PCNsim: A Flexible and Modular Simulator for Payment Channel Networks

Gabriel Antonio F. Rebello[1,2], Gustavo F. Camilo[1], Maria Potop-Butucaru[2],
Miguel Elias M. Campista[1], Marcelo Dias de Amorim[2], and Luís Henrique M. K. Costa[1]

[1]Universidade Federal do Rio de Janeiro, Brazil
[2]Sorbonne Université, CNRS, LIP6, F-75005 Paris, France

***Abstract*—Payment channel networks (PCN) enable the use of cryptocurrencies in everyday life by solving the performance issues of blockchains. Nevertheless, the main implementations of payment channel networks lack the flexibility to test new proposals that can address fundamental challenges, such as efficient payment routing and maximization of the payment success rate. In this demo paper, we propose PCNsim, an open-source simulator based on OMNeT++, which fully reproduces the default behavior of a payment channel network. We build the simulator in a modular architecture that allows easy topology/workload customization and automates result visualization. The core mechanism of PCNsim implements the specifications of the Lightning Network. We evaluate our proposal with a dataset of credit card transactions in a scale-free topology and show that it successfully demonstrates the difference between two routing methods in different setups.**

***Index Terms*—Blockchain, payment channels, simulation.**

I. Introduction

Payment channel networks (PCN) provide a scalable solution to improve the performance of cryptocurrencies [1], [2]. A payment-channel network is an *off-chain* network that allows users to safely send tokens via payment channels without the need to publish each transaction in the blockchain. For example, in Figure 1, if Alice wants to send one token to Charlie, she can send the token to Bob, and Bob relays it to Charlie. Because payment channels are bidirectional, the token could also be sent in the backwards direction. This enables micro-transactions to occur in real-time and narrows the gap between cryptocurrencies and daily needs.

Nevertheless, payment channel networks present open challenges that we cannot address efficiently given the absence of flexible simulators. This forces researchers to develop their proposals over an existing PCN implementation, which imposes several disadvantages. First, researchers are restricted to use small testnet topologies instead of large customized networks and workloads that represent real use cases. Second, it is infeasible to test proposals in low-resource computers since each user needs to run a full blockchain node that downloads the complete history of transactions. Finally, some implementations, such as the Lightning Network Daemon (LND), are not optimized for high throughput, which can result in high-latency transactions even in small networks [3].

Our contribution. We propose PCNsim[1], a flexible, lightweight, and modular open-source payment channel network simulator. PCNsim includes topology and workload generator modules that allow researchers to test their proposals on different scenarios through simple commands. The core of PCNsim extends the OMNeT++ network simulator to accurately reproduce a PCN routing model. Our simulator follows the message format defined in the Lightning Network set of specifications [4] and we can extend it to simulate other PCN implementations. Furthermore, PCNsim allows users to model channel parameters, such as capacity and fees, based on real data collected from the Lightning Network. As our design is focused on network and payment routing simulation, PCNsim waives the requirement of using an underlying blockchain and does not demand high storage capacity.

[1]PCNsim is available at https://github.com/gfrebello/pcnsim.

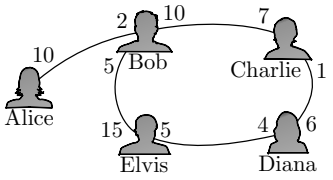

Fig. 1: A payment-channel network (PCN) composed of bidirectional payment channels with limited capacity. Users can route payments through intermediaries.

II. System Architecture

We develop PCNsim's architecture in Python, except the core module, which we write in C++ for faster transaction processing. The architecture, depicted in Figure 2, has five separate modules: (i) topology generator, (ii) workload generator, (iii) core simulator, (iv) result visualizer, and (v) result storage.

The topology generator and the workload generator provide, respectively, the network topology and the transaction set to the simulation. The topology generator allows users to import real network topologies or create topologies using random graph models. PCNsim allows users to accurately model network parameters, such as channel capacity and payment fees, with real-world PCN information collected from a snapshot of the Lightning Network. Our workload generator selects two random end-hosts from the topology to act as payment sender and receiver for each transaction. As transaction information in PCNs is usually private, PCNsim models transaction amounts by sampling values from real-world data such as credit-card

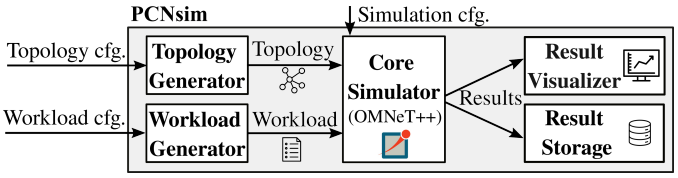

Fig. 2: PCNsim's architecture. Users can easily customize topologies and workloads via configuration files.

and e-commerce transactions, yielding realistic results. We provide an intuitive API with several models and datasets for easy module customization.

PCNsim's core simulator replicates the behavior of a payment channel network and gathers statistics. We implement the specifications of the Lightning Network's Basis of Lightning Technology (BOLTs) into the OMNeT++ simulator, including the necessary message exchanges to establish Hashed Timelock Contracts (HTLC) and forward payments. PCNsim keeps HTLCs in memory rather than on disk, which enables fast transaction processing and high-throughput payment routing. Researchers can incorporate proposals such as new routing methods and payment congestion control mechanisms into the core simulator and define which statistics they want to measure. As PCNsim runs on top of OMNeT++, the simulator also supports OMNeT++ extensions like INET to build payment channels on top of wireless communication protocols and mobile networks. PCNsim's core also removes the need for setting a full blockchain node, creating a lightweight alternative for users with low-resource computers.

The result visualizer displays system statistics as gathered by the core simulator, such as the average payment success rate or how the capacity of payment channels evolves, and allows researchers to easily analyze the impact of their proposals via automatically-generated graphs. Finally, the result storage module persists the statistics into the disk for further analysis.

III. Comparing Routing Methods

We compare two routing methods (R_M) to demonstrate how PCNsim works: (i) `fee`: Lightning Network's fee-minimization, which is a Dijkstra's shortest path algorithm using channel fees as weights, and (ii) `cap`: a variation of Dijsktra's shortest path algorithm that uses the inverse of the channel capacities as weights. The goal of `cap` is to maximize the chance that the payment will reach its destination by routing payments through high-capacity channels. We compare the methods for different transaction values (T_V): (i) small transactions ($T_V = 10$€), (ii) large transactions ($T_V = 200$€), and (iii) real credit-card transactions[2] (average $T_V = 88.3$€). We simulate a scale-free network with 10 nodes and channel capacities and fees sampled from the Lightning Network. The workload comprises 1,000 transactions between random end-hosts in the network.

We show a time-series of the payment success rates for all scenarios as measured by a random end-host in Figure 3. We observe two interesting findings. First, increasing the transaction value impacts the payment success rate because large payments have a higher chance of failing when traversing a bottleneck in the path. Second, the capacity-based variation of Dijkstra's algorithm, *cap*, yields higher success rates in all cases and even compensates the difference in transaction values between credit-card transactions and small transactions. Hence, although minimizing fees reduces the cost for the end-host, it incurs a higher chance of not completing the payment at all.

[2]Dataset available at https://www.kaggle.com/mlg-ulb/creditcardfraud.

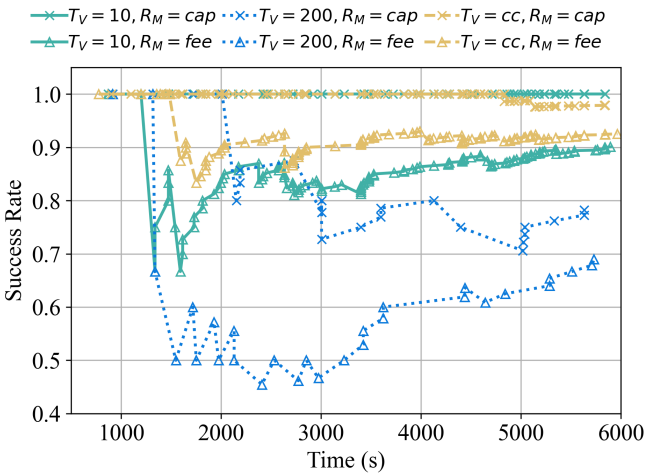

Fig. 3: Transaction success rate in our simulated PCN for several transaction values (T_V) and routing methods (R_M). The results demonstrate that the Dijkstra's shortest path approach with channel capacities as weights is more effective than the Lightning Network's fee minimization approach for all cases.

IV. Conclusion

Payment channel networks still present many open challenges that must be addressed before its mass adoption. We propose PCNsim, a PCN simulator that allows researchers to test new ideas intuitively and flexibly. Our demonstration shows that we can use the proposed system to efficiently compare different routing methods and measure their impact on the behavior of nodes and channels in a PCN.

References

[1] G. A. F. Rebello, G. F. Camilo, L. C. B. Guimarães, L. A. C. de Souza, G. A. Thomaz, and O. C. M. B. Duarte, "A security and performance analysis of proof-based consensus protocols," *Annals of Telecommunications*, pp. 1–21, 2021.

[2] J. Poon and T. Dryja, "The bitcoin lightning network: Scalable off-chain instant payments," 2016, Last access: 12 January 2022. [Online]. Available: https://www.bitcoinlightning.com/wp-content/uploads/2018/03/lightning-network-paper.pdf

[3] V. Sivaraman, S. B. Venkatakrishnan, K. Ruan, P. Negi, L. Yang, R. Mittal, G. Fanti, and M. Alizadeh, "High throughput cryptocurrency routing in payment channel networks," in *17th USENIX NSDI*, 2020, pp. 777–796.

[4] J. Poon and O. Osuntokun, "BOLT #2: Peer protocol for channel management," 2021, Last access: 12 January 2022. [Online]. Available: https://github.com/lightningnetwork/lightning-rfc/blob/master/02-peer-protocol.md

This paper was funded by CNPq, CAPES - Finance Code 001, FAPERJ, and FAPESP (18/23292-0, 15/24494-8, 15/24514-9, 15/24485-9, 14/50937-1).