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

Lucian Trestioreanu, Flaviene Scheidt, Wazen M. Shbair, Jerome Francois, Damien Magoni, Radu State

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, Proof-of-Work-based Distributed Ledger Technologies like Bitcoin or Ethereum.**

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 transaction throughput. However, it is known that significant increases in the size of the XRP Ledger network would be challenging to achieve. The main reason is the 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

Distributed Ledger Technology (DLT) is relatively new and still evolving. Its development was fostered by a diverse and enthusiastic community that sometimes forgot lessons from the past related to efficiency, resilience, and security of communication. Many blockchains rely on *flooding* as a straightforward solution for addressing the one-to-many and many-to-many communication specifics of DLT, which leads to scalability limitations. In this context, scalability concerns either being able to add more nodes to the network (*Node*-wise scalability), or the ability to process more transactions per second (TPS) (*Throughput*-wise scalability). *Throughput*-wise scalability can be achieved on-chain (sharding), off-chain (payment channels, sidechains), or otherwise e.g. specific mechanisms like XRP Ledger's (XRPL) consensus, 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 increasing throughput, while Byzantine Fault Tolerant (BFT) blockchains offer higher throughput but can not easily scale node-wise [1]. For instance, XRPL relies on a peer-to-peer (P2P) flooding mechanism leading to a multitude of redundant messages being circulated.

Previous work highlighted how latency and bandwidth of the underlying communication from physical channels to protocols can limit the scalability [2], [3]. In this perspective, the community effort was mainly focused on mainstream blockchains like Ethereum (ETH) [4], [5] or Bitcoin (BTC), both being Proof-of-Work type (PoW) (recently, ETH switched to *Proof-of-Stake*). Projects like *Fibre* [6], *Falcon* [7] or *bloXroute* [8], 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 with XRPL as an illustrative example.

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.

This work evaluates to what extent *Squelching* can improve the performance of intra-ledger communication on XRPL, (*i.e.* decrease the number of consensus-related messages), thus reducing the computational overhead, and ultimately improving XRPL's *Node*-wise scalability. For evaluation purposes, two types of experiments are performed: (1) baseline experiments on the actual XRPL network measure the impact of flooding on the computational overhead of a node (without Squelching) and (2) experiments in a controlled distributed environment to evaluate the benefit of Squelching. Together, these experiments allow assessing how Squelching improves P2P connectivity. This paper contributes to filling a research gap regarding the efficiency of the underlying communication supporting the consensus protocols of BFT blockchains like XRPL.

Our contributions are four-fold: i) Highlight how the current flooding mechanism of XRPL contributes to limiting its *Node*-wise scalability; ii) Define a measurement method to evaluate the impact of Squelching in regards to a baseline version; iii) Apply the aforementioned method to assess Squelching quantitatively; iv) Discuss the results in the context of other proposed alternatives, namely XRP-NDN [9] and GossipSub [10].

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 results obtained are shown in Section IV. In Section V, alternative solutions are discussed; Section VI provides a broader overview of related work, and Section VII concludes.

II. Problem Refinement

A. Background on the XRP Ledger

Leslie Lamport previously showed that in a synchronous environment, consensus can be achieved if at most n out of $3n + 1$ parties involved are dishonest [11]. The *Practical Byzantine Fault Tolerance* [12] algorithm makes this practicable in asynchronous environments like Internet, paving the way to consensus-validation based DLTs. XRPL implements a variant of BFT consensus called Federated Byzantine Agreement (FBA) Consensus [13]: 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.

On XRPL, there are two main node types: *trackers* in charge of processing transactions, and *validators* used for consensus voting. Figure 1 illustrates the ledger creation. A new ledger is created from the previous one by applying a new set of Tx's, a process involving 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 flooding for the dissemination of transaction, proposal, and validation messages. This is effective to explore every path and reach every node, but inefficient because it also forwards a significant number of duplicates; but this also depends on the underlying topology. As a preliminary study,

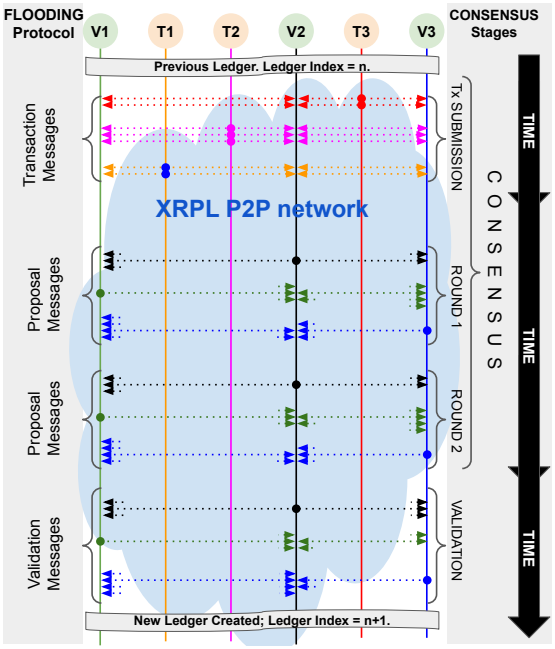


Fig. 1: The Ledger building process on XRPL.

we scanned and analyzed XRPL's topology as of 2021 with Nem [14]: it consisted of 892 nodes and 9197 edges; 152 nodes were *validators*. The average distance between a pair of nodes was 2.37, *topology diameter* was 5, and average degree 20.62. Thus on one hand, XRPL topology 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 affected by the high number of duplicates incurred when flooding is used, as the number of messages is proportional to number of edges [15].

C. Objective

Although XRPL *throughput* can reach 1500 TPS, scaling it *Node*-wise can be challenging due to the inefficient message flooding used and the dense network structure. Also as previously shown in [16], the number of XRPL *proposal* and *validation* messages represents 72% of all messages, so it is worth optimizing them. To enhance XRPL scalability by improving communication efficiency when disseminating these particular messages, *Squelching* was recently proposed. However, no in-depth assessments were performed. In this paper, our goal is to benchmark this solution, evaluate its potential benefit and compare the results with our previous assessments of other solutions based on NDN [9] and Gossipsub [10].

III. Squelching

A. Overview

The *Squelching* protocol was designed to optimize message relaying in the XRPL network. Its main goal is to reduce bandwidth consumption, CPU and memory load, and improve *Node*-wise scalability. Assuming a given validator, each node selects a subset of its peers 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 time, i.e., stop relaying messages to it. Hence, it is an active solution where a node assumes other nodes to behave compliant to the 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 the processing hosts. As a result, performance is expected to increase on multiple facets (lower CPU and bandwidth usage, lower message latency). We aim at quantitatively evaluate the achievable performance gain on production hosts.

Figure 2 exemplifies the squelching protocol [16], where a node switches between phase 2 and 3 once 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*.

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*.

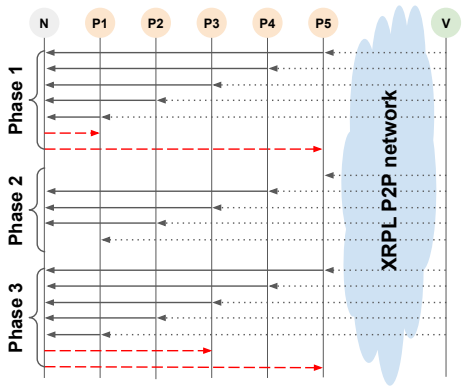


Fig. 2: Squelching sequence diagram.

B. Slot management

For explanation clarity, *uplinks* are the direct peers from which an XRPL node receives messages, while *downlinks* are the direct peers to which the given node sends messages to. So a node relays messages from its *uplinks* to *downlinks*.

Squelching introduces the concept of *slots* and *squelches* to manage message relaying among nodes. Each node 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 a node (tracker or validator) receives a *proposal* or *validation* message from one of its uplinks, it extracts the originating validator. The node then checks the corresponding slot, and if enough copies have been already received, a *squelch* message is sent back to the peer that just relayed this message. The squelch message asks the peer to stop forwarding further messages created by the respective originating validator. When receiving the *squelch* message, a peer removes the sender of the message from the *downlinks* for the corresponding slot (originating validator). Hence, 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 some time a squelch expires and the respective peer can be considered again by the node as candidate relay for the given originating validator. If a node loses an *uplink* it can replace it by sending *unsquelch* messages to peers. This allows to maintain enough connected peers to ensure security.

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 *first set* of experiments on XRPL MainNet we observed that mostly CPU usage is significantly impacted by the number of messages to handle. These experiments 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 MainNet, we measured as a function of number of peers: average CPU usage, number of messages received, and number of messages sent. This enabled us 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 experiments, to better control the experimental parameters, we deploy a small-scale XRPL network on a controlled and configurable HPC facility; squelching is applied, and its impact is monitored in terms of number of messages sent and received. Thanks to regression modeling, we will then extrapolate the number of messages saved in the Mainnet XRPL network and ultimately the number of free slots for additional peers to connect to. The baseline used for evaluation was an unmodified version of XRPL (XRPL v1.6), which was compared with a version [17] implementing the Squelching mechanism (XRPL v1.7).

B. Baseline experiments (first set, XRPL MainNet)

To perform the baseline experiments, a single node run on Ubuntu 22.04 with 16Gb RAM and 4 Intel Xeon E5-4650 v4

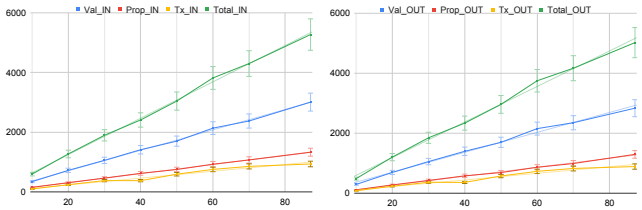


Fig. 3: IN/OUT number of msgs. *(y) vs* number of peers *(x)*, (Val: validation, Prop: proposal, Tx: transaction).

TABLE I: Mean CPU usage according to number of peers.

Peers	10	20	30	40	50	60	70
CPU	16.929	18.651	18.694	20.645	22.405	22.924	23.825

@2.2GHz is connected to the XRPL MainNet. Statistics are gathered with the RippledMon tool [18].

We configured the node to connect to an increasing number of peers over time, and collected average CPU usage and the number of messages sent and received. Results given in Table I and Figure 3 show a linear dependency between the number of peers and number of messages. As expected, 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. Due to the BFT-based validation mechanism, the number of exchanged messages varies according to the type of message, with *validations* being the most frequent. As such, Squelching could be fine-tuned for this type of message. This will still improve *Node*-wise scalability while enforcing higher robustness for other message types.

Due to a higher number of messages to be processed when the number of peers increases from 10 to 70, CPU usage also increases gradually approximately 35%, as in Table I.

C. Performance model

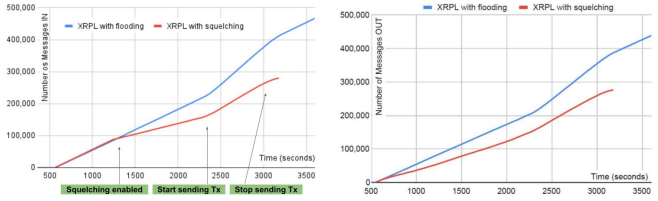
To assess the *Node*-wise scalability, we model the relationships between the resources used in terms of CPU usage, messages processed, and number of peers of an XRPL node. This allows to quantify the resources used by a node. Thanks to a linear dependency, simple regression allows to model CPU use (cpu) with respect to the number of peers the node is connected to ($peers$):

$$cpu = \beta_0 + \beta_1 peers \quad with\ \beta_0 = 15.8754, \beta_1 = 0.1177 \tag{1}$$

Messages are considered independently of their type. We do not distinguish CPU used by each type. Similarly, the total number of messages ($msgs$) is linearly dependent to number of peers ($peers$):

$$msgs = \alpha_0 + \alpha_1 peers \quad with\ \alpha_0 = -75.0943, \alpha_1 = 123.6365 \tag{2}$$

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).



(a) Total number of Messages IN (b) Total number of Messages OUT

Fig. 4: Total number of messages, *unmodified* vs. *squelching*.

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

D. Setup for squelching experiments (second set, Grid5000)

To assess Squelching, we deployed our own XRPL network to fully control the nodes. We built the testbed on Grid5000 (G5K) [19], [20], a large-scale HPC platform with interconnected sites in France and Luxembourg, featuring a large amount of resources: 15000 cores, 800 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, BlockZoom [21], [22] offering a reproducible environment for DLT experimentation.

The testbed was composed of 15 XRPL nodes evenly spread over five *G5K* locations (Luxembourg, Rennes, Nantes, Sophia, Lyon), meaning three nodes per site. Alternative solutions like Mininet come with their own limitations and challenges, for example, the requirements to run many XRPL nodes on the same machine. On the other hand, G5K is a research-oriented platform providing monitoring tools to access precisely the performance of a running algorithm/software and hardware on the platform. Moreover, G5K supports experiment reproducibility which is crucial for scientific analysis.

The nodes boot with no transactions being generated until a predefined time. Once this time expires, 1000 transactions per site were sent in parallel.

E. Overall Results

The comparative results of G5K experiments (second set) are presented in Figure 4 and Table II. Figure 4(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/stops sending *tx* messages.

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

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

much lower than its 15-peers equivalent on Mainnet (Figure 3). 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.

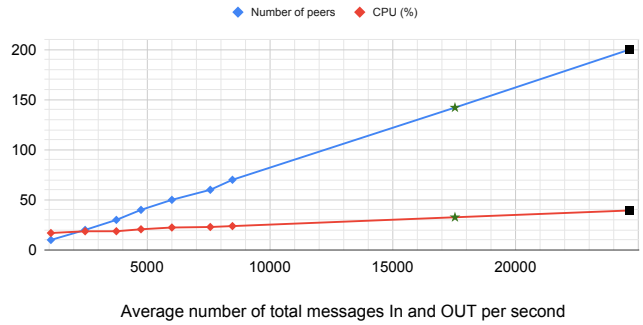


Fig. 5: Extrapolation of *number of peers* and *CPU(%)*.

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 5) is obtained by the linear regression models of equations (1) and (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 5) with our performance model (equations (1) and (2) again).

The result shows that a node connected to XRPL MainNet with 200 peers could save 17% CPU with *Squelching* and free up 58 peer slots. This actually enhances overall connectivity as a single node has 29% additional peers to connect to.

V. Discussion and Future Work

Squelching is a solution designed by XRPL, and most parameters are originally fixed. However, we expect to explore in future some parameters such as *timeout* or the *node_degree*. The number of peers a node keeps "alive" (does not squelch) at any given moment is an important aspect: a sparse network is more efficient w.r.t. the number of messages but less resilient, while a dense network will generate more duplicates, lower message latency, and will be more resilient [23]. Lowering too much the *node_degree* leads to security/resilience risks; too many active peers are "wasteful".

Even if our work is not focused on energy, the following can be considered: based on [24], saving 20% CPU roughly translates to 5% power saving. Noting that also memory and NIC would likely use less power when processing less data, we can expect savings better than 5%. Besides, network traffic can make up a large portion of a Stellar node's total consumption [25]. As Stellar shares design commonalities with XRP, there is a distinct possibility that network traffic takes a significant share of power in XRPL.

While Squelching is one solution to improve message dissemination, other alternatives for XRPL are discussed below.

(A) Gossipsub [4] is a publish-subscribe [26] protocol enabling efficient and scalable message dissemination in P2P networks. Gossipsub nodes engage in a gossip-type protocol by selecting a sub-set of peers to share messages with, which then propagate them again recursively. There can be different "topics of discussion" between the nodes, for which different sets of peers are used at a given moment. The peer selection algorithm is complex and makes use of an extensive set of parameters. Gossipsub was originally proposed for Ethereum and Filecoin, and we previously proposed an adaptation for XRPL named FlexiPipe [10]. To facilitate the discussion, we partially reproduce a relevant result from [10] in Figure 6, where this solution was compared with *squelching* and *flooding*. 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.

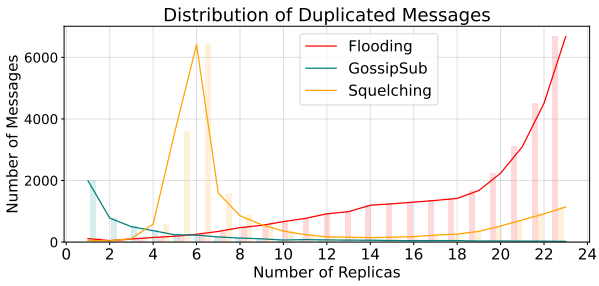


Fig. 6: Frequency of duplicated messages on XRPL with Flooding vs Squelching vs Gossipsub [10].

Also, compared to Squelching, Gossipsub relies on many parameters for fine-tuning, and most of all, it assumes the deployment of a distinct overlay next to the native XRPL overlay. Actually, the main advantage of *squelching* is to directly work over the XRPL overlay itself. The 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) [27]–[29] allows users to retrieve content by expressing what they want, similar to requesting a book by title: a user sends an interest packet with the data name. The forwarding is based on name prefixes instead of IP prefixes, and the request reaches one or multiple content providers or in-network caches with the expected content. Data is sent on the backward path and network nodes cache it distributively.

In [9] 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. However like Gossipsub, NDN implies an additional overlay network.

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

While other solutions like dissemination trees may yield better results, they exhibit several drawbacks compared to flooding-like solutions as squelching or gossipsub: i) maintaining a tree structure is complicated and costly; ii) latencies increase as paths are longer; iii) topologies that change continuously force tree rebuilds which can impact latency and message delivery (service availability), and iv) resilience to attacks and node churning.

VI. Related Work

XRP LCP was described in 2014 [30] followed by a rich literature investigating its robustness and security. It was analyzed in [31]–[33] and investigated empirically in [34]. This contrasts with our work focused on the message dissemination mechanism. In 2020, relatively simple cases were identified where consensus may violate safety and/or liveness [35], and it was argued that XRPL needs a very close *synchronization*, *interconnection*, and *fault-free operation* between validators. This is another argument for considering also the efficiency and resilience of communication - the focus of our research.

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 [36]. 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* [15] 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 [37] while *Splitstream* [38] distributes the load of forwarding messages evenly between participants. Actually, *Gossipsub* [4], previously introduced, draws upon the general concepts of epidemic broadcast and gossip protocols.

While *XRP-NDN* was also reviewed here, there are other proposals to use NDN in the context of blockchains, however focusing on PoW type: *BoNDN* [39], proposes TX dissemination for BTC through a push model over NDN interests, and a subscribe-push model for block propagation. Another design for propagating TXs and blocks over NDN was proposed in [5] for ETH, while [40] sends blocks over a multi-layer design based on NDN to achieve 74% of BlockNDN's overhead [41]. Being content-oriented, NDN is suited for data synchronization and different protocols were proposed: *Vectorsync* [42], *Chronosync* [43], *Psync* [44]. However, they were not meant originally for the byzantine blockchain environment [5]; moreover, here we want to minimize the number of messages, and 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 beneficial. We evaluated its gain using a regression model to understand its impact on the production XRPL network, from measurements done in a controllable environment. An XRPL hub node can potentially benefit from a 29% connectivity increase when using *Squelching*.

At broad level, it is expected that with respect to message efficiency over flooding, *squelching* can benefit other blockchains too. For instance, in [45], the current broadcast of messages in STELLAR is inefficient and many scaling issues can be addressed by optimizing network traffic. By using Gossipsub, the combination of node-degree tweaking and epidemic broadcast improved ETH communication layer.

Because Squelching might impact the robustness and security of the consensus mechanism itself, future work will thus consider a thorough security evaluation.

Acknowledgment

This research was funded in whole, or in part, by the Luxembourg National Research Fund (FNR), grant references PRIDE15/10621687/SPsquared and INTER/14783140/GLADIS. For the purpose of open access, and in fulfilment of the obligations arising from the grant agreement, the author has applied a Creative Commons Attribution 4.0 International (CC BY 4.0) license to any Author Accepted Manuscript version arising from this submission. In addition, we thankfully acknowledge the support from the RIPPLE University Blockchain Research Initiative (UBRI) for our research.

REFERENCES

[1] M. Vukolić, "The Quest for Scalable Blockchain Fabric: Proof-of-Work vs. BFT Replication," in *Open Problems in Network Security*, 2016, pp. 112–125.

[2] M. Kuzlu, M. Pipattanasomporn, L. Gurses, and S. Rahman, "Performance Analysis of a Hyperledger Fabric Blockchain Framework: Throughput, Latency and Scalability," in *IEEE International Conference on Blockchain*, 2019, pp. 536–540.

[3] P. W. Eklund and R. Beck, "Factors that impact blockchain scalability," in *11th Int. Conf. on Management of Digital EcoSystems*, 2020.

[4] 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.

[5] 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.

[6] M. Corallo, "Fibre fast internet bitcoin relay engine," https://bitcoinfibre.org/, 2015, accessed: January 2024.

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

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

[9] 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 *IEEE/IFIP Network Operations and Management Symposium (NOMS)*, 2023.

[10] F. Scheidt de Cristo, W. Shbair, L. A. Trestioreanu, and R. State, "Pub/sub dissemination on the xrp ledger," in *IEEE Latin-American Conference on Communications*, 2023.

[11] L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem," *ACM Transactions on Programming Languages and Systems*, pp. 382–401, 07 1982.

[12] M. Castro and B. Liskov, "Practical byzantine fault tolerance," in *3rd Symposium on Operating Systems Design and Implementation (OSDI)*. New Orleans, LA: USENIX, 02 1999.

[13] 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.

[14] D. Magoni, "Network Topology Analysis and Internet Modelling with Nem," *International Journal of Computers and Applications*, vol. 27, no. 4, pp. 252–259, 2005.

[15] Y. Mao, S. Deb, S. B. Venkatakrishnan, S. Kannan, and K. Srinivasan, "Perigee: Efficient peer-to-peer network design for blockchains," in *39th Symposium on Principles of Distributed Computing*, 2020.

[16] 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

[17] XRPL - Foundation, "The squelching protocol," https://github.com/XRPLF/rippled/pull/3412, accessed: January 2024.

[18] M. Bhandary and R. Zhang, "Rippled monitor," https://github.com/ripple/rippledmon, accessed: January 2024.

[19] INRIA, CNRS, RENATER, and al, "Grid 5000," https://www.grid5000.fr/w/Grid5000:Home, accessed: January 2024.

[20] 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. Springer, 2013, vol. 367, pp. 3–20.

[21] W. Shbair, "BlockZoom," https://github.com/wshbair/BlockZoom, accessed: January 2024.

[22] W. M. Shbair, M. Steichen, J. François, and R. State, "Blockzoom: Large-scale blockchain testbed," in *IEEE International Conference on Blockchain and Cryptocurrency*, 2019.

[23] A.-L. Barabási and M. Pósfai, *Network science*. Cambridge: Cambridge University Press, 2016. [Online]. Available: http://barabasi.com/networksciencebook/

[24] C. Roma and M. A. Hasan, "Energy Consumption Analysis of XRP Validator," *IEEE International Conference on Blockchain and Cryptocurrency*, pp. 1–3, 2020.

[25] Wanecek, Wilhelm, "Electricity Consumption of a Distributed Consensus Algorithm," Lund University, Tech. Rep., 2021, Bachelor's thesis.

[26] R. Baldoni, L. Querzoni, S. Tarkoma, and A. Virgillito, "Distributed event routing in publish/subscribe communication systems," *Middleware for Network Eccentric and Mobile Applications*, 2009.

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

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

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

[30] 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

[31] 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

[32] 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*, pp. 143–154, 2018.

[33] 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.

[34] 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

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

[36] G. Naumenko, G. Maxwell, P. Wuille, A. Fedorova, and I. Beschastnikh, "Erlay: Efficient transaction relay for bitcoin," in *ACM SIGSAC Conference on Computer and Communications Security*, 2019.

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

[38] 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*, 2003, pp. 292–303.

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

[40] G. Sedky and A. E. Mougy, "BCXP: Blockchain-Centric Network Layer for Efficient Transaction and Block Exchange over Named Data Networking," in *43rd IEEE Conference on Local Computer Networks*, 2018, pp. 449–452.

[41] T. Jin, X. Zhang, Y. Liu, and K. Lei, "BlockNDN: A bitcoin blockchain decentralized system over named data networking," in *9th International Conference on Ubiquitous and Future Networks*, 2017, pp. 75–80.

[42] W. Shang, A. Afanasyev, and L. Zhang, "Vectorsync: Distributed dataset synchronization over named data networking," in *4th ACM Conference on Information-Centric Networking*, 2017, p. 192–193.

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

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

[45] M. Lokhava, G. Losa, D. Mazières, G. Hoare, N. Barry, E. Gafni, J. Jove, R. Malinowsky, and J. McCaleb, "Fast and secure global payments with stellar," *ACM Symposium on Operating Systems Principles*, 2019.