mirroring XRPL would achieve the opposite.

C. Future work

As highlighted above, solutions based on more structured overlay networks like Gossipsub or with complete changes in the message forwarding like NDN could offer large benefits to the BFT-based blockchains. The main issue is the creation of a side overlay network, with its own potential problems regarding failures, threats, or performance degradation. This is why the DLT community and businesses are quite reluctant to this kind of design choices. Rather than multiplying different overlay networks for a single application, it would be beneficial to integrate these protocols or communication architectures within the DLT protocols in an intrinsic manner. Obviously, this would require an in-depth security analysis because this might directly impact the core functionalities of a blockchain such as the consensus mechanism and its intrinsic security properties.

VI. Related Work

The XRP LCP was described in 2014 [3] followed by a rich literature focused on investigating its robustness and security. This contrasts with our work focused on the message dissemination mechanism. The XRP LCP was analyzed in [2], [25], [26] and investigated empirically in [27]. In 2020, relatively simple cases were identified where consensus may violate safety and/or liveness [28], and it was argued that XRPL needs a very close *synchronization*, *interconnection*, and *fault-free operation* between validators. These findings are another argument for considering also the efficiency and resilience of communication, which is the focus of our research. A man-in-the-middle attack was demonstrated in [29].

Besides, XRPL-related work also investigates the energy consumption of the validators [30], the crypto-asset network flows [31], a health assessment of XRPL credit network [32], a proposal for a blockchain benchmarking framework [33], or a topology analysis [17].

While communication lacked in-depth exploration on XRPL and on consensus-validation-based blockchains in general, an efficient transaction relay for BTC named *Erlay* was proposed in [34]. It relies on two fundamental system-building strategies: delay and batching. Rather than disseminating each TX across every network link, a node selectively broadcasts it to a subset of its peers. TXs are universally propagated among well-connected public nodes through at most 8 outbound connections. Nodes regularly compare their states with their peers to consolidate them and only forward relevant information. While *Erlay* reduces bandwidth by 84%, the latency of TX dissemination increases from 3.15 to 5.75s on average, which is unacceptable on XRPL. *Perigee* [18] is an efficient P2P network design for PoW blockchains focusing on mitigating the block propagation delay, but not on the message flooding issue. Epidemic Broadcast Trees are proposed on a gossip-based overlay in [35] while *Splitstream* [36] distributes the load of forwarding messages evenly between participants. Actually, *Gossipsub* [7], previously introduced, draws upon the general concepts of epidemic broadcast and gossip protocols to achieve its objectives.

While *XRP-NDN* was also reviewed here, there are other proposals intending to use NDN in the context of blockchains, however focusing on PoW type: *BoNDN* [37], proposes TX dissemination for BTC through a push model over NDN interests, and a subscribe-push model for block propagation. Another design and implementation for propagating the TXs and blocks over NDN for PoW blockchains were proposed in [8] for ETH, while [38] sends blocks over a multi-layer design based on NDN to achieve 74% of BlockNDN's overhead [39]. Being content-oriented, the NDN architecture is suited for data synchronization, which resulted in the development of different data sync protocols, like *Vectorsync* [40], *Chronosync* [41], or *Psync* [42]. These are good candidates to be leveraged in a blockchain context where states across multiple nodes must be consolidated, and *BlockNDN* actually relies on *Chronosync*. However, originally, they were not meant for the byzantine blockchain environment [8], and on XRPL where we want to minimize the number of messages, the additional sync messages can add unwanted overhead.

VII. Conclusion

In this paper, we formulated the problem of *Node*-wise scalability in the context of XRPL to highlight the rationale behind the Squelching approach. Our empirical approach was first focused on measuring and modeling the node overhead in regard to the number of messages disseminated in the XRPL P2P network. As they represent the vast majority of messages, squelching *validation* and *proposal* messages, as proposed in the XRPL node software, is justified. We have thus evaluated the gain of *Squelching* by using a regression model to understand its impact on the production XRPL network, from measurements done in a controllable environment. Our results show that an XRPL hub node can potentially benefit from a 29% connectivity increase when using *Squelching*. As highlighted in the section before, Squelching might impact the robustness and security of the consensus mechanism itself. Future work will thus consider a thorough security evaluation.

References

[1] M. Vukolić, "The quest for scalable blockchain fabric: Proof-of-work vs. bft replication," in *Open Problems in Network Security*, J. Camenisch and D. Kesdoğan, Eds. Cham: Springer International Publishing, 2016, pp. 112–125.

[2] B. Chase and E. MacBrough, "Analysis of the XRP ledger consensus protocol," *CoRR*, vol. abs/1802.07242, 2018. [Online]. Available: http://arxiv.org/abs/1802.07242

[3] D. Schwartz, N. Youngs, and A. Britto, "The ripple protocol consensus algorithm," 2014, accessed: October 2023. [Online]. Available: https://ripple.com/files/ripple_consensus_whitepaper.pdf

[4] G. Tsipenyuk and N. D. Bougalis, "Message routing optimizations, pt. 1: Proposal & validation relaying," 2021. [Online]. Available: https://xrpl.org/blog/2021/ message-routing-optimizations-pt-1-proposal-validation-relaying.html

[5] M. Kuzlu, M. Pipattanasomporn, L. Gurses, and S. Rahman, "Performance analysis of a hyperledger fabric blockchain framework: Throughput, latency and scalability," in *2019 IEEE International Conference on Blockchain (Blockchain)*, 2019, pp. 536–540.

[6] P. W. Eklund and R. Beck, "Factors that impact blockchain scalability," in *Proceedings of the 11th International Conference on Management of Digital EcoSystems*, ser. MEDES '19. New York, NY, USA: Association for Computing Machinery, 2020, p. 126–133. [Online]. Available: https://doi.org/10.1145/3297662.3365818

[7] D. Vyzovitis, Y. Napora, D. McCormick, D. Dias, and Y. Psaras, "Gossipsub: Attack-resilient message propagation in the filecoin and eth2.0 networks," https://arxiv.org/abs/2007.02754, 07 2020.

[8] Q. T. Thai, N. Ko, S. H. Byun, and S.-M. Kim, "Design and implementation of ndn-based ethereum blockchain," *Journal of Network and Computer Applications*, vol. 200, p. 103329, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804521003143

[9] A. E. Gencer, S. Basu, I. Eyal, R. van Renesse, and E. G. Sirer, *Decentralization in Bitcoin and Ethereum Networks*. Springer Berlin Heidelberg, 2018.

[10] U. Klarman, S. Basu, A. Kuzmanovic, and E. G. Sirer, "bloXroute: A scalable trustless blockchain distribution network WHITEPAPER," in *IEEE Internet of Things Journal*, 2018.

[11] L. Trestioreanu, W. M. Shbair, F. S. d. Cristo, and R. State, "Xrp-ndn overlay: Improving the communication efficiency of consensus-validation based blockchains with an ndn overlay," in *NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium*, 2023.

[12] F. Scheidt de Cristo, W. Shbair, L. A. Trestioreanu, and R. State, "Pub/sub dissemination on the xrp ledger," in *2023 IEEE Latin-American Conference on Communications (LATINCOM)*. IEEE, Nov. 2023. [Online]. Available: https://hdl.handle.net/10993/57069

[13] L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem," *ACM Transactions on Programming Languages and Systems*, pp. 382–401, 07 1982. [Online]. Available: https://www.microsoft.com/en-us/research/publication/byzantine-generals-problem/

[14] M. Castro and B. Liskov, "Practical byzantine fault tolerance," in *3rd Symposium on Operating Systems Design and Implementation (OSDI 99)*. New Orleans, LA: USENIX Association, 02 1999. [Online]. Available: https://www.usenix.org/conference/osdi-99/practical-byzantine-fault-tolerance

[15] G. A. F. Rebello, G. F. Camilo, L. C. B. Guimarães, L. A. C. de Souza, and O. C. M. B. Duarte, "Security and performance analysis of quorum-based blockchain consensus protocols," in *2022 6th Cyber Security in Networking Conference (CSNet)*, 2022, pp. 1–7.

[16] D. Magoni, "Network Topology Analysis and Internet Modelling with Nem," *International Journal of Computers and Applications*, vol. 27, no. 4, pp. 252–259, 2005. [Online]. Available: https://hal.science/hal-00344484

[17] 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*, ser. SAC '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 1277–1284. [Online]. Available: https://doi.org/10.1145/3555776.3577611

[18] Y. Mao, S. Deb, S. B. Venkatakrishnan, S. Kannan, and K. Srinivasan, "Perigee: Efficient peer-to-peer network design for blockchains," in *Proceedings of the 39th Symposium on Principles of Distributed Computing*, ser. PODC '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 428–437. [Online]. Available: https://doi.org/10.1145/3382734.3405704

[19] D. Balouek, A. Carpen Amarie, G. Charrier, F. Desprez, E. Jeannot, E. Jeanvoine, A. Lèbre, D. Margery, N. Niclausse, L. Nussbaum, O. Richard, C. Pérez, F. Quesnel, C. Rohr, and L. Sarzyniec, "Adding virtualization capabilities to the Grid'5000 testbed," in *Cloud Computing and Services Science*, ser. Communications in Computer and Information Science, I. I. Ivanov, M. van Sinderen, F. Leymann, and T. Shan, Eds. Springer International Publishing, 2013, vol. 367, pp. 3–20.

[20] W. M. Shbair, M. Steichen, J. François, and R. State, "Blockzoom: Large-scale blockchain testbed," in *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 2019, pp. 5–6.

[21] R. Baldoni, L. Querzoni, S. Tarkoma, and A. Virgillito, *Distributed Event Routing in Publish/Subscribe Communication Systems*. Springer Berlin Heidelberg, 02 2009.

[22] A. Afanasyev, J. Burke, T. Refaei, L. Wang, B. Zhang, and L. Zhang, "A brief introduction to named data networking," *MILCOM 2018 - 2018 IEEE Military Communications Conference (MILCOM)*, pp. 1–6, 2018.

[23] "Named data networking," online, accessed: October 2023. [Online]. Available: https://named-data.net/

[24] V. J. et al., "Named data networking (ndn) project," http://named-data.net/techreport/TR001ndn-proj.pdf, accessed: October 2022.

[25] S. F. D'Agostino and J. P. Timpanaro, "Ripple protocol performance improvement: Small world theory applied to cross border payments," *XIX Simposio Argentino de Ingeniería de Software (ASSE)*, pp. 143–154, 09 2018.

[26] R. Yousuf, Z. Jeelani, D. Khan, O. Bhat, and T. Teli, "Consensus algorithms in blockchain-based cryptocurrencies," in *International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT)*, 02 2021, pp. 1–6.

[27] M. Roelvink, M. Olsthoorn, and A. Panichela, "Log inference on the ripple protocol: testing the system with an empirical approach," Delft University of Technology, 06 2020. [Online]. Available: http://resolver.tudelft.nl/uuid:ee55a433-e514-4507-8912-4196f0a9ba1c

[28] I. Amores-Sesar, C. Cachin, and J. Mićić, "Security analysis of ripple consensus," 2020. [Online]. Available: https://arxiv.org/abs/2011.14816

[29] W. Bubberman and S. Roos, "Tls mitm attack on the ripple xrp ledger," online, TU Delft, 2020. [Online]. Available: http://resolver.tudelft.nl/uuid:393083dc-a364-477a-afc8-faaca0a244c6

[30] C. A. Roma and M. Anwar Hasan, "Energy consumption analysis of xrp validator," in *2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 2020, pp. 1–3.

[31] H. Aoyama, Y. Fujiwara, Y. Hidaka, and Y. Ikeda, "Cryptoasset networks: Flows and regular players in bitcoin and xrp," *PLOS ONE*, vol. 17, 08 2022.

[32] P. Moreno-Sanchez, N. Modi, R. Songhela, A. Kate, and S. Fahmy, "Mind your credit: Assessing the health of the ripple credit network," in *WWW '18: Proceedings of the 2018 World Wide Web Conference*, 04 2018, pp. 329–338.

[33] M. Touloupou, K. Christodoulou, A. Inglezakis, E. Iosif, and M. Themistocleous, "Benchmarking blockchains: The case of xrp ledger and beyond," in *Hawaii International Conference on System Sciences*, 01 2022.

[34] G. Naumenko, G. Maxwell, P. Wuille, A. Fedorova, and I. Beschastnikh, "Erlay: Efficient transaction relay for bitcoin," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 817–831. [Online]. Available: https://doi.org/10.1145/3319535.3354237

[35] J. Leitao, J. Pereira, and L. Rodrigues, "Epidemic broadcast trees," in *2007 26th IEEE International Symposium on Reliable Distributed Systems (SRDS 2007)*, 2007, pp. 301–310.

[36] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "Splitstream: High-bandwidth content distribution in cooperative environments," in *Peer-to-Peer Systems II*, M. F. Kaashoek and I. Stoica, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 292–303.

[37] J. Guo, M. Wang, B. Chen, S. Yu, H. Zhang, and Y. Zhang, "Enabling blockchain applications over named data networking," in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, 2019, pp. 1–6.

[38] G. Sedky and A. E. Mougy, "Bcxp: Blockchain-centric network layer for efficient transaction and block exchange over named data networking," in *2018 IEEE 43rd Conference on Local Computer Networks (LCN)*, 2018, pp. 449–452.

[39] T. Jin, X. Zhang, Y. Liu, and K. Lei, "Blockndn: A bitcoin blockchain decentralized system over named data networking," in *2017 Ninth International Conference on Ubiquitous and Future Networks (ICUFN)*, 2017, pp. 75–80.

[40] W. Shang, A. Afanasyev, and L. Zhang, "Vectorsync: Distributed dataset synchronization over named data networking," in *Proceedings of the*

4th ACM Conference on Information-Centric Networking, ser. ICN '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 192–193. [Online]. Available: https://doi.org/10.1145/3125719.3132106

[41] Z. Zhu and A. Afanasyev, "Let's chronosync: Decentralized dataset state synchronization in named data networking," in *2013 21st IEEE International Conference on Network Protocols (ICNP)*, 2013, pp. 1–10.

[42] M. Zhang, V. Lehman, and L. Wang, "Scalable name-based data synchronization for named data networking," in *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, 2017, pp. 1–9.