Offline Bitcoin Transactions via Incentivised BLE Mesh Networks

Arash Ershadi Esmaeilabadi
arash.ershadi@hotmail.com

Abstract

We propose a protocol for enabling Bitcoin transactions in environments without direct Internet access, by leveraging Bluetooth Low Energy (BLE) mesh networks with economic incentives. In this system, nodes lacking Internet connectivity can broadcast their transactions through peer devices ("beacons") that do have connectivity. Beacons periodically issue cryptographically signed *proof-of-internet* messages, attesting to the current blockchain state. Senders verify these proofs and construct transactions that include a small relay fee output paying the beacon. The beacon relays the transaction to the Bitcoin network only if the fee is present, aligning incentives. Our protocol requires no changes to the Bitcoin consensus rules and can be implemented using existing SPV wallets and BLE mesh technology. We detail the system architecture, transaction workflow, and security analysis, showing how offline Bitcoin payments can be achieved with this approach.

1 Introduction

Bitcoin provides a decentralised, peer-to-peer electronic cash system [1]. Transactions and blocks are propagated by a global network of nodes, and consensus is secured by proof-of-work. A key assumption is that participants have continuous Internet connectivity to broadcast and receive their transactions. However, in many scenarios (e.g., natural disasters, remote areas, or censorship environments), Internet access may be limited or absent. Without connectivity, users cannot broadcast transactions to the Bitcoin network and thus cannot spend their coins.

Existing solutions only partially address this challenge. Layer-2 approaches like the Lightning Network enable rapid off-chain payments [2], but still require pre-funded payment channels and periodic connectivity for settlement. Other methods (e.g., payment vouchers or prepaid tokens) rely on trusted intermediaries or specialised hardware. More recently, wireless mesh networks have been explored to reduce reliance on centralised networks. For example, Kurt *et al.* introduce LNMesh, which enables offline Lightning payments over local

Bluetooth/WiFi mesh [3]. While LNMesh demonstrates promising results, it inherits Lightning's security requirements: nodes must monitor the blockchain to detect channel breaches and respond with penalty transactions within timeouts. These requirements often necessitate watchtower services and limit applicability in prolonged offline scenarios.

Our approach operates directly at the Bitcoin blockchain layer, requiring no channel setup, watchtower dependencies, or protocol changes. This paper presents *Offline Bitcoin Transactions via Incentivised BLE Mesh Networks*. In our protocol, wallets can broadcast Bitcoin payments offline using standard on-chain transactions that settle immediately upon reaching the network. Users form a BLE mesh network, within which some nodes act as *beacons* with Internet access. Each beacon periodically broadcasts a signed *proof-of-internet* containing the latest block header and its address. Upon receiving this proof, a sender constructs a transaction that includes an additional output paying a small relay fee to the beacon's address. The beacon then forwards the transaction to the Bitcoin network only if this fee output is present. This incentive mechanism ensures beacons are compensated for relaying while avoiding the security complexities of off-chain payment channels.

The remainder of the paper is structured as follows: Section 2 describes the system design, Section 3 details the transaction workflow, Section 4 analyses security considerations, Section 5 discusses implementation aspects, and Section 6 outlines future work. We conclude in Section 7.

2 Architecture

The system comprises three types of nodes: *senders* (wallets with a transaction to broadcast, but no Internet), *beacons* (nodes with Internet connectivity that assist in relaying), and *relays* (nodes without Internet who forward messages within the mesh). All nodes participate in a Bluetooth Low Energy (BLE) mesh network, which supports many-to-many communication via a managed flooding protocol with a limited time-to-live (TTL) hop count [4]. We assume the mesh is connected and that messages eventually reach all nodes if relayed.

The mesh supports two classes of messages:

- **Proof-of-Internet (PoI) messages:** Each beacon periodically broadcasts a PoI containing the current chain tip's block header H and its own address A. Formally, the beacon sends $\{H \parallel A, \sigma\}$ where $\sigma = \text{Sign}_{\text{sk}_A}(H \parallel A)$ is a digital signature on the data using the beacon's private key. Receiving nodes verify that H is a valid block header (correct PoW) and that σ is valid for address A. Valid PoIs are rebroadcast until every node in the mesh has the beacon's recent proof.
- **Transaction messages:** Senders create Bitcoin transactions T and broadcast them in the mesh. Each transaction includes standard inputs and outputs (recipient and change) *plus* one extra output of value f paying to a beacon's address A_i (or an OP_RETURN encoding A_i for zero-fee

beacons). After signing T, the sender floods it into the mesh. Intermediate nodes forward all incoming transactions. When a beacon i receives a transaction, it checks whether T contains the expected fee output to A_i. If and only if the fee is present, the beacon broadcasts T to the Bitcoin network and confirms it has been relayed. Otherwise the beacon ignores it.

Incentives are embedded directly in the protocol. Each transaction variant pays the beacon a small relay fee via the extra output. Because the beacon will only broadcast transactions that pay it, maliciously omitting the fee yields no benefit. Rational beacons are thus motivated to propagate the transaction into the Bitcoin network. If a beacon advertises "free" forwarding, the sender still includes its identity (e.g. in an OP_RETURN) so the beacon can recognise its transactions. In sum, no trusting is required: economic incentive and protocol rules enforce correct behaviour.

3 Transaction Workflow

The end-to-end process of making an offline payment proceeds as follows:

1. **Wallet synchronisation (online).** When a node has Internet access, it behaves like a standard SPV wallet: it downloads all block headers and queries a server (e.g. Electrum) for its wallet's UTXOs, using Merkle proofs to validate their inclusion [1, 5]. This initial sync ensures the wallet knows the current chain tip and its own balance.
2. **Offline PoI collection.** When the sender is offline, it listens for PoI broadcasts from nearby beacons. For each PoI it receives, it checks:
   - *Timestamp freshness:* The block header's timestamp t_H should be recent (e.g. within 1 hour of the local clock) to guard against replaying stale proofs.
   - *Difficulty plausibility:* Let the wallet's last known header be H_0 (timestamp t_{H_0}, difficulty D_0) and the PoI header be H (difficulty D_H). By Bitcoin's rules, network difficulty adjusts only at discrete retarget intervals and cannot be made arbitrarily small without mining work; the sender checks that D_H is not implausibly low given the time elapsed since t_{H_0} and the number of retarget intervals passed [1]. This prevents an attacker from presenting a fake low-difficulty chain.
   - *Proof-of-work validity:* Verify that H's hash meets the target, proving it is a mined block.
   - *Signature validity:* Verify σ against address A. This proves the beacon indeed created the PoI.

   PoIs that pass all checks are accepted. The sender typically hears multiple beacons, gathering proofs from each.

3. **Transaction creation.** With valid proofs in hand, the sender constructs its Bitcoin transaction. The inputs and outputs to recipient and change are fixed. For each beacon A_i from which a PoI was received, the sender creates a transaction variant T_i that includes an extra output of value f sending to A_i (or an OP_RETURN for zero-value relay). The sender signs each T_i and prepares to broadcast them.
4. **Mesh propagation.** The sender broadcasts all signed T_i into the mesh. Relays simply forward them. Each beacon i filters incoming transactions and only rebroadcasts the variant T_i that contains its fee output to A_i. Other variants are ignored by that beacon. In this way, each beacon acts on the single transaction that compensates it.
5. **Broadcast and confirmation.** Any beacon receiving a valid T_i (with its fee) will then broadcast it to the Bitcoin P2P network and wait for miners. Eventually, one of the T_i will be mined into a block, or if none does (e.g. double-spends are resolved), the wallet will find out upon the next sync. Once a beacon's transaction is confirmed, it can propagate a Merkle inclusion proof (via the mesh) back to the sender, giving the sender confidence that the payment is on-chain.

This workflow inherently creates multiple conflicting transactions (different T_i with the same inputs). Bitcoin's consensus resolves these: at most one variant can confirm. The wallet treats all other variants as unconfirmed double-spends. Since all variants differ only by a tiny fee output, miners will choose whichever yields the highest total fee; in practice, even a small extra fee suffices. Senders assume rational behaviour: paying multiple small fees (once per beacon) has limited cost and provides redundancy. **Critically**, the redundancy and potential conflict are contained *within* the local BLE mesh. The global Bitcoin P2P network is protected from spam by its standard mempool policy: nodes reject and do not relay transactions that conflict with ones already in their mempool. Therefore, the *first* beacon to relay its version T_i to the Bitcoin network will have it propagate normally. Any subsequent beacon attempting to relay a different variant T_j (spending the same inputs) to any Bitcoin node will find that the node already has a conflicting transaction and will reject T_j, preventing its further propagation. This ensures that the protocol provides sender-side redundancy without imposing additional bandwidth costs on the Bitcoin network.

4 Security Considerations

We consider possible attacks and mitigations:

4.1 Forgery of Proof-of-Internet

A malicious node might try to forge PoIs (for example, fabricating a beacon or inventing a new header). However, forging a valid PoI requires solving Bitcoin's proof-of-work or breaking digital signatures. Since PoIs contain real block headers, an attacker would need to find a block header H with valid hash and difficulty. In effect, this means re-mining a valid block, which is infeasible without significant hash power. Even if an attacker replays an old header, the timestamp/difficulty checks will reject it as stale. If an attacker intercepts a genuine PoI and changes the beacon address to its own, that transaction version will fail at the real beacon (no fee to original) and also offer no advantage to the attacker (they must still mine a block to exploit it). Thus forgery is not practical.

4.2 Proof Replay Attacks

Replay of old PoIs could mislead a sender. We require PoI freshness checks (e.g. a 1-hour window). Bitcoin occasionally has blocks taking multiple hours; in such cases, the next honest PoI (when the block finally arrives) will override an older one. An attacker trying to replay a very old PoI would need to meet the freshness window or re-mine the block at current difficulty, which is infeasible. In summary, with regular PoI updates, stale replays will be ignored.

4.3 Mesh Partition and Sybil Attacks

If an adversary could partition the mesh or flood it with Sybil beacons, it could show different proofs to different segments. For instance, if one attacker is the only connector to a segment, it could present its own PoIs there. This is analogous to the classic network-partition problem and cannot be fully prevented without a global link; see the Byzantine coordination literature [6]. However, our protocol does not give the attacker any undue advantage beyond what Bitcoin's own network structure already suffers under partitioning. As long as some honest beacons reach each user eventually, those users will have correct proofs. Using multiple beacons and flood-based propagation also reduces the risk that one adversarial node can isolate a segment. Ultimately, like any distributed system, we rely on having at least one honest path to the real Bitcoin network.

4.4 Double-Spend and RBF

Our protocol deliberately constructs conflicting transactions T_i. Bitcoin's consensus rules treat them as double-spends. Only one can confirm; the others will be rejected. This is safe because the sender's wallet can detect which UTXO got spent. We recommend that senders prefer broadcasting enough fee (miner fee plus relay fee) to ensure confirmation of some variant. Standard SPV and Electrum checks upon reconnect will show that the transaction inputs were spent in exactly one way. This is no more risky than usual wallet behaviour (similar to replace-by-fee scenarios) [5].

4.5 Incentive Analysis and Nash Equilibrium

The protocol's security relies on economic incentives. A game-theoretic analysis shows that the desired behaviour is a Nash Equilibrium. The players are the

Sender (strategies: include fee or not) and a *Beacon* (strategies: relay or ignore). Let V_s be the value to the sender of a confirmed transaction, f the relay fee, and C_b the beacon's cost to relay ($f > C_b$). The payoff matrix is:

	Beacon: Relay	**Beacon: Ignore**
Sender: Pay Fee	$(V_s - f,\ f - C_b)$	$(-f,\ 0)$
Sender: No Fee	$(V_s,\ -C_b)$	$(0,\ 0)$

The strategy pair (**Pay Fee**, **Relay**) is a Nash Equilibrium. Unilaterally, the sender would not deviate to *No Fee* (as the beacon would then ignore, yielding 0, which is worse than $V_s - f$). The beacon would not deviate to *Ignore* (as it would forfeit the profit $f - C_b$). No other strategy pair is stable. Therefore, rational actors are incentivized to follow the protocol, making it self-enforcing. The presence of multiple beacons reinforces this equilibrium through competition.

4.6 Cryptographic Guarantees

The security of the protocol relies only on Bitcoin's existing cryptography. Beacons sign PoIs with their private keys (ECDSA or Schnorr over secp256k1); recipients verify via the published address. Block headers provide hash-based PoW. Transaction signatures and Merkle proofs are handled by the SPV wallet as usual. We have introduced no new cryptographic primitives beyond what Bitcoin uses, so all standard guarantees (unforgeability of signatures, immutability of PoW) apply directly.

5 Implementation Considerations

5.1 Hardware and Software

This protocol can be implemented on off-the-shelf devices (smartphones or IoT) with BLE radios. Modern smartphones support BLE advertisements and scanning, and BLE mesh libraries (or Bluetooth Mesh Profile implementations) can carry custom data. The wallet software is a standard SPV (Electrum-like) client, storing block headers (tens of MB) and UTXO data for the wallet's keys. We assume beacons have Internet via cellular or WiFi and run or access a full node or public server to broadcast transactions.

On Android/iOS, an application could use native BLE APIs or existing mesh SDKs to exchange PoIs and transactions. Beacons might be ordinary phones or dedicated gateways (e.g. Raspberry Pi with BLE and 4G). The overhead is modest: PoI messages might be on the order of a block header plus signature, and transactions are typically under 1 KB. Constant BLE advertising and scanning will consume battery, but infrequent PoIs (minutes apart) mitigate this.

5.2 BLE Mesh Specifics

Bluetooth Mesh (as standardised by the Bluetooth SIG) supports many-to-many messaging with managed flooding, TTLs, and sequence numbers for de-duplication [4]. Payload sizes are limited and messages may need to be chunked or efficiently encoded; in practice, PoIs and transactions can be segmented across multiple mesh PDUs. Link-layer or application-layer encryption is available in Bluetooth Mesh but is orthogonal to the protocol described here.

Since BLE bandwidth is limited, we minimise chatter. Beacons should not flood PoIs too frequently (once per new block is sufficient). Transaction propagation can use short TTL (just enough hops to reach nearby beacons) instead of global flood to reduce load. A routing overlay that biases forwarding toward nodes likely to have Internet access can further reduce mesh traffic.

5.3 Scalability

On the Bitcoin side, scalability is unaffected: each payment still results in at most one on-chain transaction. The mesh network scales similarly to other BLE mesh deployments: it can handle local networks of dozens to hundreds of nodes depending on physical topology and radio environment. The main limits are radio range, airtime contention, and battery. In dense deployments, standard mesh de-duplication and backoff reduce collisions.

6 Discussion and Future Work

This protocol makes some assumptions: rational, fee-seeking beacons and reasonably synchronised clocks. If a beacon goes offline or becomes malicious mid-transaction, other beacons' fees still allow propagation (the sender broadcasts to all known beacons). If the sender remains offline indefinitely, it cannot learn final confirmation (as any SPV wallet); this is inherent in the offline model.

Potential improvements include adaptive fee selection and targeted routing. A dynamic fee schedule could adjust f based on beacon response times or competition. Optimising broadcast to target likely-beacon nodes first (e.g. based on signal strength or recent PoI freshness) could speed confirmation. Privacy is a concern: broadcasting a transaction in cleartext tags the sender to the mesh. Future work might explore mixing, onion-like forwarding within mesh hops, or other privacy-preserving overlays, but the fundamental limitations imposed by the Two Generals / coordination problems remain [6].

While we focused on Bitcoin SPV, similar ideas could apply to other UTXO blockchains. Integrating this with higher-layer protocols (e.g. atomic swaps or cross-chain settlement) is another research direction.

7 Conclusion

We have presented a full protocol for broadcasting Bitcoin transactions without direct Internet, using an incentivised BLE mesh network. By having beacons provide signed proofs of Internet connectivity and embedding relay fees in the transaction, our scheme creates a self-enforcing overlay: senders pay beacons to forward their transactions, and beacons only forward transactions that pay them. Bitcoin's own consensus handles double-spending resolution automatically. The design requires no changes to Bitcoin's consensus rules and uses well-understood SPV and BLE technologies. This work demonstrates a practical approach to extending Bitcoin's reach into offline environments, with potential applications in disaster relief, rural areas, and censorship resistance.

References

[1] S. Nakamoto, *Bitcoin: A Peer-to-Peer Electronic Cash System*, 2008. Available: `https://bitcoin.org/bitcoin.pdf`.

[2] J. Poon and T. Dryja, "The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments," DRAFT (version 0.5.9.2), Jan 2016. Available: `https://lightning.network/lightning-network-paper.pdf`.

[3] A. Kurt, E. Yildiz, and O. Gurbuz, "LNMesh: Enabling Bitcoin Lightning Payments over Wireless Mesh Networks," in *Proc. IEEE INFOCOM*, 2023.

[4] Bluetooth SIG, "Mesh Profile Specification 1.0.1," Available: `https://www.bluetooth.com/specifications/specs/mesh-profile-1-0-1/`.

[5] Electrum Protocol Documentation, ReadTheDocs. Available: `https://electrum-protocol.readthedocs.io/en/latest/`.

[6] L. Lamport, R. Shostak, and M. Pease, "The Byzantine Generals Problem," *ACM Trans. Program. Lang. Syst.*, vol. 4, no. 3, pp. 382–401, 1982. Available: `https://lamport.azurewebsites.net/pubs/byz.pdf`.