# arXiv:2108.09509v2 [cs.NI] 15 Mar 2022

# Hop-by-hop Accounting and Rewards for Packet dIspAtching

Caciano Machado\*, Renan R. S. dos Santos<sup>†</sup> and Carla Merkle Westphall<sup>‡</sup>

Network and Management Laboratory – LRG

Graduate Program in Computer Science - PPGCC

Federal University of Santa Catarina – UFSC

PO Box 476, 88040-900 - Florianópolis, Brazil

caciano.machado at ufrgs.br\*, renan.rocha at grad.ufsc.br<sup>†</sup>, carla.merkle.westphall at ufsc.br<sup>‡</sup>

Abstract-Community networks are prone to free-riders, i.e., participants who take advantage of cooperation from others' routers but do not contribute reciprocally. In this paper, we present HARPIA, a system for credit-based incentive mechanisms for data forwarding in community networks aimed to prevent selfish behavior. HARPIA does not require a trusted thirdparty or tamper-resistant security modules as in other incentive mechanisms. Instead, it uses a distributed accounting scheme (DPIFA) to estimate the balance of data forwarding contribution and consumption of each network router and settle correspondent cryptocurrency debts on an Ethereum smart contract. On-chain settlement transactions are performed every HARPIA cycle (e.g., daily, weekly, monthly) and must be validated by at least m-of-n network routers using a multi-signature scheme (MuSig). We also realized a performance evaluation, security threat assessment, and cryptocurrency costs estimation. Results show that our proposal is suitable for community networks with up to 64 infrastructure routers under specific m-of-n MuSig thresholds.

*Index Terms*—community network, free-riding, incentive mechanism, blockchain, multi-signature, smart contract.

# I. INTRODUCTION

Despite the huge Internet expansion since the ARPANET, there are many regions and populations in the world that still lack connectivity. Today, the digital divide is another factor that deepens socioeconomic inequality. In community networks, to fill the gap left by insufficient market-based and state-sponsored Internet access solutions, community members deploy and operate the network infrastructure [1]. Still, community networks are prone to the free-rider problem [2], i.e., routers from selfish participants who take advantage of the cooperation from others' routers but do not contribute reciprocally. Free-riders arbitrarily discard or delay packets from others' routers to save network, computing, and energy resources. This behavior tends to undermine network reliability and inhibit its expansion. Cooperation enforcement (or incentive) mechanisms [3] have been proposed to mitigate potential free-riders in computer networks. Those mechanisms adopt credit-based, reputation-based, or algorithmic game theory to encourage cooperative behavior among routers.

Credit-based mechanisms model the data-forwarding task as a service valuated and charged using a virtual currency to regulate the dealings among various routers in multihop networks. Two approaches have been adopted for secure credit-based mechanisms: tamper-resistant security modules (TRSM) attached to the network interfaces that secure credit accounting and virtual banks that depend on a trusted thirdparty (TTP) service for centralized accounting. Recently, blockchains started to be adopted in credit-based incentive mechanisms to manage virtual coins in a trustless based approach, i.e., eliminating the need for centralized elements or TRSM. Though, existing proposals present significant limitations. The main limitations found are the need for a TTP or frequent expensive on-chain transactions.



Fig. 1. HARPIA architecture

This paper proposes the *Hop-by-hop Accounting and Re-wards for Packet dIspAtching* (HARPIA) (Fig. 1), a system for credit-based data forwarding incentive mechanisms built on top of the Ethereum blockchain. HARPIA relies on network traffic accounting based on a decentralized version of the PIFA from Yoo *et al.* [4] called DPIFA. Instead of relying on TRSM or TTP, routers share network accounting reports so that any router can validate its credibility. HARPIA does not require frequent on-chain transactions, but only periodic ones (e.g., hourly, daily, weekly, monthly), secured by *m*-of-*n* MuSig, that settles pending debts correspondent to routers' utilization and contribution in data forwarding.

The goal of this work is to present HARPIA and investigate its advantages, limitations, and applications. Our contribution is a new blockchain-enabled system for credit-based incentive mechanisms suitable for community networks. The main advantages of HARPIA compared to related works (Tab. V) are the independence of a TTP or TRSM and fewer on-chain transactions.

The rest of the paper is organized as follows. The next section summarizes the fundamental concepts used in this paper. Section III presents the HARPIA architecture and its components. Section IV is a performance analysis of com-

## This paper has been accepted in the IEEE TRUSTCOM 2021 (https://doi.org/10.1109/TrustCom53373.2021.00152)

©2021 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

puting, storage, communication, and cryptocurrency costs involved in HARPIA. Section VI presents a comparative analysis of related works. Section V is a security threat assessment. Finally, Section VII shows our conclusions and future works.

# II. PRELIMINARIES

This section introduces the technologies and cryptographic building blocks of HARPIA. First, we summarize blockchain and smart contract concepts. Second, we describe the work from Yoo *et al.* [4] that inspired DPIFA. Finally, we present MuSig and how to build a m-of-n multi-signature scheme.

# A. Blockchain and smart contracts

Blockchains are distributed databases organized as sequential chains of blocks secured using cryptographic techniques. Each ordered block contains transactions validated by a consensus protocol, e.g., Proof of Work (PoW), Proof of Stake (PoS), Proof of Authority (PoA), or byzantine algorithms. Each participant can trace and verify the correctness of the data. Recorded data is hard to tamper with, ensuring immutability and irreversibility. Bitcoin was the first blockchain that proposed a secure and P2P system independent of a trusted third-party for payments over the Internet [5]. It secures transactions of the most used and valuable cryptocurrency (BTC) today and inspired many other systems for innovations other than financial transactions over the Internet.

Blockchains such as Bitcoin also allow encoding rules and scripts for processing transactions. This feature evolved to support programs called smart contracts [6]. The consensus protocol automatically enforces smart contracts' trusted execution in a traceable and irreversible way. Ethereum [7], for example, transforms blockchains in finite state machines, which state transitions are equivalent to cryptocurrency transactions, enabling secure and decentralized applications.

# B. PIFA

The Protocol Independent Fairness Algorithm (PIFA) [4] is a data forwarding incentive mechanism for computer networks. The algorithm assumes that routers do not know full path information from source to destination of packets so that it could be applied independently of the routing protocol.

TABLE I PIFA REPORT FIELDS

Field	Description
RID	ID of the reporter router
NID	ID of the neighbor router
I	Number of input packets from the neighbor
0	Number of output packets to the neighbor
S	Number of packets starting at the current router
Т	Number of packets terminated at the current router
OFN	Number of packets originated from the neighbor

The original version of PIFA has a central service called Credit Manager (CM) that receives network traffic reports from all network routers. Periodically, every reporter router (RID) sends one traffic report for each of its interfaces to the CM. Each message has the traffic statistics with a specific neighbor router (NID) within a period identified by a sequence number (SEQ). The report fields are described in Tab. I. With the aggregated reports, CM can infer the current network topology and the credibility of individual reports based on the following three criteria (considering that  $Q_{n,m}$  denotes the Q field of a report where n = RID and m = NID are neighbors).

a)  $O_{n,m} = I_{m,n}$ : The number of output packets from a router must be the same as the number of input packets to the opposite side router for every link.

b)  $F_n$ : The number of packets forwarded by a router n with  $A_n$  neighbors must be the same as the number of input packets that have not terminated on it and the number of output packets that have not been originated from it:

$$\sum_{m \in A_n} \mathbf{I}_{n,m} - \sum_{m \in A_n} \mathbf{T}_{n,m} = \sum_{m \in A_n} \mathbf{O}_{n,m} - \sum_{m \in A_n} \mathbf{S}_{n,m} \quad (1)$$

c)  $OFN_{m,n} = S_{n,m}$ : Finally, the value S from a router should be identical to the OFN of the next-hop router. OFN aims to prevent a malicious router from manipulating  $F_n$  by changing both  $\sum T$  and  $\sum S$  in criteria (b). This OFN is computed by counting the number of packets originated at a neighbor out of all input packets from that neighbor.

CM increases the credit for each router in proportion to  $F_n$ , and all routers have to pay these credits as much as the number of packets they originate. In networks using PIFA, routers would not deliberately avoid packet forwarding because they need to earn credits to send their packets.

# C. MuSig

MuSig [8] is a Schnorr-based multi-signature scheme that allows a group of signers to produce a short and joint signature on a common message. It is provably secure under the discrete logarithm assumption and in the plain public-key model (meaning that signers are only required to have a public key, but do not have to prove knowledge of the private key corresponding to their public key to some certification authority or other signers before engaging the protocol). The scheme also allows public key aggregation meaning that verifiers do not need all public keys used in the signature. These properties are helpful to provide compact multi-signatures in blockchain features such as multi-party transactions and permissioned consensus.

MuSig is parameterized by group parameters  $(\mathbb{G}, p, g)$ where p is a k-bit integer,  $\mathbb{G}$  is a cyclic group of order p, and g is a generator of  $\mathbb{G}$ , and by hash functions  $(H_{com}, H_{agg}, H_{sig}, H_{tree})$  from  $\{0, 1\}^*$  to  $\{0, 1\}^{\ell}$ . The multi-signature process is divided into three rounds as follows:

a) Round 1: A group of n signers wants to cosign a message m. Let  $X_1$  and  $x_1$  be the public and private keys of a specific signer, let  $X_2, ..., X_n$  be the public keys of other cosigners, and let  $\langle L \rangle$  be the multi-set of all public keys involved in the signing process. For  $i \in \{1, ..., n\}$ , the signer

computes  $a_i = H_{agg}(\langle L \rangle, X_i)$  as well as the aggregated public key

$$\tilde{X} = \prod_{i=1}^{n} X_i^{a_i}.$$
(2)

b) Round 2: The signer generates a random private nonce  $r_1 \leftarrow \mathbb{Z}_1$ , computes  $R_1 = g^{r_1}$  (the public nonce), the commitment  $t_1 = \operatorname{H}_{com}(R_1)$ , and sends  $t_1$  to all other cosigners. When receiving the commitments  $t_2, \ldots, t_n$  from all other cosigners, the signer sends  $R_1$  to them. This procedure ensures that the public nonce is not exposed until all commitments have been received. Upon receiving  $R_2, \ldots, R_n$  from other cosigners, the signer verifies that  $t_i = \operatorname{H}_{com}(R_i)$  for all  $i \in \{2, \ldots, n\}$ . The protocol is aborted if this is not the case.

*c) Round 3:* If all random public nonces can be verified using the commitments, then compute

$$R = \prod_{i=1}^{n} R_i, \tag{3}$$

$$c = \mathcal{H}_{sig}(\tilde{X}, R, m), \tag{4}$$

and

$$s_1 = r_1 + ca_1 x_1 \mod p.$$
 (5)

The signature  $s_1$  is sent to all other cosigners. When receiving  $s_2, ..., s_n$  from other cosigners, the signer can compute

$$s = \sum_{i=1}^{n} s_i \mod p. \tag{6}$$

Then, the signature is

$$\sigma = (R, s). \tag{7}$$

d) Verification: Given a signed message m and the respective aggregated public key  $\tilde{X}$  (or the set of public keys to compute  $\tilde{X}$  with Eq. 2), in order to verify if a multi-signature  $\sigma$  is valid, a verifier can compute

$$c = \mathcal{H}_{sig}(\tilde{X}, R, m), \tag{8}$$

and accepts  $\sigma$  if

$$g^s = R\tilde{X}^c.$$
 (9)

e) m-of-n: MuSig also allows implementing a threshold policy where m valid signatures of n possible ones are required to confirm a multi-signature. This feature can be implemented by building a Merkle hash tree [9] where the leaves are permitted combinations of public keys (in the aggregated form), and the nodes are  $H_{tree}$  hashes. The Merkle tree should be a complete binary tree. Thus the last aggregated public key combination is repeated until the number of leaves is a power of 2. The verification, in this case, would take as input an aggregated public key  $\tilde{X}$ , a multi-signature  $\sigma = (R, s)$  and a Merkle proof P. Its validity would depend on the multisignature being valid with the provided key, and the proof establishing that the key is, in fact, one of the leaves of the Merkle tree, identified by its root hash  $\mathcal{M}_{root}$ . This approach is only possible when the number of combinations of aggregated public keys is feasible, given a m-of-n threshold

# III. HARPIA

The Hop-by-hop Accounting and Rewards for Packet dIspAtching is a system that implements a data forwarding incentive mechanism suitable for community networks built on top of DPIFA, *m*-of-*n* MuSig, and Ethereum smart contracts. The next subsections present an overview of the mechanism and detail each component of HARPIA.

#### A. Overview

In our proposal, routers that originate packets should pay the next-hop routers that forward their packets. Next-hop routers would not deliberately avoid data forwarding because their owners want to be paid. HARPIA does not distribute tokens to routers. Routers should earn cryptocurrency tokens beforehand through Ethereum mining, an advertising model for content downloaded, raising donations, or any other means. HARPIA is focused on the dealings among infrastructure routers and do not impose any limitations for the number of client devices inside the local network of each infrastructure router.

DPIFA is the component responsible for the traffic accounting. It is inspired in PIFA but does not require a trusted thirdparty (CM). Instead, all routers share accounting reports and calculate how much cryptocurrency each router would pay or receive. Like PIFA, HARPIA is routing protocol-agnostic and does not require topology information to be known. It does not require that all the network routers participate in the system. However, only HARPIA members can engage in traffic accounting and cryptocurrency dealings. Additionally, HARPIA members must be network neighbors to produce credible traffic accounting.

HARPIA performs an on-chain settlement of pending debts on each HARPIA cycle. Each cycle takes at least  $\beta$  Ethereum blocks mined to complete considering an average block time  $\gamma$ . Each router aggregates DPIFA reports within a cycle to validate their credibility and calculate pending debts. Thus, any router can propose a transaction to update the cryptocurrency balance of all routers. If at least *m*-of-*n* routers sign this transaction, the proposer can call a smart contract function that verifies its validity and settles pending debts. Fig. 2 illustrates the messages in a typical cycle in a network with 4 routers. These messages are described in the next subsections, except for MuSig, already explained in Section II-C.

## B. DPIFA accounting reports

Each router keeps track of network traffic statistics for sent and received unicast packets according to DPIFA parameters. Periodically, e.g., every  $\lambda = 10$  minutes, each router disseminates DPIFA accounting reports to all other routers and keep the accounting information since the last HARPIA settlement.



Fig. 2. Messages in a HARPIA cycle

Each message corresponds to statistics on a neighbor link and has the same fields as the original PIFA shown in Tab. I.

After receiving DPIFA reports from the other routers, each router can calculate how much data each other router forwarded and how much data originated from it have been forwarded by neighbor routers. If a router loses reports at the current HARPIA cycle due to communication errors, router failure, or even malicious packet discarding, it can request the reports directly from their originators.

# C. HARPIA STP

In HARPIA, payments are not performed directly between pairs of neighbor routers. HARPIA computes DPIFA aggregated reports to calculate the total of cryptocurrency tokens that each router n would pay or receive ( $C_n$ ) according to their consumption and contribution in data forwarding. Any router could create a Settlement Transaction Proposal (STP) and disseminate it on every HARPIA cycle. In Fig. 2, routers a and d create and disseminate an STP. An STP includes a  $C_n$  entry for each router n calculated as shown in Eq. 10 where:  $A_n$  is the set of adjacent routers of router n;  $F_{n,m} = I_{n,m} - T_{n,m}$  is the amount of data received by router n (NID) from router m(RID) that has been forwarded toward the destination;  $S_{n,m}$ represents the data produced by router n and sent to router mthrough a link n-m;  $H_{avg}$  is the average hop count between two routers; and  $P_{avg}$  is the average link price. For each link n-m in the network, both routers n and m should specify the same link price  $(P_{n,m})$  in the smart contract, so that any router can calculate the STP and all the routers can validate it.

$$C_n = \sum_{a \in A_n} F_{n,a} \cdot P_{n,a} - S_{n,a} \cdot P_{avg} \cdot H_{avg}$$
(10)

The STP also includes a reward for its creator with a twofold goal. It serves as an incentive for the creator to perform the operation and compensate for the public blockchain transaction costs spent in mining.

All routers can verify the fairness of an STP using the traffic accounting credibility criteria described in Section II-B and the price of the network links  $P_{n,m}$  of each router. Every router that agrees with an STP sends back a confirmation using a signed acknowledge message to the STP creator. In Fig. 2, routers b and c confirm STP<sub>a</sub>, and router a answers with a message m that informs which routers confirmed the STP. A minimum percentage of members ( $\zeta$ ) should confirm an STP to engage in MuSig. In Fig. 2,  $\zeta = 75\%$ , equivalent to a 3-of-4 routers. If MuSig succeeds, the STP creator can send the transaction to the Settle smart contract function described in the next subsection. HARPIA MuSig uses the elliptic curve secp256k1 and the hash algorithm SHA-256.

HARPIA admits an error in accounting up to a percentage threshold  $\delta$ . For each router p that creates and disseminates an STP<sub>p</sub>, every other router l checks whether every  $C_n^p$  from this STP is within a local calculated  $C_n^l \pm \delta$ . Furthermore, our scheme allows each router to validate STPs according to other arbitrary criteria, such as router reputation mechanisms.

# D. HARPIA smart contract

The central component of our system is a smart contract built on top of the Ethereum blockchain using the Solidity language v0.7.5. A HARPIA smart contract instance for a network (from now on, we call it just as a HARPIA instance) manages membership and cryptocurrency dealings. The identification of each router member is its Ethereum account public address. Tab. II shows a list of HARPIA parameters set in the smart contract.

 TABLE II

 HARPIA PARAMETERS STORED IN THE SMART CONTRACT

Parameter	Description
β	HARPIA cycle, in number of Ethereum blocks
$\lambda$	DPIFA report period
ζ	m-of-n MuSig percentage threshold
$P_{n,m}$	Price (tokens/GB) for traffic from router $m$ to $n$
δ	Tolerated error for each $C_n$ in an STP
ξ	Duration of an STP, in number of Ethereum blocks
au	Minimum ether balance to keep as a member
$\phi$	Minimum ether deposit to Join an instance
$\mathcal{M}_{root}$	Merkle tree root of current $\tilde{X}$ combinations

a) Initialization and membership: To enable HARPIA in a network, a router must be an Ethereum full node and deploy a HARPIA smart contract instance on the blockchain. The deployment includes an implicit Join operation that assigns this router as the first member of this HARPIA instance. After that, the smart contract allows any neighbor router in the network to join this HARPIA instance using the Join function. Join and Leave operations are administrative tasks that require interaction among different routers' owners for approval. This interaction is a reasonable requirement for community networks because physical link establishment among the routers also depends on owners' negotiation. Join operations also require an initial deposit of ether, which minimum value  $\phi$ is defined in the smart contract deployment. This deposit is converted into HARPIA tokens valid within the smart contract instance.

b) Settle: A Settle function call validates and executes an STP. It can be called once per HARPIA cycle. At the end of the Settle function, any pending Join or Leave operation is completed by updating the Merkle tree root of current Xcombinations ( $\mathcal{M}_{root}$ ) in the smart contract. Pending Leave operations also withdraw the ether respective to the remaining HARPIA tokens for the router's Ethereum account. HARPIA limits the number of Join and Leave pending operations to one per Settle call. This is an acceptable limitation for community networks because they are relatively static when compared to other ad hoc networks.

An STP is valid for a time corresponding to  $\xi$  Ethereum blocks. Thus, the number of Ethereum blocks mined since the STP has been proposed and the Settle operation could not exceed  $\xi$ , otherwise, the Settle call fails.

Initially, each ether unit in the smart contract instance corresponds to one token, but the Settle call creates new tokens to reward the STP proposer, changing the rate between ether and tokens. This procedure works also as an inflation mechanism and inhibit inactive members in the HARPIA instance because idle tokens devalue every settlement, and could trigger an automatic Leave in the Settle call when a router's ether balance reaches  $\tau$ . Thus, HARPIA tends to keep only members that actively use the network, provide data forwarding services or create HARPIA STPs for the Settle function.

c) MuSig verification: Join, Leave and Settle function calls require a MuSig verification. This requirement means these transactions should precede a 3-round MuSig involving at least m-of-n of the current HARPIA instance members. The smart contract deployment defines the *m*-of-*n* threshold  $\zeta$  for MuSig.  $\zeta$  is a number in the interval [50 - 100) representing the minimum percentage of member signatures required in a MuSig. A threshold  $\zeta$  is satisfied by any *m*-of-*n* scheme that  $\frac{100m}{r} \geq \zeta$  (e.g., if  $\zeta = 75\%$ , the schemes 3-of-4, 4-of-5, and 5-of-6 satisfy  $\zeta$ ). The call for functions that require a MuSig verification should include the multi-signature  $\sigma$  for the transaction parameters, the respective aggregated public key X, and the Merkle proof P. The call validates MuSig in two steps. First, it validates X using P and  $\mathcal{M}_{root}$ . Second,

it validates  $\sigma$  using  $\tilde{X}$ . The implicit Join in the smart contract deployment does not require this verification, and Joins in HARPIA instances with only one member require only a simple signature of the current member instead of a MuSig. Furthermore, every time a router performs a Join or a Leave, all valid combinations of aggregated public keys and respective Merkle tree should be calculated using current members' public keys. This is a computation intensive operation that should be executed by all members individually. The router calling Join/Leave is responsible for writing a new  $\mathcal{M}_{root}$  in the smart contract.

d) Other functions: The smart contract also provides functions that do not require MuSig and allow: 1) to read information about current router members in a HARPIA instance; 2) to register/unregister/update router links and respective prices; 3) to buy/redeem tokens for a router so that it can pay for data forwarding.

# IV. EVALUATION

This section presents an evaluation of computational and cryptocurrency costs involved in HARPIA. We evaluated processing, storage and communication performance using Python, Gnuplot and R to identify potential bottlenecks. Also, we calculated Ethereum smart contract's gas requirements to estimate cryptocurrency costs. Through this evaluation we discuss the applicability of the system.



Fig. 3. Storage requirements for MuSig aggregated public keys with respective Merkle hash tree and DPIFA report messages

The main bottleneck of HARPIA resides in the creation of the aggregated public keys of all valid combinations of public keys  $(\mathcal{C}_{\tilde{X}})^{1}$  and all H<sub>tree</sub> (SHA-256) in the Merkle hash tree <sup>2</sup> required for Join and Leave. We evaluated the execution times using off-the-shelf hardware <sup>3</sup>. Tab. III shows results using chosen thresholds  $\zeta$  and Fig. 4 shows estimated times based on partial executions. Results show feasible execution times for specific thresholds. For example, if we set HARPIA cycle time  $(\beta \cdot \gamma)$  to 1 day, then any threshold  $\zeta$  with execution time

 ${}^{1}C_{\tilde{X}} = \sum_{k=m}^{n} \frac{n!}{k!(n-k)!}$  possible  $\tilde{X}$  in MuSig *m*-of-*n*  ${}^{2}\mathcal{C}_{\tilde{X}} - 1$  H<sub>tree</sub> operations in the Merkle hash tree

<sup>3</sup>Single threaded Python script. Linux Kernel 5.4.0, glibc 2.31, Python 3.8.5 w/ fastecdsa 2.1.5. CPU Intel Core i7-8550U 4GHz, Cache L2 8MB, RAM 16GB, SSD storage w/ 31.6 Gb/s speed.

under 1 day would be acceptable. A plane with level curves is fixed at 16 hours ( $\approx 10^{4.76}$  seconds) execution time in Fig. 4. If we arbitrarily set 16 hours as a limit, any valid m and nthat result in execution times under the plane are acceptable m-of-n configurations.

TABLE III Execution times to calculate  $\mathcal{C}_{ ilde{X}}$  and respective Merkle hash TREE

$\zeta = \min(m \text{-of-} n)$	Avg. of 30 execs. for each number of routers			
<b>,</b> (	8	16	32	64
75%	0.308s	39.17s	-	_
87.5%	0.100s	2.490s	1547s	-
93.75%	-	0.335s	21.15s	53458s



Fig. 4. Estimated times to calculate  $C_{\tilde{X}}$  and Merkle hash tree

Fig. 3 illustrates how the storage costs scale in HARPIA with the number of routers. MuSig storage requirements<sup>4</sup> are shown for different *m*-of-*n* thresholds  $\zeta$ . DPIFA storage requirements<sup>5</sup> are shown for different cycle times  $\beta \cdot \gamma$  considering an average of  $\nu = 5$  neighbors and period  $\lambda = 10$ minutes. The evaluation shows that HARPIA is suitable for community networks up to 64 infrastructure routers with attached storages to keep MuSig and DPIFA data. In a realistic scenario ( $n = 32, \beta \cdot \gamma = 1$  week, and  $\zeta = 87.5\%$ ), HARPIA requires 36MB for DPIFA records<sup>[5]</sup>, 3.83MB for MuSig<sup>[4]</sup>, and 419GB<sup>[7]</sup> for the Ethereum blockchain.

A typical HARPIA cycle disseminates around 5.24MB of messages (Fig. 2) from DPIFA, STP and MuSig among all routers considering a community network with 64 routers,

 ${}^{5}116\nu \cdot n\frac{\gamma \cdot \beta}{\lambda}$ . Each DPIFA report message with 116 bytes: 44 bytes for DPIFA fields, 4 bytes for a timestamp, 4 bytes for a nonce, and 64 bytes for the digital signature.

cycle time  $\beta \cdot \gamma = 1$  day, period  $\lambda = 10$  minutes, and an average of  $\nu = 5$  neighbors. For bigger networks, processing times and storage space requirements become a bottleneck before the network overhead.



Fig. 5. HARPIA gas costs

TABLE IV HARPIA GAS COSTS IN US\$<sup>A</sup> (APRIL, 28 2021)

Gas cost	Cost in Mainnet	Cost in Classic			
$pprox$ 4.52M $^{ m d}$	269.41	0.73			
<sup>a</sup> Cost in US\$: Gas cost × Gas price <sup>b</sup> × Exchange rate <sup>c</sup> <sup>b</sup> Gas price: $\approx 22$ Gwei <sup>e</sup> in Mainnet and $\approx 4.8$ Gwei <sup>e</sup> in Classic					
<sup>c</sup> Exchange rate: 1 ETH = US	\$ 2,709.34 and 1 ET	C = US\$ 34.02			

<sup>d</sup> Gas limit:  $\approx$  14.9M in Mainnet and  $\approx$  8M in Classic

<sup>e</sup> 1 Gwei = 1 ether  $\times 10^{-9}$ 

Smart contracts' execution consumes ether, the Ethereum cryptocurrency. The amount of ether that a function requires is measured in gas units. Gas prices oscilate according to blockchain transaction's demand. It is crucial to estimate gas costs involved in the smart contract because the Ethereum blockchain imposes a gas limit for functions and to figure out the scenarios that HARPIA is applicable. We used Remix <sup>6</sup> to calculate smart contract gas costs, and obtained gas prices, gas limits and exchange rates from Tokenview<sup>7</sup>. Fig. 5 shows gas costs for functions that require MuSig for different  $\zeta$  values. We can identify that the number of routers in a HARPIA instance has little influence on gas costs. Tab. IV shows gas costs for function calls converted to US\$ using current exchange rates for Ethereum Classic (ETC) and Ethereum Mainnet (ETH). Estimations indicate that the HARPIA smart contract has relatively high gas costs in the Mainnet but low costs in Classic. HARPIA applicability would depend on the community network budget and the frequency of smart contract calls. This frequency can be adjusted with a longer HARPIA cycle that reduces the number of Settle, Join and Leave operations.

# V. SECURITY THREATS

The main goal of this paper was the presentation of HARPIA and an evaluation of its applicability. However, there

 $<sup>{}^{4}33\</sup>mathcal{C}_{\tilde{X}} + 32(2\mathcal{C}_{\tilde{X}} - 1)$ . 33 bytes for each  $\tilde{X}$  (elliptic curve point in the compact form) and 32 bytes for each  $H_{tree}$  (SHA-256).

<sup>&</sup>lt;sup>6</sup>https://remix.ethereum.org/

<sup>7</sup>https://tokenview.com

is a series of security threats specific to our protocol that we discuss here.

*a) Key management:* HARPIA does not need to rely on PKI or any other trusted third-party. Instead, the router's public keys (for ECDSA and MuSig) must be registered in the smart contract on Join operations. Join operations require a MuSig involving current HARPIA instance members. This MuSig depends on an admission process (for example, during a periodic meeting of current community network members) that validates new members' public keys. Details of such an admission process are outside the scope of this paper. Also, MuSig is provably secure under the plain public-key model, and then members do not have to prove knowledge of the private key corresponding to their public key.

b) Malicious DPIFA reports: Similarly to PIFA, we assume that truth-telling will be the dominant strategy because otherwise, neighbor routers will not engage in the HARPIA instance. Routers will leave the system if they do not have their data properly forwarded or their data forwarding services correctly paid. Any router can verify accounting correctness using aggregated DPIFA reports. We also assume that there is no collusion among routers. Nevertheless, we digitally sign (ECDSA) DPIFA records with the Ethereum private key of its creator using timestamps and nonces to prevent replay attacks.

c) Traffic spoofing: A router can spoof the source address from its packets to cheat the next-hop router to believe that those packets have originated from another router. The goal is to avoid these packets from being counted as their traffic on DPIFA reports, thus, avoiding paying the next-hop router. These inconsistencies will influence in DPIFA report validations. Nonetheless, an additional countermeasure can be applied for this specific threat. Routers could implement traffic authentication using a public/private key scheme. In this case, routers should register the public keys of their interfaces on the HARPIA smart contract so that all the other routers can authenticate their traffic.

d) Malicious STPs: We assume that a majority of honest routers will never validate unfair STPs. It is not in the interest of routers to produce malicious STPs because they want their STPs to be credible and validated by other routers to receive a reward for their proposals. For the same reason, we assume that routers will not engage in producing valid STPs and maliciously aborting the process at any stage after the 3round MuSig begins. A malicious router would adhere to this behavior to undermine a HARPIA instance postponing Settle operations indefinitely. However, it is more likely that the router engages in submitting the transaction on-chain as soon as possible to be rewarded. Nevertheless, STPs contain the current Ethereum block hash and are digitally signed using timestamps and nonces to prevent replay attacks.

e) Quorum for MuSig: There is a risk that the m-of-n MuSig quorum cannot be achieved because part of the routers or respective network links are faulty or even because they lost their private keys. To prevent this situation, the inflation mechanism in the Settle operation removes from the HARPIA

instance routers which cryptocurrency balance is under a predefined threshold  $\tau$ .

# VI. RELATED WORKS

This section highlights key aspects of representative works in blockchain-enabled data forwarding incentives for computer networks [10] summarized in Tab. V. Those works have been designed for community networks [11], [12], device-to-device (D2D) networks [13], delay-tolerant networks (DTN) [14] and the Internet [15], [16].

Most of the works deal with the free-riding problem similarly to pre-blockchain credit-based incentive mechanisms. In this context, packet forwarding is a service rewarded with cryptocurrency. We name as *payment proofs* and *forwarding proofs* the methods for assuring that a party paid and the other party performed the correspondent packet forwarding correctly. However, we understand that some mechanisms unify these proofs, i.e., the same mechanism provides both proofs.

Payment proofs are secure blockchain transactions for cryptocurrency transfers that fall into two categories: onchain [11], [12], [14], [15] or off-chain [13]. The first consists in typical blockchain transactions that require mining (e.g., PoW) or validation (e.g., PoS or PoA) for consensus. Usually, validation allows faster (more transactions per second) and cheaper (fewer cryptocurrency fees) transactions. Off-chain transactions consists in blockchain scalability solutions [17] such as channels and childchains.

Forwarding proofs can be classified according to two criteria. First, according to the mechanism itself: traffic monitoring [11], [12], [15] or cryptographic receipts in the network protocol [13], [14]. Second, whether they need a trusted thirdparty [11]–[13], [15] or not [14], [16].

RouteBazaar [15], MeshDapp [11] and Althea [12] need to trust in the agents that perform traffic accounting. In Route-Bazaar, the intermediate autonomous systems (AS) monitor traffic and perform expensive on-chain accounting. MeshDapp relies on an oracle service responsible for the network traffic monitoring. In Althea, every pair of neighbors should perform traffic monitoring for its neighbors' network interfaces. Also, Althea has a method to detect fraud in neighbors' accounting that uses tunnels to exit nodes on the Internet.

Receipt-based forwarding proofs can be combined with onchain [14] or off-chain transactions [13]. While LOT49 [13] supports cheap off-chain transactions with the Lightning protocol micropayments, it requires a witness node that acts as a trusted third-party. Trautmann and Burnell's patent [16] describes a system that introduces a Proof of Routing (PoR) scheme that can securely implement a blockchain network and provide useful consensus. Their blockchain-based router idea includes different nodes that process data packets between endpoints to produce cryptographic proofs, similarly to PoW schemes. Nodes can be router nodes, which analyze and route data packets, or block nodes that manage collections of specially labeled packets and generate new blocks in the blockchain.

 TABLE V

 Related works in blockchain-enabled data forwarding incentives for computer networks

System	Blockchain	Network	Payment proof	Forwarding proof	TTP
Truthful Inc. [14]	Bitcoin	DTN	On-chain PoW	Receipts	No
MeshDapp [11]	Ethereum	Com. Net.	On-chain PoA	Traf. acct.	Oracle service
RouteBazaar [15]	Bitcoin	Internet	On-chain PoW	GRE tun. acct.	Intermediate ASs
Althea [12]	Cosmos	Com. Net.	On-chain PoS	VPN tun. acct.	Peers and exit nodes
LOT49 [13]	Bitcoin	D2D	Off-chain Lightning	Receipts	Witness nodes
Rout. Based Blockc. [16]	-	Internet	On-chain PoR	PoR	No
HARPIA	Ethereum	Com. Net.	On-chain PoW	DPIFA	No

# VII. CONCLUSIONS AND FUTURE WORKS

This paper presented HARPIA, a blockchain-enabled system for credit-based incentive mechanisms for data forwarding in computer networks. Unlike related works, HARPIA does not require frequent on-chain transactions, and is independent of a TTP or TRSM. HARPIA is built on top of DPIFA traffic accounting and Ethereum smart contract transactions secured by m-of-n MuSig.

Our evaluation shows that lower *m*-of-*n* MuSig thresholds increase processing, network, and storage requirements with combinatorial complexity. Also, DPIFA storage requirements increase linearly with shorter periods and longer cycles. The analysis reveals that HARPIA is suitable for community networks with up to 64 infrastructure routers with thresholds above 75% and periods greater than 10 minutes. These limitations are acceptable since a considerable number of community networks have less than 64 routers (e.g., 60% of Freifunk list <sup>8</sup>). HARPIA smart contract's gas costs are relatively high in the Ethereum Mainnet and present affordable fees in the Ethereum Classic. Gas costs can also be reduced increasing the cycle to produce fewer MuSig signed transactions.

For future works, we will simulate representative community networks' topologies under different HARPIA parameters emulating selfish and malicious routers. We also plan to investigate other threshold multi-signature schemes, such as those presented by Boneh *et al.* [18] and Nick *et al.* [19], to evaluate the most appropriate alternatives.

#### REFERENCES

- [1] P. Micholia, M. Karaliopoulos, I. Koutsopoulos, L. Navarro, R. B. Vias, D. Boucas, M. Michalis, and P. Antoniadis, "Community Networks and Sustainability: A Survey of Perceptions, Practices, and Proposed Solutions," *IEEE Communications Surveys Tutorials*, vol. 20, no. 4, pp. 3581–3606, Mar. 2018.
- [2] E. C. Efstathiou, P. A. Frangoudis, and G. C. Polyzos, "Stimulating Participation in Wireless Community Networks," in *Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications*, Apr. 2006, pp. 1–13.
- [3] G. F. Marias, P. Georgiadis, D. Flitzanis, and K. Mandalas, "Cooperation enforcement schemes for MANETs: a survey," *Wireless Communications* and Mobile Computing, vol. 6, no. 3, pp. 319–332, May 2006.

<sup>8</sup>https://freifunk.net/

- [4] Y. Yoo, S. Ahn, and D. P. Agrawal, "A credit-payment scheme for packet forwarding fairness in mobile ad hoc networks," in *IEEE International Conference on Communications, 2005. ICC 2005. 2005*, vol. 5, Aug. 2005, pp. 3005–3009 Vol. 5.
- [5] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," Bitcoin.org, Tech. Rep., Oct. 2008.
- [6] Z. Zheng, S. Xie, H.-N. Dai, W. Chen, X. Chen, J. Weng, and M. Imran, "An overview on smart contracts: Challenges, advances and platforms," *Future Generation Computer Systems*, vol. 105, pp. 475–491, Apr. 2020.
- [7] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger – Istambul version cfa3f42," Ethereum & Parity, Tech. Rep., Jun. 2021.
- [8] G. Maxwell, A. Poelstra, Y. Seurin, and P. Wuille, "Simple Schnorr multi-signatures with applications to Bitcoin," *Designs, Codes and Crypto.*, vol. 87, no. 9, pp. 2139–2164, Sep. 2019.
- [9] M. Szydlo, "Merkle Tree Traversal in Log Space and Time," in Adv. in Cryptology - Eurocrypt 2004, C. Cachin and J. Camenisch, Eds. Berlin, Heidelberg: Springer, May 2004, pp. 541–554.
- [10] C. Machado and C. M. Westphall, "Blockchain incentivized data forwarding in MANETs: Strategies and challenges," Ad Hoc Networks, vol. 110, p. 102321, Jan. 2021.
- [11] E. Dimogerontakis, L. Navarro, M. Selimi, S. Mosquera, and F. Freitag, "MeshDapp – Blockchain-Enabled Sustainable Business Models for Networks," in *Economics of Grids, Clouds, Systems, and Services -GECON 2019*, vol. 11819. Cham: Springer, Nov. 2019, pp. 286–290.
- [12] J. Tremback, J. Kilpatrick, D. Simpler, and B. Wang, "Althea white paper v1.51," Available at: https://althea.net/whitepaper, May 2020.
- [13] R. Myers, "A lightweight protocol to incentivize mobile peer-topeer communication," Available at: https://global-mesh-labs.gitbook.io/ lot49/, Jun. 2019.
- [14] Y. He, H. Li, X. Cheng, Y. Liu, C. Yang, and L. Sun, "A Blockchain Based Truthful Incentive Mechanism for Distributed P2P Applications," *IEEE Access*, vol. 6, pp. 27 324–27 335, Apr. 2018.
- [15] I. Castro, A. Panda, B. Raghavan, S. Shenker, and S. Gorinsky, "Route Bazaar: Automatic Interdomain Contract Negotiation," in *15th Workshop* on Hot Topics in Operating Systems. Kartause Ittingen, Switzerland: USENIX, May 2015, pp. 1–7.
- [16] T. Trautmann and A. Burnell, "Routing Based Blockchain (US Patent 16/104,849 430)," Feb. 2020.
- [17] Q. Zhou, H. Huang, Z. Zheng, and J. Bian, "Solutions to scalability of blockchain: A survey," *IEEE Access*, vol. 8, pp. 16440–16455, Jan. 2020.
- [18] D. Boneh, M. Drijvers, and G. Neven, "Compact multi-signatures for smaller blockchains," in *Advances in Cryptology – ASIACRYPT 2018*, T. Peyrin and S. Galbraith, Eds. Cham: Springer, Oct. 2018, pp. 435– 464.
- [19] J. Nick, T. Ruffing, and Y. Seurin, "Musig2: Simple two-round schnorr multi-signatures," in Advances in Cryptology – CRYPTO 2021, T. Malkin and C. Peikert, Eds. Cham: Springer, Aug. 2021, pp. 189–221.