MeshPay: Resilient Offline Payment with Wireless Mesh Network

Quang Huy Do[†‡], Sara Tucci-Piergiovanni[†], Justice Owusu Agyemang[‡], Sami Souihi[†§]
[†]LISSI-TincNET Research Team, University of Paris-Est Creteil, France
[‡] Ejara Lab, France
[§] Université Paris-Saclay, CEA, List, F-91120, Palaiseau, France
Email: huy.quangdo@ejara.africa, sara.tucci@cea.fr, justice.agyemang@ejara.africa, sami.souihi@u-pec.fr

Abstract—**Mobile payment systems have become an essential part of modern digital services, supporting applications such as money transfer, mobile banking, and digital wallets. Despite their widespread adoption, these systems rely heavily on stable Internet connectivity, which remains unavailable or unreliable in many parts of the world. To address these challenges, we propose MeshPay, a decentralized offline payment framework that integrates wireless mesh networking with a blockchain-based settlement layer, enabling secure, real-time transaction settlement without Internet access. Blockchain technology provides a transparent and tamper-resistant ledger, reduces transaction costs and long process of validity by eliminating intermediaries. The proposed system is implemented and evaluated using Mininet-WiFi, demonstrating its ability to achieve up to 5000 transactions per second and keep latencies below 800 milliseconds with 20 authority nodes under ideal conditions. These results highlight the feasibility of mesh-based offline payment systems as a robust alternative for connectivity-constrained environments.**

Index Terms—**Wireless Mesh Networks, Offline Payment, Blockchain**

I. Introduction

According to [1], the global mobile payment market is projected to reach 7.58 trillion USD by 2027, at a Compound Annual Growth Rate of 29.0% during the forecast period. However, internet infrastructure is one the main reasons that challenges mobile payment for wide adoption. The internet infrastructure deployment in developing countries—particularly across Africa—remains severely constrained. Less than one-third (27%) of adults in Africa can have access mobile Internet services [2]. In bustling open-air markets, traders and shoppers often have internet capable phones in their pockets yet still pay with cash because the network is down or too slow to authorize transactions in real time. Additionally, internet shutdowns occured 21 times in 2024 in African countries, the highest number in a single year [3]. During shutdowns, mobile money agents and end users lose access to their wallets, resulting in the suspension of peer-to-peer transfers and merchant transactions.

Traditional mobile money systems further worsens these limitations, as they depend on centralized banking infrastructure for transaction settlement. These systems become inaccessible during network disruptions, with studies showing that internet shutdowns could have an estimated GDP impact of approximately $3 million per day of disruption, amounting to 0.4% of daily GDP [4]. Additionally, the complex security validation processes require extensive Know Your Customer (KYC) procedures that systematically exclude populations, particularly affecting the 850 million adults globally who do not have a form of ID, while mobile money providers typically charge significant fees ranging from 4.64% to 4.80% for cross border remittances [5]. In contrast, a blockchain based settlement layer enables peer-to-peer value transfer without relying on any single institution. When deployed over resilient or opportunistic communication channels such as mesh networks, it can maintain transaction functionality even amid Internet outages, while reducing costs and offering a transparent, tamper-resistant record system [6].

Thus, in this paper, we propose an off-chain method that leverages wireless mesh networking and blockchain technology to enable devices to form decentralized, peer-to-peer communication networks that support payments even under unreliable Internet connectivity. At a high level, the on-chain smart contract [7] governs the overall trust structure of the system: it registers a set of authority nodes (such as mobile money agents, merchants, or trusted community members) and manages the deposit funds that users lock before transacting offline. Once collateral is deposited, users can exchange tokens directly through the mesh network, where nearby authority nodes verify and collectively approve each transfer through a weighted voting protocol. And once the required quorum is collected, the resulting transaction is finalized locally and later reconciled on the blockchain. Our approach dynamically adjusts each authority's voting influence based on its historical performance within the mesh. As a result, quorum formation becomes both adaptive and locally verifiable, enabling the system to maintain secure and efficient transaction validation even under frequent network partitions.

Contributions. We make the following contributions:

- Propose a novel mobile payment architecture that uses wireless mesh networking to enable resilient, peer-to-peer transactions in environments with unreliable or limited internet connectivity.
- Introduce a decentralized weighted voting to ensure secure transaction validation within mesh networks where network partitions happen frequently, mitigating risks of fraud and double spending.
- Evaluate the proposed system under realistic connectivity constraints and user mobility scenarios.

Outline. The remainder of this paper is structured as follows. Section II provides background and related work. Section III shows the overview of the whole proposed system. Section IV details the proposed offline payment methodology. Section V presents the experimental results and analysis. Finally, Section VI concludes the paper with a discussion of key findings and future research directions.

II. Related Work

The work BRICK [8] introduces a novel off-chain payment channel design that remains secure under network asynchrony by employing a committee of external wardens to validate state closings, and making a call to on-chain contract only for settlements. This architecture demonstrates that by moving updates off-chain, transaction throughput and latency can be improved, while maintaining safety via on-chain anchoring. However, BRICK assumes connectivity sufficient for the committee to communicate and reach agreement—conditions that may not hold in a mobile mesh network where partitions, mobility and unreliable links are present.

[9] formalizes an off-chain execution model in which payments can proceed in parallel with fewer than f + 1 validations, by partitioning account balances among subsets of validators. This approach demonstrates that strong safety guarantees can be achieved even without full consensus, provided that final reconciliation is anchored on-chain. Our work builds on this principle but adapts it to decentralized wireless mesh environments, where validators (authority nodes) may experience intermittent connectivity. We introduce a weighted membership mechanism that dynamically adjusts validator influence based on performance, enabling consistent transaction validation.

It is also worth mentioning FastPay [10], which guided our design choice. FastPay employs a committee of authorities that pre-sign account certificates, enabling clients to gather a quorum of signatures within a single network round. This architecture achieves remarkable performance, delivering sub-100 ms settlement latency and throughput exceeding 80,000 transactions per second (TPS), while preserving strong Byzantine fault tolerance guarantees. We extend FastPay by replacing its static weighted quorum with a dynamic, decentralized weighted voting scheme inspired by Cabinet [11]. Specifically, nodes must lock a minimum stake to get admitted into the authority set. Once joined, each authority's voting power is updated according to an audit score that reflects performance (total number of validated transaction). Moreover, we remove infrastructure dependencies, we embed this enhanced FastPay atop a mesh network formed directly among mobile nodes. By employing decentralized weighted voting with a mesh overlay, our system sustains secure payments across multi hop paths during prolonged internet or cellular outages, overcoming the deployment barriers faced by prior offline payment solutions.

III. System Overview

Our solution is an *off-chain* payment system: payments are executed and validated locally over a wireless mesh, while a smart contract on the layer-1 blockchain provides immutable settlement and governance. The smart contract (i) registers and enforces authority membership, including epoched voting powers, and (ii) holds user deposits that back off-chain balances 1:1. Authorities maintain the off-chain ledger, fetch epoch snapshots from the contract, validate transfer orders over the mesh, and produce signed, epoch-tagged certificates. We introduce a weighted-membership mechanism: authority voting power is computed from performance counters and applied per epoch to make quorum formation adaptive and fair (later shown in Sec. III-D).

A. Participants

MeshPay involves three types of participants: (i) users, (ii) Authorities and (iii) Smart Contract.

Users are ordinary nodes that hold cryptographic key pairs and initiate payments. Users deposit funds into the smart contract and generate off-chain transfer requests. Users hold following information:

- Key pair (sk_u, pk_u).
- Committee configuration $\mathcal{C}$ (authority's public keys and weighted voting power).
- Local sequence counter $seq_u \in \mathbb{N}$ for off-chain orders.
- Weighted certificates $\mathsf{WCerts}_u = \{\langle name_i, \sigma_i, e, \hat{w}_i^e \rangle\}$ collected from authorities where:
  - $name_i$ denotes the **identifier (name)** of the i-th authority,
  - σ_i represents the **signature** generated by that authority, and
  - e is epoch identifier
  - $\hat{w}_i^e$ indicates its **weighted voting power** in epoch e.

Authorities is a fixed committee of validator nodes that maintains the off-chain ledger. To become authority, node must deposit a pre-defined fixed amount of token on-chain via a smart contract. Each authority has a key pair and holds the committee configuration. Honest authorities strictly follow the protocol; faulty ones may behave arbitrarily. The state of an authority consists of the following information:

- Key pair (sk_a, pk_a)
- Committee configuration $\mathcal{C}$ (authority's public keys and current epoch weighted voting power).
- Off-chain account table $\mathsf{Ledger}[u] = (\mathsf{bal}_u, \mathsf{next}_u)$ with current balance and next valid sequence of all Users.
- Pending orders $\mathsf{Pending}[u, s]$: validated user orders (sender u, sequence s) and local signature of all Users.
- Weighted certificates WCerts_u: quorum-certified transfers of all Users.
- Redeem log $\mathsf{Redeemed}[u]$: set or prefix of sequences already settled on-chain.
- Performance counters $\mathsf{tx\_count}_a$ (successful validation).
- Current normalized voting power $\hat{w}_a \in [0, 1]$, derived from performance (see Sec. III-D).

Smart Contract the smart contract serves as the immutable settlement point. Users deposit and later redeem their balances via this contract. Each Authority updates his voting power at the end of each epoch and fetch voting power from other authorities from the smart contract. The storage of the smart contract contains following information:

- Authority set $\mathcal{C}$ (public keys, locked funds, normalized voting power for each epoch) and weighted quorum threshold q for each epoch.
- Locked funds mapping $\mathsf{Locked}[u] \in \mathbb{N}$ for user deposits.
- Redeem log $\mathsf{Redeemed}[u]$ (e.g., highest redeemed sequence or a membership set) to prevent double-withdrawals.

B. Protocol Messages

The protocol distinguishes between on-chain transactions and off-chain orders exchanged over the mesh. Key message types include:

1) **Funding Transaction:** A user funds the system by sending an on-chain deposit transaction to the smart contract, locking a specified amount in $\mathsf{Locked}[u]$. Each authority periodically fetches the balances from smart contract, credits the user's balance in the local ledger $\mathsf{Ledger}[u]$.
2) **Transfer Order:** When a sender wishes to pay a recipient, they create a transfer order O with:
   - $sender(O)$: Sender's account
   - $recipient(O)$: Recipient's account
   - $amount(O)$: Transfer amount
   - $sequence(O)$: Monotonically increasing sequence number
   - Optional memo or metadata
   - σ_{sender}: Sender's signature over the fields

   The order O is broadcasted to all authorities by mesh network routing protocol, which validate balances and sequence numbers.
3) **Authority Response:** Each honest authority α that accepts O adds its weighted signature σ_α and replies. These signatures form the basis of a certificate.
4) **Transfer Certificate:** Once the sender collects at least two-thirds weighted voting power on O, he assembles them into a transfer certificate C. This certificate is proof that the payment is valid. The sender delivers C to the recipient.
5) **Confirmation Broadcast:** Either sender or recipient broadcasts C to all authorities. Upon receiving C, each authority debits the sender and credits the recipient. At this point, the transfer is final.
6) **Settlement Transaction:** Periodically, or when a user withdraws, a settlement transaction is sent on-chain including the relevant certificates. The smart contract verifies these certificates and updates the on-chain balances accordingly.

C. Wireless Mesh Network

The off-chain protocol operates atop a wireless mesh network, where each node, a user or authority, communicates with neighboring peers over wireless links such as Wifi or Bluetooth. The mesh network is self-organizing and infrastructure-independent, allowing devices to form a multi-hop topology that persists even when Internet access is unavailable. Each node functions both as a host and as a router, relaying packets for other participants. Established routing protocols such as OLSR [12] are employed to dynamically determine optimal paths to the destination, ensuring efficient data dissemination despite network variability.

The wireless mesh network also inherently introduces asynchronous communication characteristics, including variable delays, packet loss, and temporary disconnections. To tolerate these uncertainties, the protocol assumes eventual delivery for honest nodes. As long as a quorum of two-thirds weighted signatures reached for a transfer order, the system can make progress without requiring synchronized connectivity. This quorum-based mechanism forms the backbone of trust in the network, enabling payment finalization even when the Internet is intermittently unavailable.

D. Weighted Voting

We adopt a performance-based weighted voting scheme where each authority's raw weight is derived from its recent validation performance, e.g.

$$w_i = \max\big(\mathsf{tx}_i,\, 0\big),$$

with tx_i the authority's transfer-success counter. To ensure off-chain certificates are verifiable under asynchronous mesh conditions, all weight vectors are published on-chain. Formally, for epoch e the contract holds

$$\mathbf{w}^e = (w_1^e, \ldots, w_n^e)$$

and normalized voting powers

$$\hat{w}_i^e = \frac{w_i^e}{\sum_j w_j^e}, \quad \sum_i \hat{w}_i^e = 1.$$

and any change to weights becomes effective only at a deterministic activation point (epoch boundary). Authorities include the epoch identifier e and their epoch weight $\hat{w}_i^e$ inside each signature over a transfer order, and clients only aggregate signatures that reference the same epoch. A set of authorities S is said to reach quorum for epoch e if

$$\sum_{i \in S} \hat{w}_i^e \;\geq\; q^e.$$

Authorities include the epoch identifier e and their epoch weight $\hat{w}_i^e$ inside each signature over a transfer order. Clients accumulate weighted responses from authorities that reference the same epoch e and detect quorum reach q^e at which point the transaction can be said to be finalized.

At the end of each epoch, authorities may request an on-chain voting power update by submitting a transaction that carries the proposed raw scores (performance counters $\mathsf{tx\_count}_i$) together with a signed attestation set from authorities; the smart contract validates the attestation signatures and records the proposed vector $\mathbf{w}^{\text{next}}$. The contract does not apply $\mathbf{w}^{\text{next}}$ immediately: instead it schedules the vector to become active at a deterministic activation block or timestamp (the next epoch boundary), and until that activation point the contract continues to expose the current snapshot $\mathbf{w}^e$.

IV. Methodology

A. Depositing Funds

As illustrated in Figure 1, User wants to use MeshPay and become authorities can initiate a deposit by invoking the on chain smart contract on the main blockchain. This transaction is called a funding transaction, and includes the recipient address for the funds and the amount of value to transfer. The contract locks the specified amount and returns a signed deposit receipt that binds the value to the user's public key and the current mesh-committee epoch. The User forwards this receipt to a nearby authority, which verifies the on chain signature and records the corresponding balance in the local mesh ledger. Once at least a quorum of Authorities attest to the receipt, the funds become immediately spendable inside the offline mesh, guaranteeing that every token circulating off chain is fully collateralized on the main chain smart contract.

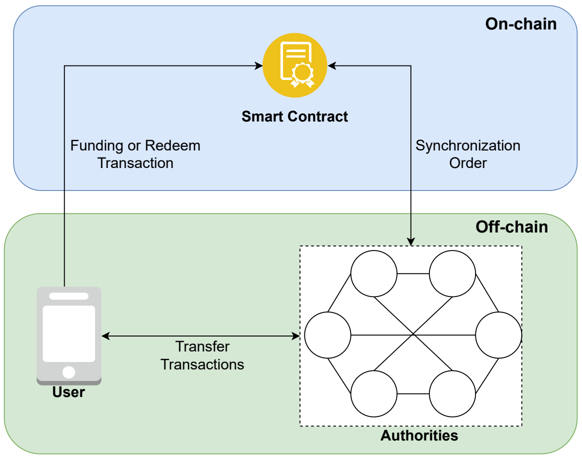


Fig. 1: Transfer of funds between Blockchain and Protocol

B. Transferring Funds

Figure 2 depicts an offline payment between two Users. The Sender creates a transfer order O, sign and broadcasts it to the neighbor nodes. The Sender also stores the transfer order O to the $Pending_{sender}$ list. The sender periodically broadcasts the orders to all other neighbor nodes if there is still pending orders inside $Pending_{sender}$. When Authority receives the order, it independently checks (i) the signature is valid, (ii) no previous transfer is pending, (iii) the amount is positive, (iv) the sequence number matches the expected next one, and (iv) the balance is sufficient. And if valid, authorities adds its weighted signature to the order, mark it as pending order, send back a signed transfer order. The algorithm to handle transfer order is presented in Algorithm 1.

After collecting greater or equal two-thirds weighted signatures from Authorities, the Sender can form a transfer certificate. With the certificate, the Sender can forward it to the Receiver as proof of payment. To conclude the order, the Sender or Recipient broadcast the transfer certificate (called confirmation order) to all authorities. Upon reception of a confirmation order for the current sequence number, each authority (i) checks that a quorum of signatures was reached,

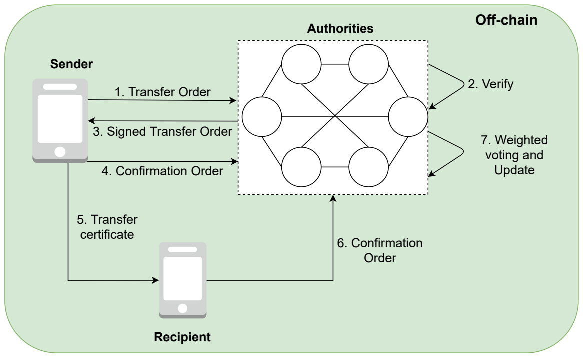


Fig. 2: Transfer of funds between Users

(ii) decreases the balance of the sender, (iii) increments the sequence number to ensure 'deliver once' semantics, and (iv) sets the pending order to None. Each authority also (v) adds the certificate to the list confirmed, (vi) increases the balance of the recipient account asynchronously and finally (vii) adds one to the performance counters tx_count. The authority algorithm to handle confirmation orders is presented in Algorithm 2.

Algorithm 1 Authority: Handle Transfer Order

```
1: function HANDLE_TRANSFER_ORDER($\alpha, O$)
2:     ensure $O$.has_valid_signature()
3:     $account \leftarrow$ Ledger[$O$.sender]
4:     if $account$ is None then
5:         return REJECT("acct-not-found")
6:     end if
7:     $p \leftarrow$ Pending[$O$.sender, next$_u$]
8:     if $p$ exists then
9:         if $p$.transfer $==$ $O$ then
10:            return ACCEPT
11:        else
12:            return REJECT("conflict")
13:        end if
14:    end if
15:    ensure next$_u$ $==$ $O$.sequence
16:    ensure $account$.balance $\geq$ $O$.amount
17:    ensure $O$.amount $> 0$
18:    $\sigma_\alpha \leftarrow$ Sign($sk_\alpha, O$)
19:    Pending[$O$.sender, $O$.sequence] $\leftarrow (O,\ \sigma_\alpha)$
20:    return ACCEPT($O,\ \sigma_\alpha$)
21: end function
```

Algorithm 2 Authority: Handle Confirmation Order

```
1: function HANDLE_CONFIRMATION_ORDER($\alpha, C$)
2:     ensure $C$.is_valid($\mathcal{C}, q$)
3:     $O \leftarrow$ value($C$)
4:     $sender\_account \leftarrow$ Ledger[$O$.sender]
5:     if $sender\_account$ is None then
6:         $sender\_account \leftarrow$ (bal $= 0$, next $= 0$)
7:     end if
8:     if $sender\_account$.next $>$ $O$.sequence then
9:         return ACCEPT("already-confirmed")
10:    end if
11:    ensure $sender\_account$.next $==$ $O$.sequence
12:    ensure $sender\_account$.bal $\geq$ $O$.amount
13:    $sender\_account$.bal $\leftarrow$ $sender\_acct$.bal $-$ $O$.amount
14:    $sender\_account$.next $\leftarrow$ $sender\_acct$.next $+ 1$
15:    Pending[$O$.sender, $O$.sequence] $\leftarrow$ None
16:    WCerts[$O$.sender, $O$.sequence] $\leftarrow C$
17:    $recipient\_account \leftarrow$ Ledger[$O$.recipient]
18:    if $recipient\_account$ is None then
19:        $recipient\_account \leftarrow$ (bal $= 0$, next $= 0$)
20:    end if
21:    $recipient\_account$.bal $\leftarrow$ $recipient\_account$.bal $+$ $O$.amount
22:    tx_count $\leftarrow$ tx_count $+ 1$
23:    return ACCEPT("confirmed")
24: end function
```

C. Withdraw Funds

When Internet connectivity resumes, any authorities can aggregate their off-chain ledger view and submit a quorum-certified update to the smart contract. The contract verifies the weighted committee signatures and updates its state, ensuring that every off-chain balance remains 1:1 backed by locked funds on the primary chain.

As illustrated in Figure 1, an user withdraws by issuing a direct on-chain request to the contract, specifying a destination and amount. The contract checks admissibility against its reconciled state: $0 \le \mathit{amount} \le \mathsf{Locked}[u] - \mathsf{Redeemed}[u]$, updates the redeem log to prevent double-withdrawals, and transfers the requested value to the user's address.

Post-withdrawal, authorities continuously fetch the smart contract to update per-account state and detect divergence. In detail, each authority periodically fetches the on-chain balance and the authoritative per-account sequence counter $seq_{\text{chain}}[u]$ and cross-checks it against its local next-sequence $seq_{\text{off}}[u]$. Authorities enforce $bal_{\text{off}}[u] = bal_{\text{chain}}[u]$ and that the locally available balance matches the on-chain collateral.

V. Evaluation

A. Experimental Setup

We perform the mesh network emulations on an enhanced version of **Mininet-WiFi** [13], modified to support our experimental requirements. The emulations are experimented over an area of 1000 × 1000 m^2. We instantiate 50 geo-distributed Users within the area and Users submit transactions in an open loop model, at a fixed rate. Authorities nodes run from 5 to 20. Both Authority and User node are equipped with mesh-enabled wireless interfaces. The nodes communicate over a single mesh network identified by the SSID `meshNet`. Each User sends one transfer order at 10 s interval time. Propagation loss model `logDistance` to simulate the limited communication range typical of Bluetooth devices. The mobility model used is a realistic version of Random Direction [14] to mimic the movement of mobile user in real world. A summary of all emulation parameters is shown in Table I. The implementation, bench marking scripts, and measurements data to enable reproducibility.[1]

B. Results And Analysis

For our emulation, latency is the interval time from the moment User submits a transfer order to when it receives two-thirds weighted signatures, and throughput refers to the number of transactions committed per second. In our experiment, we increase the load of transactions sent to the protocol, and record the throughput and latency of transfer orders.

To show the performance of our protocol, we compare the system's performance when using the proposed weighted voting mechanism called MeshPay-Weighted against a baseline configuration called MeshPay-Normal. In MeshPay-Normal, we employ a standard Pratical Byzantine Fault Tolerant (PBFT) voting protocol [15] where transactions reach finality after receiving 2f+1 signed transfer orders from authorities. The comparison are conducted in 2 cases: one is ideal conditions where there is no faulty nodes, and the other is with faults where we have faulty or Byzantine nodes.

[1]https://github.com/HuyPHD2024-2027/smart-pay

Parameter	**Value**
Area	1000×1000 m
Number of users	50
Number of authorities	5-20
Mesh SSID	meshNet
Transfer order generating time	10 s
Propagation loss model	logDistance
Mobility model	Random Direction
Transmission range	20 m
Node speed	$0-20$ m/s

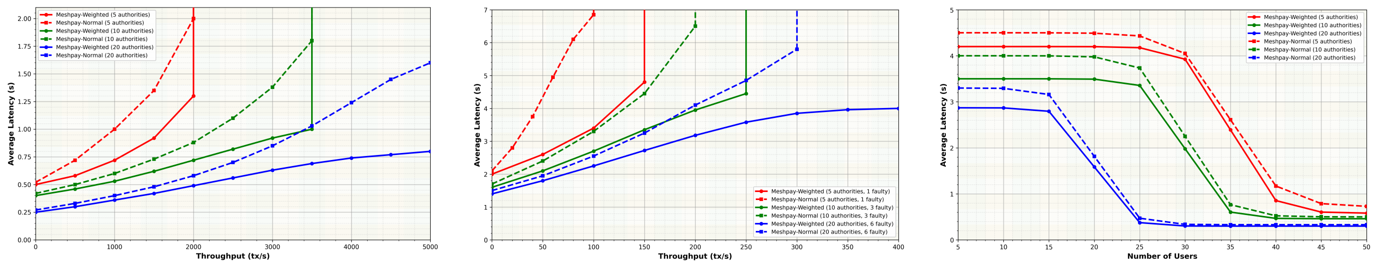
TABLE I: Emulation Parameters

1) Performance under ideal conditions: We evaluate MeshPay under an ideal wireless mesh network in which no authority behaves maliciously or fails. Results are shown in Figure 3a.

The results show that increasing the number of authorities proportionally improves throughput. Under ideal conditions (Figure 3a), both configurations achieve comparable peak throughput—1,500, 3,000, and 5,000 tx/s for 5, 10, and 20 authorities, respectively. However, MeshPay-Weighted achieves significantly lower latency, reducing average confirmation time by 35–50%. Specifically, at peak throughput, for authority size of 5, 10 and 20 authorities the average commit latencies for the MeshPay-Normal are about 2.0 s, 1.8 s and 1.6 s, respectively; the MeshPay-Weighted achieves lower latencies of approximately 1.3 s, 1.0 s and 0.8 s for the same authority sizes. This improvement arises from its adaptive quorum formation: finality can be reached once the cumulative weight exceeds two-thirds, without requiring all authorities to respond. In practice this reduces the number of message rounds and the number of hops required to complete a transfer order. The empirical results thus demonstrate that weighted membership preserves throughput while lowering latency under ideal mesh conditions.

2) Performance under faults: Figure 3b depicts the performance of two systems when we suffers 1,3 and 6 Byzantine nodes for 5,10 and 20 authority size (the maximum that can be tolerated for this authority size in PBFT voting protocol).

Under faulty conditions, MeshPay-Weighted continues to outperform MeshPay-Normal, maintaining higher throughput and lower latency despite the presence of Byzantine nodes. In detail, MeshPay-Weighted reaches maximum throughput at 150, 250 and 400 tx/s while MeshPay-Normal only achieves 100, 200 and 300 tx/s when authority size is 5, 10 and 20 respectively. Additionally, MeshPay-Weighted helps orders reach finality faster than normal voting protocol. MeshPay-Weighted records a latency of 4.6s and 4.4s and 4s when running with peak throughput. However these numbers for MeshPay-Normal are 6.85s, 6.5s and 5.8s. We observe that de-



(a) Throughput-latency under ideal conditions.

(b) Throughput-latency with Byzantine nodes.

(c) Latency with number of user nodes.

Fig. 3: Performance of MeshPay under different conditions.

spite the presence of faulty validators, the MeshPay-Weighted protocol will gradually reduce the voting power of the faulty authority while increase the voting power of correct ones which help system to process transactions. Concretely, when a subset of authorities are faulty or Byzantine, Meshpay-Normal must wait for two-thirds of signatures and thus exhibits increasing latency; by contrast, the weighted scheme can reach quorum using a smaller set authorities, which reduce the latency and handle more throughput.

3) Impact of Network Density on Latency Performance: Figure 3c illustrates the relationship between network density and transaction latency for both MeshPay-Weighted and MeshPay-Normal voting mechanisms when number of users running from 5 to 50 under ideal conditions. At low user densities, both voting mechanisms maintain relatively high and stable latency regardless of authority count, attributed to insufficient network coverage where sparse node distribution necessitates prolonged route discovery processes. With limited number of neighbors, nodes must frequently wait to establish forwarding paths to destination authorities, resulting in high end-to-end latency.

Beyond critical density thresholds, the network transitions to a well-connected network where nodes maintain stable multi-hop forwarding paths without route discovery delays. Specifically the critical number of users are 35, 30 and 20 for system with 5, 10 and 20 authorities respectively. Beyond these critical points, the latency gap between the weighted voting and standard BFT voting protocols becomes negligible, as the network density ensures that transfer orders can be propagated and confirmed within a single message round, eliminating the need for retransmissions or delayed relays.

VI. Conclusion

Our early results show that MeshPay, an offline payment protocol over a mesh network can reach 5000 tps and clear retail payments in less than 800 milliseconds even when internet connectivity is unavailable. These results, however, reflect best-case laboratory conditions: traffic is evenly distributed across nodes and authority load is still modest. Future work will therefore (i) validate the protocol on real mobile devices, (ii) integrate privacy primitives such as Coconut [16] credentials and confidential balances, and (iii) design a fee model that discourages abuse while incentivizing honest authorities. Addressing these challenges will transform the current protocol into a deployable, privacy-preserving and economically sustainable offline payment system.

References

[1] M. K. Regragui, "The african mobile wallets: an empirical analysis of the services and the anticipated trends," 2022.

[2] O. Mothobi and K. Kebotsamang, "The impact of network coverage on adoption of fintech and financial inclusion in sub-saharan africa," *Journal of Economic Structures*, vol. 13, no. 1, p. 5, 2024.

[3] digWatch, "Internet shutdowns surge in africa despite human rights concerns," 2025.

[4] M. Patnam and W. Yao, "The real effects of mobile money: Evidence from a large-scale fintech expansion," 2020.

[5] C. Lowe, "The commercial sustainability of mobile money providers in interoperability initiatives," 2024, accessed: 2024-02.

[6] F. d. S. Momo and A. Behr, "Blockchain: Effects in transactions costs from information governance," *BAR-Brazilian Administration Review*, vol. 18, no. spe, p. e200047, 2021.

[7] B. K. Mohanta, S. S. Panda, and D. Jena, "An overview of smart contract and use cases in blockchain technology," in *2018 9th international conference on computing, communication and networking technologies (ICCCNT)*. IEEE, 2018, pp. 1–4.

[8] G. Avarikioti, E. K. Kogias, R. Wattenhofer, and D. Zindros, "Brick: Asynchronous payment channels," *arXiv preprint arXiv:1905.11360*, 2019.

[9] R. Bazzi and S. Tucci-Piergiovanni, "The fractional spending problem: Executing payment transactions in parallel with less than f+ 1 validations," in *Proceedings of the 43rd ACM Symposium on Principles of Distributed Computing*, 2024, pp. 295–305.

[10] M. Baudet, G. Danezis, and A. Sonnino, "Fastpay: High-performance byzantine fault tolerant settlement," in *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, 2020, pp. 163–177.

[11] G. Zhang, S. Zhang, M. Bachras, Y. Zhang, and H.-A. Jacobsen, "Cabinet: Dynamically weighted consensus made fast," *arXiv preprint arXiv:2503.08914*, 2025.

[12] T. Clausen and P. Jacquet, "Optimized link state routing protocol (olsr)," Tech. Rep., 2003.

[13] Q. H. Do, T. Abreu, B. Kusi, N. C. Diop, and S. Souihi, "An efficient epidemic routing protocol with reinforcement learning algorithm in opportunistic networks," in *ICC 2025-IEEE International Conference on Communications*. IEEE, 2025, pp. 3497–3502.

[14] B. Gloss, M. Scharf, and D. Neubauer, "A more realistic random direction mobility model," *TD (05)*, vol. 52, pp. 13–14, 2005.

[15] M. Castro, B. Liskov *et al.*, "Practical byzantine fault tolerance," in *OsDI*, vol. 99, no. 1999, 1999, pp. 173–186.

[16] A. Sonnino, M. Al-Bassam, S. Bano, S. Meiklejohn, and G. Danezis, "Coconut: Threshold issuance selective disclosure credentials with applications to distributed ledgers. arxiv 2018," *arXiv preprint arXiv:1802.07344*, 2018.