CS-740 Blockchain Design Architecture

A Report on the Project Entitled

Leveraging Public-Private Blockchain Interoperability for Closed Consortium Interfacing

Group Members:

Happy Jangir	242CS019
Kalp Patel	242CS028
Priya Jha	242CS030

Semester 2, M.Tech CSE

Department of Computer Science and Engineering
National Institute of Technology Karnataka, Surathkal

April 2025

Contents

Abstract

This report presents *CollabFed*, a groundbreaking decentralized framework designed to enable secure and verifiable interactions between private blockchain-based consortia and open consumer networks through public-private blockchain interoperability. By addressing the limitations of existing private blockchain platforms, which confine data and asset exchanges to closed boundaries, CollabFed introduces a decentralized gateway that facilitates trusted interactions in complex networks, such as e-commerce, cloud federations, and supply chains. The architecture employs a novel *Consensus on Consensus* mechanism to ensure safe and ordered processing of consumer requests, leverages Boneh-Lynn-Shacham (BLS) collective signing for verifiable responses, and optimizes latency through off-chain multi-signature aggregation. A proof-of-concept (PoC) implementation for a decentralized cloud federation, utilizing Ethereum as the public blockchain and Hyperledger Fabric/Burrow as private blockchains, demonstrates robust scalability and performance. Evaluations indicate acceptable overheads and scalability up to 32 cloud service providers (CSPs), affirming CollabFed's suitability for enterprise applications requiring seamless consumer interaction.

Chapter 1

Introduction

The proliferation of online marketplaces, encompassing Business-to-Business (B2B) and Business-to-Consumer (B2C) interactions, has revolutionized industries such as e-commerce, ride-hailing, cloud service provisioning, and supply-chain management. However, centralized platforms often suffer from market monopolization, lack of transparency, and unfair practices, as highlighted by [1]. These challenges have spurred interest in blockchain-based decentralized marketplaces to ensure transparency, fairness, and trust. Private blockchain platforms, such as Hyperledger Fabric and Corda, facilitate secure B2B transactions within closed consortia but lack mechanisms to interface with open consumer networks, limiting their applicability to B2C scenarios.

The *CollabFed* framework, proposed by Ghosh et al., addresses this gap by introducing a decentralized interface that bridges private blockchain consortia with public blockchain networks like Ethereum. This report elucidates CollabFed's architecture, which enables secure, verifiable, and fair interactions between closed business networks and open consumer networks. Key contributions include:

- A decentralized gateway leveraging interoperability between public and private blockchains.
- A *Consensus on Consensus* mechanism to ensure safe and ordered consumer request processing.
- A collective signing protocol using BLS signature for verifiable consortium responses.
- A PoC implementation for a cloud federation, demonstrating scalability and performance.

The report is organized as follows: Chapter 2 reviews related work, Chapter 3 formulates the problem, Chapter 4 details the proposed methodology, Chapter 5 describes the performance evaluation, Chapter 6 discusses results, and Chapter 7 concludes with future directions.

Chapter 2

Related Work

Blockchain technology has gained significant traction in enterprise applications, particularly for forming closed consortia in sectors like energy trading, supply chains, and cloud computing. Below, we summarize key works relevant to CollabFed, categorized by their focus:

- **Closed Consortium Blockchains**: Platforms such as IBM Food Trust [4], TradeLens [5], and Marcopolo [6] utilize private blockchain frameworks (e.g., Hyperledger Fabric, Corda) for secure B2B transactions within closed networks. These platforms ensure data privacy and efficiency but lack mechanisms to interact with open consumer networks, restricting their use in B2C applications.

- **Public Blockchain Marketplaces**: Systems like BlockV [2] for ride-sharing and ArtChain [3] for art marketplaces leverage public blockchain platforms (e.g., Ethereum) to connect businesses and consumers. However, public blockchains are unsuitable for enterprise use cases involving sensitive data due to their open nature and high computational overhead for complex smart contracts.

- **Cross-Chain Interoperability**: Protocols like Tesseract [7], Xclaim [8], and AMHL focus on public-public blockchain interoperability for cryptocurrency exchanges and asset transfers. Omniledger's Atomix [9] and Fabric Channels enable shard-based interoperability within the same blockchain. These approaches do not address public-private interoperability, which is critical for enterprise-consumer interactions.

- **Private Blockchain Interoperability**: [10] proposed protocols for trusted data transfer across private blockchains using endorsement collection, while [11] introduced a two-tier public-private architecture for data sharing. Neither framework supports consumer-facing interfaces, limiting their applicability to B2C scenarios.

- **Consensus Mechanisms**: Public blockchains employ consensus protocols like Proof of Work (PoW) [12], Proof of Stake (PoS), and Algorand [13], designed to resist Sybil attacks. Private blockchains use Byzantine Fault Tolerance (BFT) protocols, such as Byzcoin [14], assuming up to one-third of participants are malicious.

CollabFed is the first framework to address public-private blockchain interoperability for interfacing closed consortia with open consumer networks, filling a critical gap in enterprise blockchain applications.

Chapter 3

Problem Formulation

The objective is to design a decentralized architecture that enables a private blockchain-based consortium to securely and verifiably interact with an open consumer network on a public blockchain. The system must ensure fairness, security, and scalability while mitigating threats such as Byzantine behavior, Sybil attacks, impersonation, and sensitive data leakage.

3.1 System Model

We consider a partially synchronous network with an upper bound Δ on message delivery time. The system comprises:

- **Closed Consortium**: A private blockchain network of businesses (e.g., CSPs in a cloud federation) using platforms like Hyperledger Fabric or Burrow.
- **Open Consumer Network**: Consumers submitting service requests via a public blockchain (e.g., Ethereum).
- **Threat Model**: Up to one-third of participants (businesses or consumers) may exhibit Byzantine behavior. Consumers may launch Sybil attacks using multiple identities, and businesses may attempt impersonation or data leakage.

3.2 Design Guarantees

CollabFed aims to satisfy two primary guarantees:

- **Consortium Interface Safety**: All correct consortium members agree on the same set of consumer requests in the same order.
- **Consortium Interface Liveness**: All correct consumer requests are eventually processed and committed by the consortium.

3.3 Challenges

The design faces several challenges:

- **Consensus Propagation**: Securely and verifiably transferring consensus information between public and private blockchains.
- **Sensitive Data Transfer**: Ensuring confidential service responses (e.g., access credentials) are securely delivered to the intended consumer.
- **Verifiability**: Enabling consumers to verify consortium decisions without participating in the private blockchain's consensus.
- **Latency Optimization**: Minimizing overheads in request processing and signature collection.

Chapter 4

Proposed Methodology

CollabFed introduces a decentralized consortium interface that facilitates bidirectional data transfer between private blockchain consortia and public blockchain consumer networks. The architecture comprises two primary components: (1) transferring consumer requests to the consortium, and (2) transferring consortium responses to consumers.

4.1 Key Components

- **Public Blockchain Interface**: Consumers submit requests as transactions to a *User Request Contract* on a public blockchain (e.g., Ethereum), committed via consensus (e.g., PoW, PoA).
- **Consensus on Consensus**: A *Propagation Contract* in the private blockchain collects endorsements from consortium members to validate consumer requests, ensuring safety and liveness.
- **Collective Signing**: BLS signature multi-signatures are used to generate verifiable consortium responses, with off-chain signature aggregation to optimize latency.
- **Fair Scheduling**: A *Scheduling Contract* allocates consumer requests to consortium members (e.g., CSPs) based on their resource contributions.

4.2 Consensus on Consensus Mechanism

The *Consensus on Consensus* mechanism ensures secure propagation of consumer requests from the public blockchain to the private blockchain. The process is detailed in Algorithm 1.

Algorithm 1 Consensus on Consensus (Propagation Contract)

```
 1: Input: Consumer request transaction $T$ from public blockchain, sequence number
    {blocknumber, offset}
 2: Initialize: Endorsement count $EC \leftarrow 1$, private ledger $L$
 3: if no endorsement for $T$ exists in $L$ then
 4:     Consortium member commits signed endorsement to $L$
 5:     Broadcast $T$ and {blocknumber, offset} to other members
 6: end if
 7: for each consortium member receiving $T$ do
 8:     Verify $T$ using Simplified Payment Verification (SPV)
 9:     Add signed endorsement to $L$, increment $EC$
10:     Execute consensus on endorsement transaction
11: end for
12: if $EC > \frac{2}{3} \cdot |\text{consortium}|$ then
13:     Mark $T$ as approved for scheduling
14:     Trigger Scheduling Contract
15: end if
16: Return: Approved request $T$
```

4.3 Secure and Verifiable Response Transfer

Consortium responses are signed using BLS signature multi-signatures, aggregated off-chain to reduce latency. The process includes:

- **Public Information**: Consortium information (e.g., service catalogs) is signed by at least $\frac{2}{3}$ of members and posted to the public blockchain as $\{\mathcal{I}, \mathcal{H}(\mathcal{I}), B, [\mathcal{H}(\mathcal{I})]_{S_k}\}$.
- **Private Information**: Sensitive responses (e.g., access credentials) are encrypted with the consumer's public key $\mathcal{P}_{\mathcal{U}}$ and posted as $\{\langle\mathcal{M}\rangle_{\mathcal{P}_{\mathcal{U}}}, \mathcal{H}(\langle\mathcal{M}\rangle_{\mathcal{P}_{\mathcal{U}}}), B, [\mathcal{H}(\langle\mathcal{M}\rangle_{\mathcal{P}_{\mathcal{U}}})]_{S_k}\}$.
- **Verification**: Consumers verify signatures using aggregated public keys and check message integrity via hashes.

4.4 Fair Request Scheduling

The *Fair Request Scheduling Contract* (Algorithm 2) allocates consumer requests to CSPs based on their infrastructure contributions, ensuring fairness.

Algorithm 2 Fair Request Scheduling Contract

```
1: **Input**: Consumer request $\mathcal{R}_i$, contribution proportions $K$, window $W$
2: **Output**: Scheduled CSP $\mathcal{C}_s$
3: for each $\mathcal{C}_j \in \mathcal{F}$ do
4:     $\mathcal{G}_{\mathcal{C}_j} \leftarrow 0$
5:     for $i \leftarrow 1$ to $|W|$ do
6:         if $W[i] = \mathcal{C}_j$ then
7:             $\mathcal{G}_{\mathcal{C}_j} \leftarrow \mathcal{G}_{\mathcal{C}_j} + 1$
8:         end if
9:     end for
10:    $\mathcal{G}_{\mathcal{C}_j} \leftarrow \mathcal{G}_{\mathcal{C}_j}/|W|$
11:    $\mathcal{D}_{\mathcal{C}_j} \leftarrow K_{\mathcal{C}_j} - \mathcal{G}_{\mathcal{C}_j}$
12: end for
13: $\mathcal{C}_s \leftarrow \arg\max_{\mathcal{C}_j \in \mathcal{F}} \mathcal{D}_{\mathcal{C}_j}$
14: Update $W$ with $\mathcal{C}_s$
15: **Return**: $\mathcal{C}_s$
```

4.5 Latency Optimization

Multi-signature collection is performed off-chain using an M-ary tree structure to minimize latency. A blockchain-based fallback mechanism handles denial-of-service attacks by malicious members, ensuring robustness.

Chapter 5

Performance Evaluation and Experimental Setting

CollabFed was evaluated through a PoC implementation of *CollabCloud*, a decentralized cloud federation, using Ethereum (public blockchain) and Hyperledger Fabric/Burrow (private blockchain). The evaluation included a testbed with 3 CSPs and a Mininet-based emulation with up to 32 CSPs.

5.1 Evaluation Metrics

- **Latency**: Time taken for request processing, scheduling, VM provisioning, and multi-signature collection.
- **Resource Consumption**: CPU and memory usage by CollabFed services.
- **Gas Consumption**: Transaction fees for Ethereum smart contracts.
- **Scalability**: Performance with increasing CSPs and network latency.

5.2 Experimental Setup

Table 5.1: Experimental Parameters

Parameter	**Description**
Public Blockchain	Ethereum (Ropsten: PoW, Rinkeby: PoA)
Private Blockchain	Hyperledger Fabric, Burrow
Testbed CSPs	3 (6 servers: 4-core i5, 88-core Xeon)
Mininet CSPs	2–32
Inter-CSP Latency	50–400 ms
Consumer Requests	4–64 (parallel)
Services	Docker containers, VirtualBox VMs

5.3 Testbed Configuration

Each CSP in the testbed had two servers:

- **CollabFed Server**: 4-core Intel i5, 8GB memory, Ubuntu 18.04.
- **VM Hosting Server**: 88-core Intel Xeon, 256GB memory, CentOS 7.7.

Services ran in Docker containers, with VirtualBox for VM provisioning and Flask for request handling.

Chapter 6

Results and Discussion

6.1 Public Blockchain Performance

Figure 6.1 illustrates the latency distribution for consumer request processing on Ethereum's Ropsten (PoW) and Rinkeby (PoA) networks. Rinkeby's faster consensus mechanism results in lower latency.

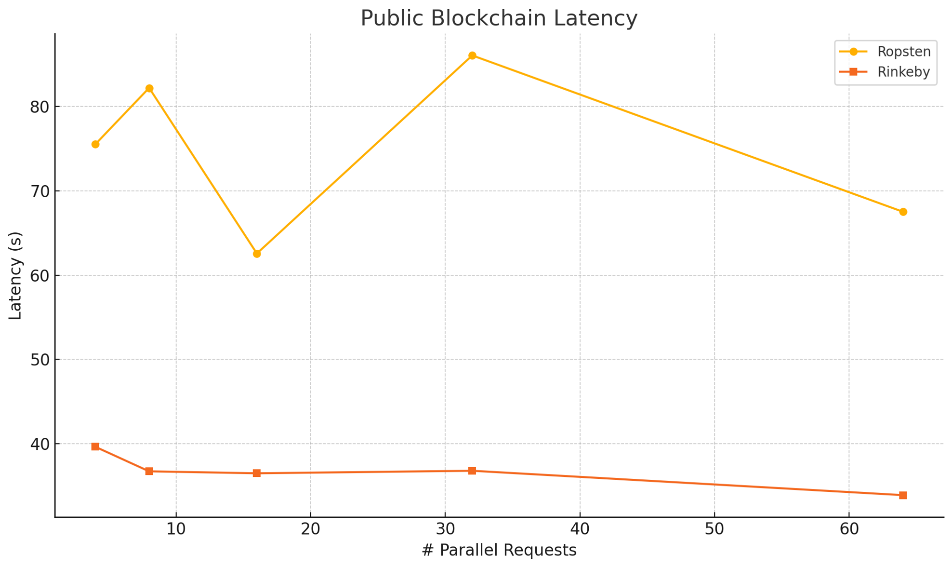


Figure 6.1: Public Blockchain Latency

Figure 6.2 shows gas consumption for CollabFed's smart contracts. The *Resource Provisioning Contract* incurs the highest cost due to storing multi-signatures and encrypted data, comparable to Ethereum applications like CryptoKitties [12].

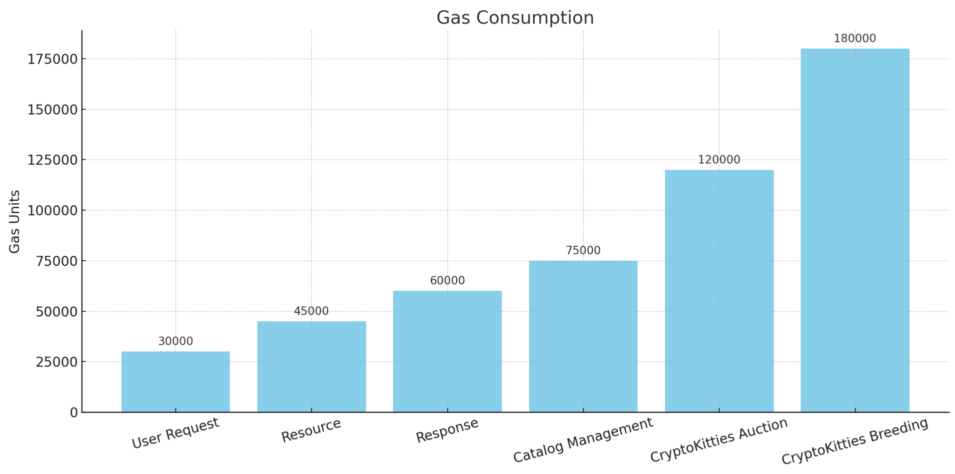


Figure 6.2: Gas Consumption

6.2 Private Blockchain Scheduling

Figure 6.3 compares scheduling latency for Fabric (execute-order) and Burrow (order-execute). Burrow outperforms Fabric for the *Fair Request Scheduling Contract* due to fewer transaction failures, as Fabric's execute-order flow causes state inconsistencies.

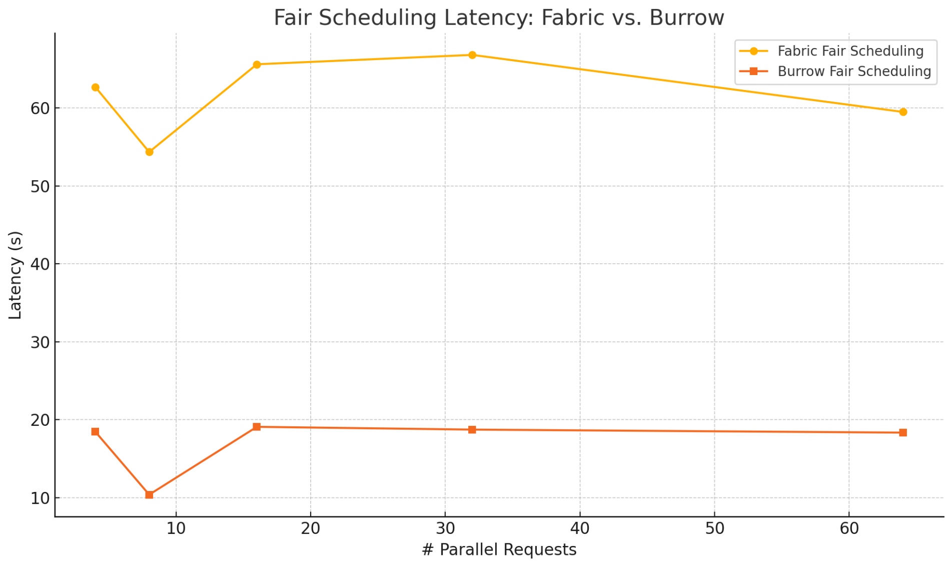


Figure 6.3: Fair Scheduling Latency: Fabric vs. Burrow

Figure 6.4 demonstrates that a naive static scheduling contract significantly reduces Fabric's latency, confirming that state-dependent contracts are the bottleneck.

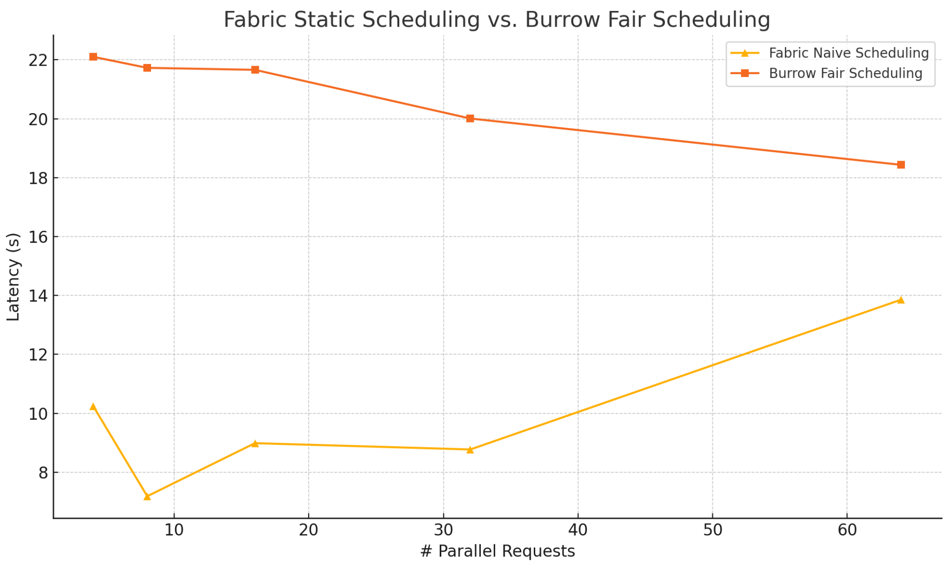


Figure 6.4: Fabric Static Scheduling vs. Burrow Fair Scheduling

6.3 VM Provisioning and Signature Collection

Figure 6.5 shows VM provisioning latency, which increases with parallel requests due to hardware limitations. Figure 6.6 indicates consistent BLS signature collection latency, unaffected by request volume.

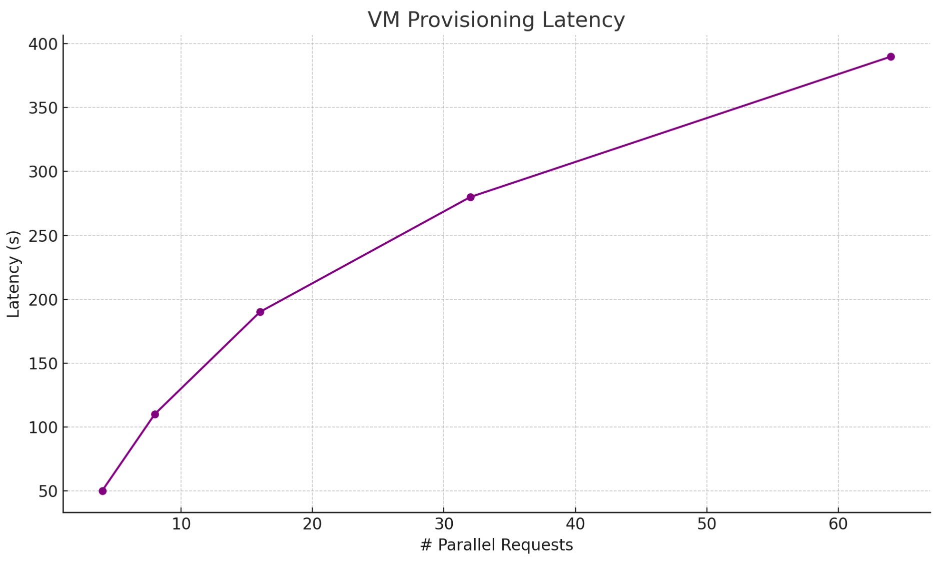


Figure 6.5: VM Provisioning Latency

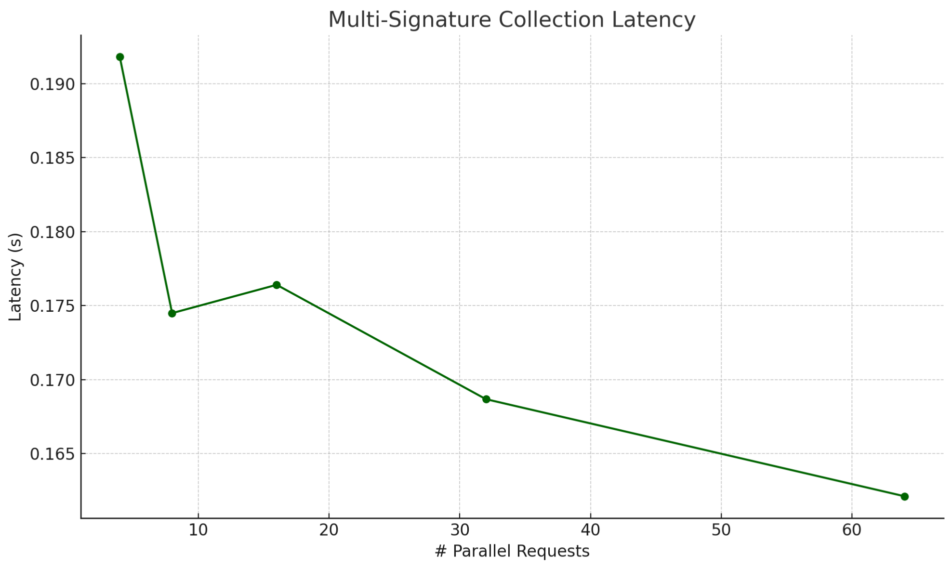


Figure 6.6: Multi-Signature Collection Latency

6.4 Resource Consumption

Figures 6.7 and 6.8 show CPU and memory usage, respectively. CPU usage remains below 10%, and memory usage is under 200 MB, indicating low overhead.

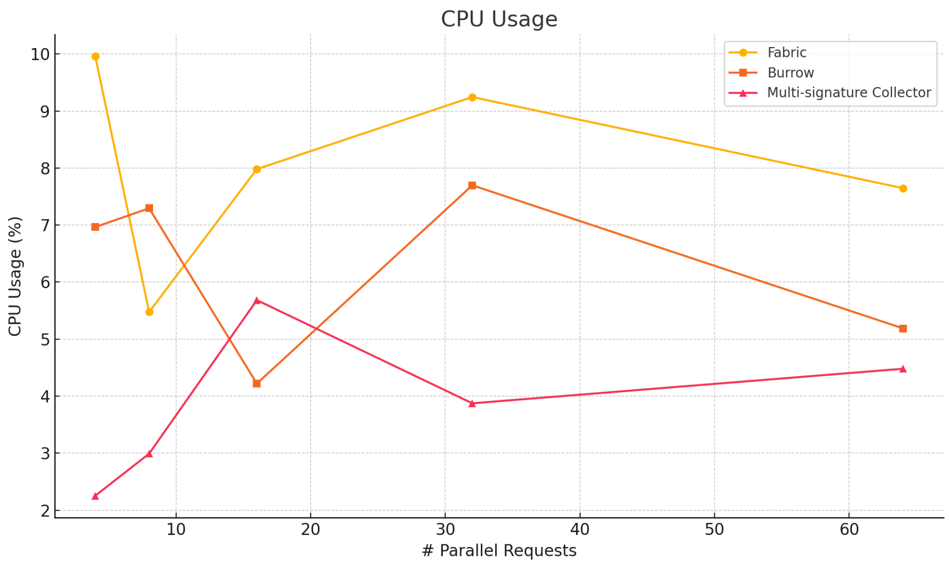


Figure 6.7: CPU Usage

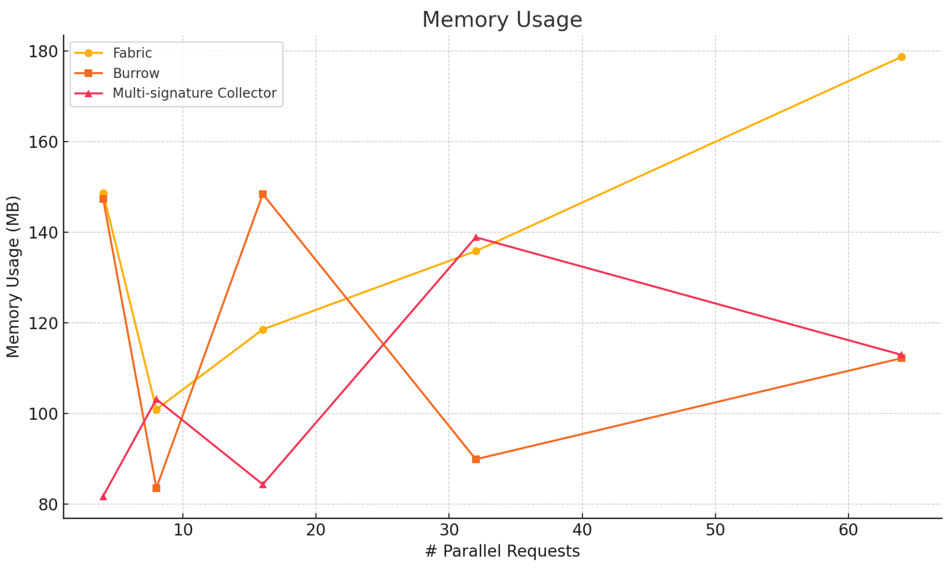


Figure 6.8: Memory Usage

6.5 Scalability

Figure 6.9 shows Burrow's propagation contract latency, with a median of 2.5 seconds for 32 CSPs at 400 ms inter-CSP latency. Figure 6.10 indicates BLS signature aggregation latency of 3.5 seconds for 32 CSPs, demonstrating scalability.

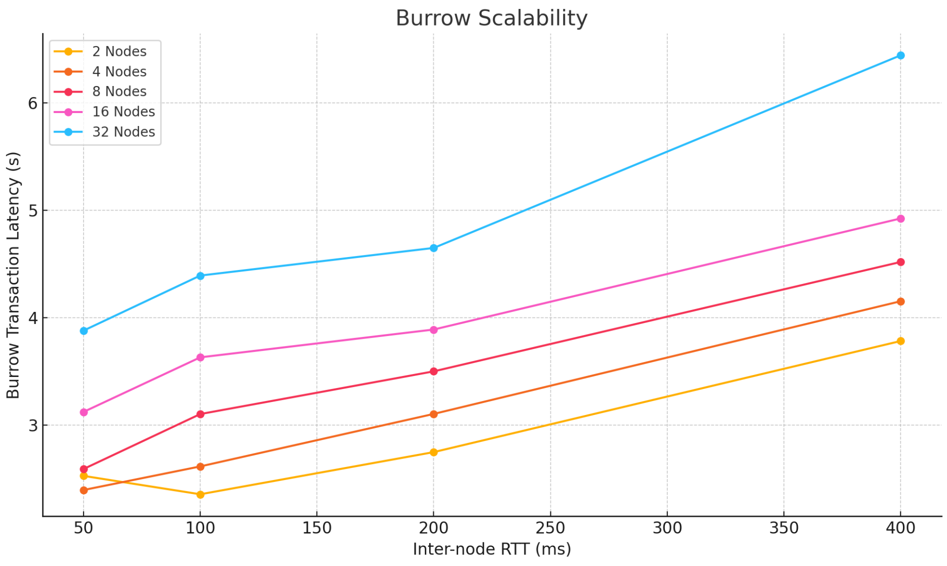


Figure 6.9: Burrow Scalability

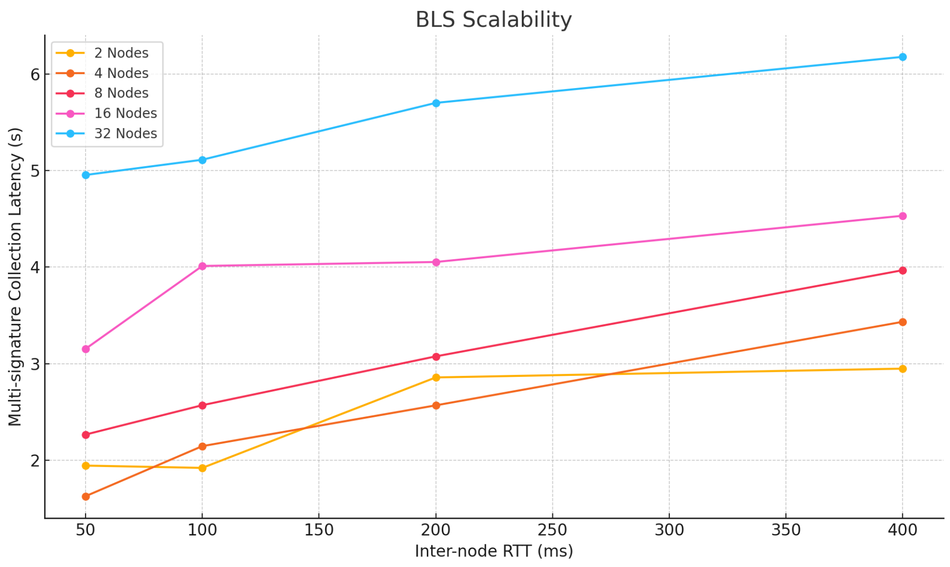
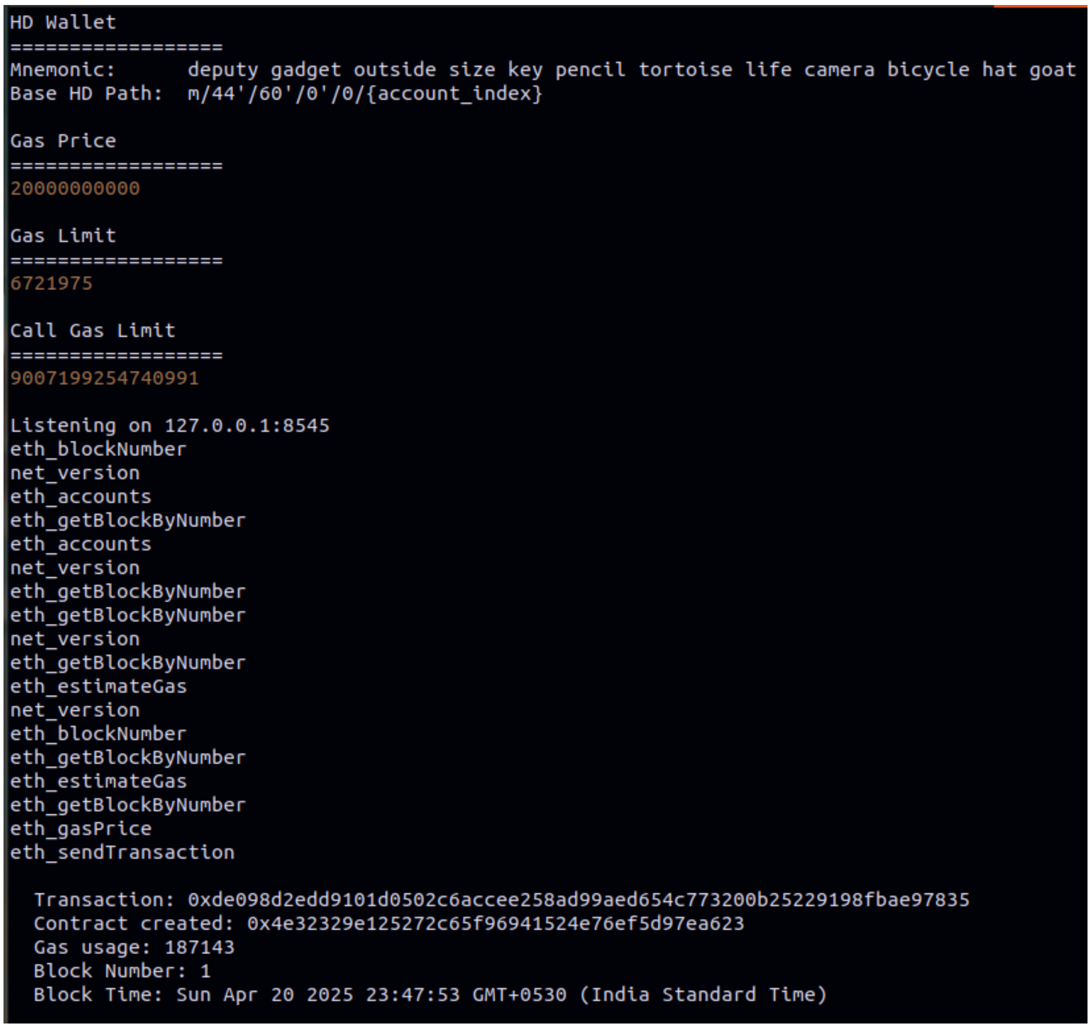


Figure 6.10: BLS Scalability

Table 6.1 illustrates the effect of the M-ary tree structure on BLS signature collection latency, with optimal performance at $M = 4$.

Table 6.1: Effect of Communication Tree on Multi-Signature Collection Latency

M	1	2	3	4	5	6	7	8
Mean Latency (s)	29.9	5.0	3.2	2.3	2.4	2.5	3.0	2.9
Standard Deviation	1.8	0.3	0.2	0.1	0.2	0.3	0.6	1.5

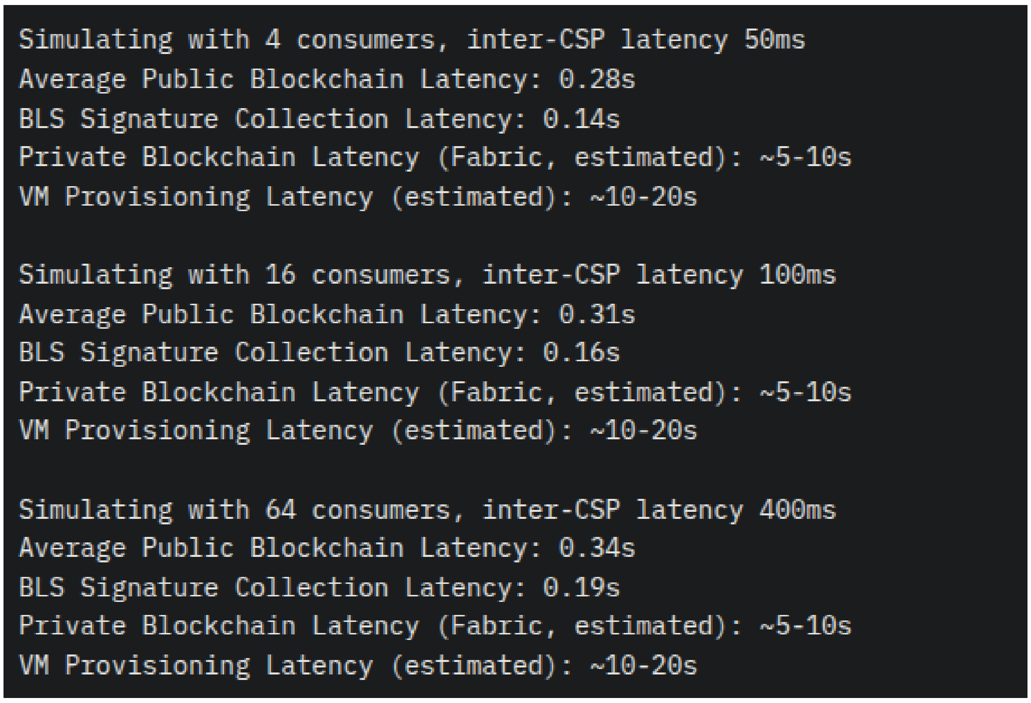


```
HD Wallet
==================
Mnemonic:      deputy gadget outside size key pencil tortoise life camera bicycle hat goat
Base HD Path:  m/44'/60'/0'/0/{account_index}

Gas Price
==================
20000000000

Gas Limit
==================
6721975

Call Gas Limit
==================
9007199254740991

Listening on 127.0.0.1:8545
eth_blockNumber
net_version
eth_accounts
eth_getBlockByNumber
eth_accounts
net_version
eth_getBlockByNumber
eth_getBlockByNumber
net_version
eth_getBlockByNumber
eth_estimateGas
net_version
eth_blockNumber
eth_getBlockByNumber
eth_estimateGas
eth_getBlockByNumber
eth_gasPrice
eth_sendTransaction

  Transaction: 0xde098d2edd9101d0502c6accee258ad99aed654c773200b25229198fbae97835
  Contract created: 0x4e32329e125272c65f96941524e76ef5d97ea623
  Gas usage: 187143
  Block Number: 1
  Block Time: Sun Apr 20 2025 23:47:53 GMT+0530 (India Standard Time)
```

Figure 6.11: HD Wallet Setup Output

The screenshot in Figure 6.11 displays the hierarchical deterministic (HD) wallet configuration used for contract deployment. It lists the 12-word mnemonic seed phrase, the base derivation path (‘m/44’/60’/0’/0/account$_i ndex‘), andtheconfiguredgasparameters(gasprice :$ 20$gwei, gaslimit :$ 6721975$).TheoutputfurtherconfirmsthattheJSON-RPCserverislisteningon‘$127
8545‘, $andenumeratesavailableRPCmethodssuchas‘eth_blockNumber‘, ‘eth_accounts‘, and‘eth_sendTran$

These details imply that the experiments were conducted on a controlled local Ethereum environment, ensuring reproducible account generation and deterministic transaction behavior. By fixing the HD path and gas settings, the deployment scripts achieve consistent address derivation, cost estimation, and broadcast performance across test runs.

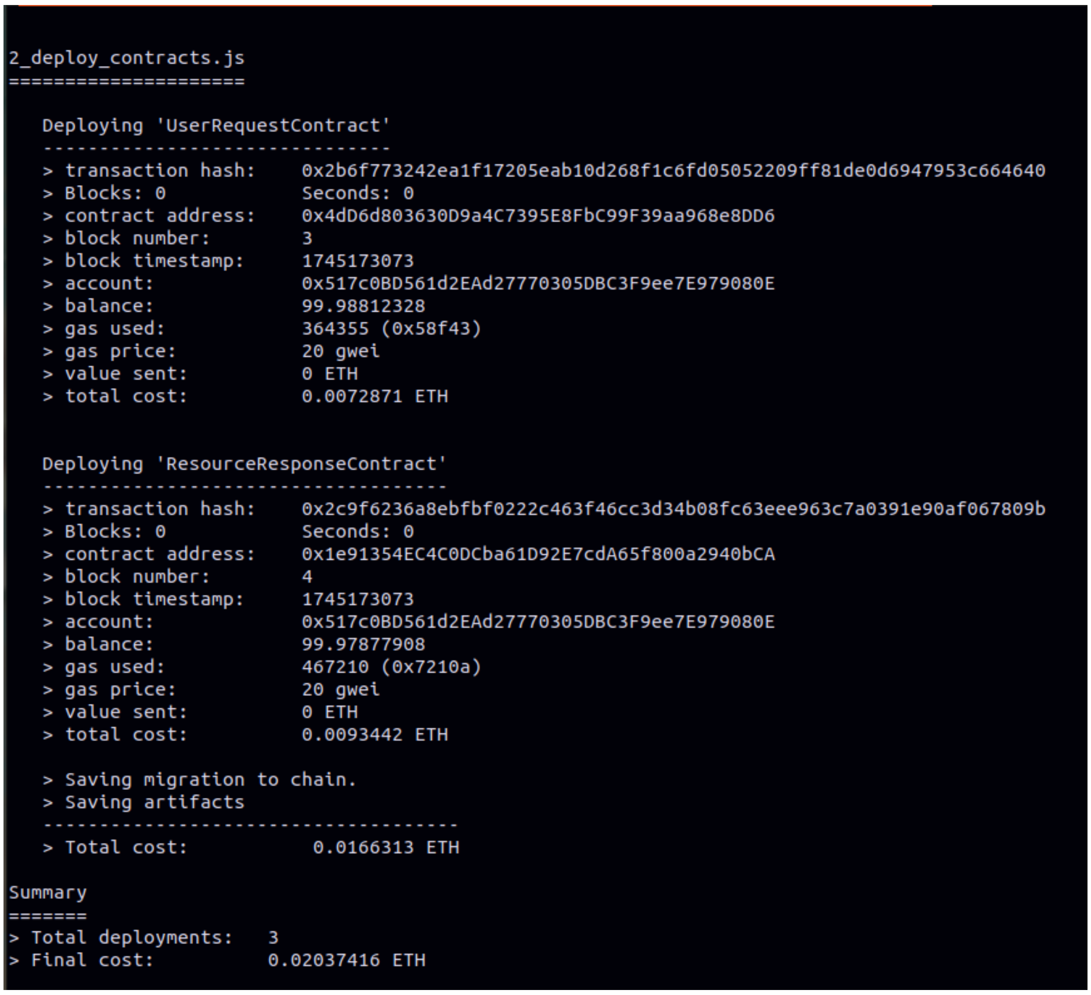


```
Simulating with 4 consumers, inter-CSP latency 50ms
Average Public Blockchain Latency: 0.28s
BLS Signature Collection Latency: 0.14s
Private Blockchain Latency (Fabric, estimated): ~5-10s
VM Provisioning Latency (estimated): ~10-20s

Simulating with 16 consumers, inter-CSP latency 100ms
Average Public Blockchain Latency: 0.31s
BLS Signature Collection Latency: 0.16s
Private Blockchain Latency (Fabric, estimated): ~5-10s
VM Provisioning Latency (estimated): ~10-20s

Simulating with 64 consumers, inter-CSP latency 400ms
Average Public Blockchain Latency: 0.34s
BLS Signature Collection Latency: 0.19s
Private Blockchain Latency (Fabric, estimated): ~5-10s
VM Provisioning Latency (estimated): ~10-20s
```

Figure 6.12: End-to-End Latency Summary with Consumer Load Variations

Figure 6.12 presents a consolidated view of end-to-end latency when varying the number of consumers (4, 16, 64) alongside inter-CSP latencies (50ms, 100ms, 400ms). It confirms that public blockchain latency remains under 0.35s in all scenarios, whereas private blockchain scheduling (Fabric) and VM provisioning latency dominate the total response time, scaling from roughly 5–10s and 10–20s, respectively.

This summary underscores that while public-chain operations remain performant under load, private infrastructure processes become the bottleneck at higher concurrency levels. The observed escalation in VM provisioning latency signals the need for optimized orchestration or additional compute resources to maintain service-level agreements under peak demands.

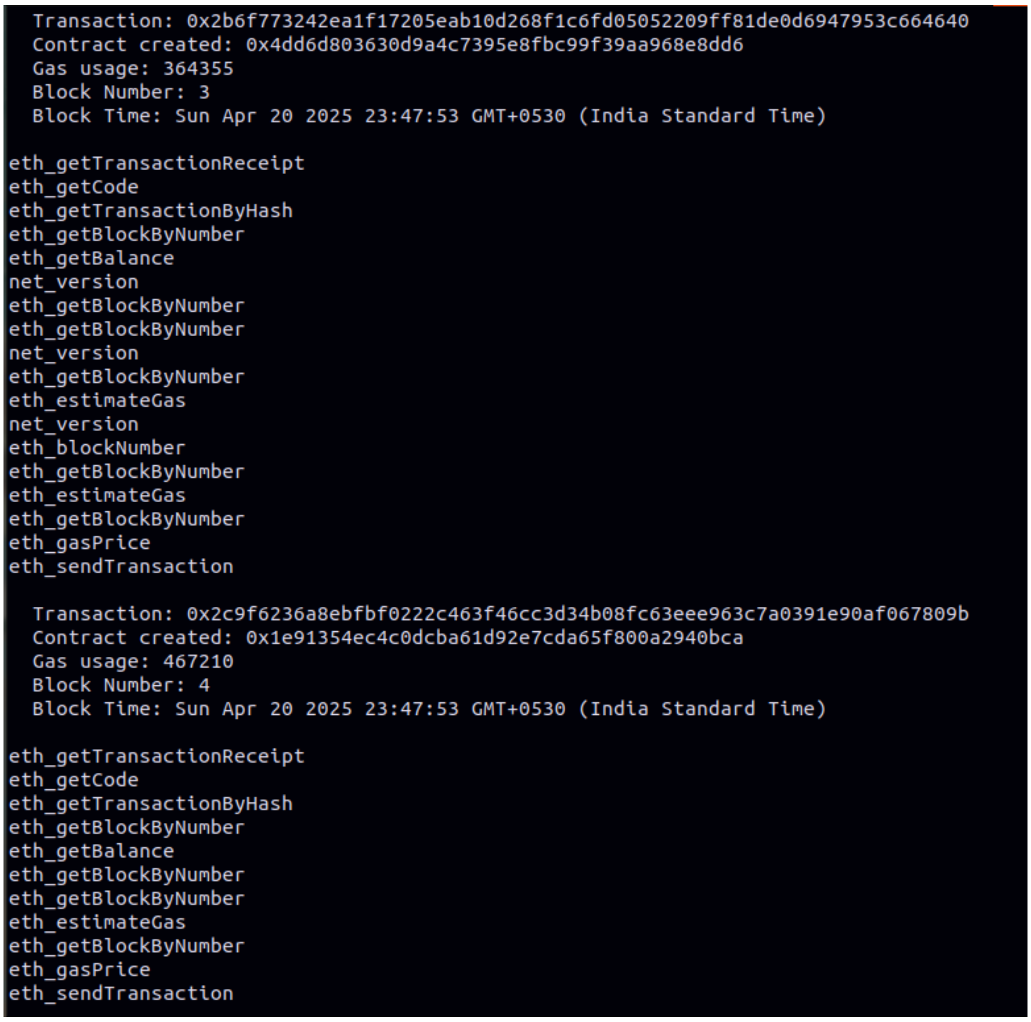


```
2_deploy_contracts.js
=====================

   Deploying 'UserRequestContract'
   -------------------------------
   > transaction hash:    0x2b6f773242ea1f17205eab10d268f1c6fd05052209ff81de0d6947953c664640
   > Blocks: 0            Seconds: 0
   > contract address:    0x4dD6d803630D9a4C7395E8FbC99F39aa968e8DD6
   > block number:        3
   > block timestamp:     1745173073
   > account:             0x517c0BD561d2EAd27770305DBC3F9ee7E979080E
   > balance:             99.98812328
   > gas used:            364355 (0x58f43)
   > gas price:           20 gwei
   > value sent:          0 ETH
   > total cost:          0.0072871 ETH


   Deploying 'ResourceResponseContract'
   ------------------------------------
   > transaction hash:    0x2c9f6236a8ebfbf0222c463f46cc3d34b08fc63eee963c7a0391e90af067809b
   > Blocks: 0            Seconds: 0
   > contract address:    0x1e91354EC4C0DCba61D92E7cdA65f800a2940bCA
   > block number:        4
   > block timestamp:     1745173073
   > account:             0x517c0BD561d2EAd27770305DBC3F9ee7E979080E
   > balance:             99.97877908
   > gas used:            467210 (0x7210a)
   > gas price:           20 gwei
   > value sent:          0 ETH
   > total cost:          0.0093442 ETH

   > Saving migration to chain.
   > Saving artifacts
   -------------------------------------
   > Total cost:           0.0166313 ETH

Summary
=======
> Total deployments:   3
> Final cost:          0.02037416 ETH
```

Figure 6.13: Smart Contract Deployment Output

Figure 6.13 details gas usage and ETH cost for deploying the ‘UserRequestContract‘ (364355 gas 0.0073ETH) and ‘ResourceResponseContract‘ (467210 gas 0.0093ETH), culminating in a total cost of 0.0204ETH. It logs transaction hashes, block numbers, and gas metrics, providing visibility into economic overheads for contract complexity.

These metrics illustrate that additional contract features—such as multi-signature handling and encrypted data storage—drive up deployment costs. Quantifying this overhead is crucial for assessing the viability of mainnet deployments versus alternative chains or Layer-2 scaling strategies.



```
  Transaction: 0x2b6f773242ea1f17205eab10d268f1c6fd05052209ff81de0d6947953c664640
  Contract created: 0x4dd6d803630d9a4c7395e8fbc99f39aa968e8dd6
  Gas usage: 364355
  Block Number: 3
  Block Time: Sun Apr 20 2025 23:47:53 GMT+0530 (India Standard Time)

eth_getTransactionReceipt
eth_getCode
eth_getTransactionByHash
eth_getBlockByNumber
eth_getBalance
net_version
eth_getBlockByNumber
eth_getBlockByNumber
net_version
eth_getBlockByNumber
eth_estimateGas
net_version
eth_blockNumber
eth_getBlockByNumber
eth_estimateGas
eth_getBlockByNumber
eth_gasPrice
eth_sendTransaction

  Transaction: 0x2c9f6236a8ebfbf0222c463f46cc3d34b08fc63eee963c7a0391e90af067809b
  Contract created: 0x1e91354ec4c0dcba61d92e7cda65f800a2940bca
  Gas usage: 467210
  Block Number: 4
  Block Time: Sun Apr 20 2025 23:47:53 GMT+0530 (India Standard Time)

eth_getTransactionReceipt
eth_getCode
eth_getTransactionByHash
eth_getBlockByNumber
eth_getBalance
eth_getBlockByNumber
eth_getBlockByNumber
eth_estimateGas
eth_getBlockByNumber
eth_gasPrice
eth_sendTransaction
```

Figure 6.14: JSON-RPC Call Trace for Deployment Transactions

The trace in Figure 6.14 captures the full JSON-RPC call sequence for both deployment transactions, including ‘eth$_{g}etTransactionReceipt$‘, ‘$eth_{g}etCode$‘, and‘$eth_{g}etBalance$‘, $pairedwith$
47 : 53$GMT + 0530$‘), $highlightingthedeterministicfinalityofthelocaltestnet$.

Logging each RPC invocation with a uniform timestamp ensures a repeatable and synchronous experimental setup, enabling precise breakdown of latency contributions from blockchain interactions. This level of detail is essential for thorough performance profiling and comparative analysis across different deployment environments.

Chapter 7

Conclusion and Future Work

CollabFed introduces a pioneering framework for public-private blockchain interoperability, enabling closed consortia to interface seamlessly with open consumer networks. Key contributions include:

- A decentralized gateway ensuring secure and verifiable data transfer.
- A *Consensus on Consensus* mechanism for safe and ordered request processing.
- BLS signature-based collective signing for verifiable consortium responses.
- A scalable PoC implementation for a cloud federation, with low overheads and scalability up to 32 CSPs.

The evaluation demonstrates CollabFed's applicability to enterprise use cases, with acceptable latency, resource consumption, and gas costs. The framework's scalability and robustness make it suitable for diverse applications, including e-commerce, supply chains, and healthcare.

7.0.1 Future Work

Future research directions include:

- **Protocol Optimization**: Investigating alternative consensus protocols (e.g., Proof of Stake, HotStuff) to further reduce latency and gas costs.
- **Security Enhancements**: Developing advanced cryptographic techniques, such as zero-knowledge proofs, to enhance privacy and prevent data leakage.
- **Diverse Applications**: Extending CollabFed to other domains, such as decentralized finance (DeFi) and IoT networks, to validate its versatility.
- **Fault Tolerance**: Analyzing the impact of network partitions and higher Byzantine node fractions on system performance.

By addressing these areas, CollabFed can evolve into a robust, general-purpose framework for blockchain interoperability, driving the adoption of decentralized systems in enterprise and consumer applications.

Bibliography

[1] J. Haucap and U. Heimeshoff. Google, Facebook, Amazon, eBay: Is the Internet driving competition or market monopolization? *International Economics and Economic Policy*, vol. 11, 2014.

[2] P. Pal and S. Ruj. BlockV: A blockchain enabled peer-peer ride sharing service. In *IEEE Blockchain*, 2019.

[3] Z. Wang et al. ArtChain: Blockchain-enabled platform for art marketplace. In *IEEE Blockchain*, 2019.

[4] IBM. IBM Food Trust. `https://www.ibm.com/in-en/blockchain/solutions/food-trust`, Accessed Jan 5, 2021.

[5] TradeLens. `https://www.tradelens.com/`, Accessed Jan 5, 2021.

[6] Marco Polo - A Trade Finance Initiative. `https://www.marcopolo.finance/`, Accessed Jan 5, 2021.

[7] I. Bentov et al. Tesseract: Real-time cryptocurrency exchange using trusted hardware. In *ACM CCS*, 2019.

[8] A. Zamyatin et al. Xclaim: Trustless, interoperable, cryptocurrency-backed assets. In *IEEE Symposium on Security and Privacy*, 2019.

[9] E. Kokoris-Kogias et al. Omniledger: A secure, scale-out, decentralized ledger via sharding. In *IEEE Symposium on Security and Privacy*, 2018.

[10] E. Abebe et al. Enabling enterprise blockchain interoperability with trusted data transfer. In *International Middleware Conference Industrial Track*, 2019.

[11] M. Cash and M. Bassiouni. Two-tier permission-ed and permission-less blockchain for secure data sharing. In *IEEE International Conference on Smart Cloud*, 2018.

[12] G. Wood. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Project Yellow Paper*, 2014.

[13] Y. Gilad et al. Algorand: Scaling byzantine agreements for cryptocurrencies. In *SOSP*, 2017.

[14] E. K. Kogias et al. Enhancing bitcoin security and performance with strong consistency via collective signing. In *USENIX Security*, 2016.