Leveraging Public-Private Blockchain Interoperability for Closed Consortium Interfacing

CS-740 Blockchain Design Architecture

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

April 2025

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

Outline

Introduction

Introduction

- Online marketplaces face challenges: monopolization, lack of transparency, unfair practices [1].
- Private blockchains (e.g., Hyperledger Fabric, Corda) excel in B2B but lack consumer interfaces.
- **CollabFed**: Decentralized framework for public-private blockchain interoperability.
- Key contributions:
  - Secure decentralized gateway.
  - *Consensus on Consensus* mechanism.
  - BLS signature for verifiable responses.
  - Cloud federation proof-of-concept.

Related Work

Related Work

- **Closed Consortium Blockchains**: IBM Food Trust [2], TradeLens [5] ensure B2B security but lack consumer interfaces.
- **Public Blockchain Marketplaces**: BlockV [3], ArtChain [6] are consumer-facing but unsuitable for sensitive data.
- **Cross-Chain Interoperability**: Tesseract [7], Xclaim [8] focus on public-public interoperability.
- **Private Blockchain Interoperability**: [9], [10] enable private-private data transfer but not B2C.
- CollabFed bridges public-private gap for enterprise-consumer interactions.

Problem Formulation

Problem Formulation

Objective
Enable secure, verifiable interactions between private blockchain consortia and public blockchain consumer networks.

System Model

- **Closed Consortium**: Private blockchain (e.g., Hyperledger Fabric).
- **Open Consumer Network**: Public blockchain (e.g., Ethereum).
- **Threat Model**: Up to 1/3 Byzantine participants, Sybil attacks, impersonation.

Challenges

Design Guarantees

- **Safety**: Agreement on consumer requests.
- **Liveness**: All valid requests processed.

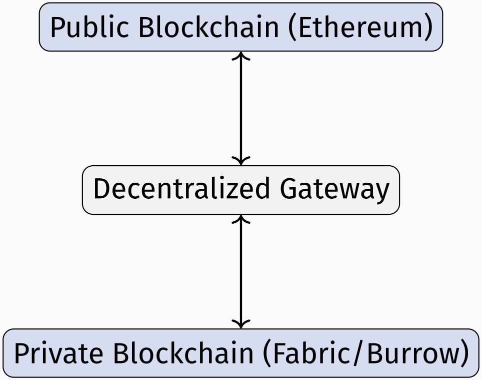
- **Consensus Propagation**: Securely transfer consensus between blockchains.
- **Sensitive Data Transfer**: Protect confidential responses.
- **Verifiability**: Enable consumer verification of consortium decisions.
- **Latency Optimization**: Minimize processing and signature collection overheads.

Proposed Methodology

CollabFed Architecture

- **Public Blockchain Interface**: Consumer requests via Ethereum smart contracts.
- **Consensus on Consensus**: Validates requests in private blockchain.
- **Collective Signing**: BLS multi-signatures for verifiable responses.
- **Fair Scheduling**: Allocates requests based on contributions.

CollabFed Architecture



Public Blockchain (Ethereum)
Decentralized Gateway
Private Blockchain (Fabric/Burrow)

Consensus on Consensus

Algorithm 1 Consensus on Consensus

```
1:  Input: Consumer request $T$, {blocknumber, offset}
2:  Initialize: Endorsement count $EC \leftarrow 1$, ledger $L$
3:  if no endorsement for $T$ in $L$ then
4:      Commit signed endorsement to $L$
5:      Broadcast $T$ and {blocknumber, offset}
6:  end if
7:  for each consortium member do
8:      Verify $T$ using SPV
9:      Add endorsement to $L$, increment $EC$
10: end for
11: if $EC > \frac{2}{3} \cdot |\text{consortium}|$ then
12:     Approve $T$ for scheduling
13: end if
```

Secure Response Transfer

- **Public Information**: Signed consortium data on Ethereum.
- **Private Information**: Encrypted responses using consumer's public key.
- **Verification**: BLS signature and hash verification.
- **Latency Optimization**: Off-chain signature aggregation with M-ary tree.

Fair Request Scheduling

Algorithm 2 Fair Request Scheduling

```
1: **Input**: Request $\mathcal{R}_i$, contributions $\mathbb{K}$, window $\mathbb{W}$
2: **Output**: Scheduled CSP $\mathcal{C}_s$
3: **for** each $\mathcal{C}_j \in \mathcal{F}$ **do**
4:     Compute allocation gap $\mathcal{D}_{\mathcal{C}_j}$
5: **end for**
6: $\mathcal{C}_s \leftarrow \arg\max_{\mathcal{C}_j \in \mathcal{F}} \mathcal{D}_{\mathcal{C}_j}$
7: Update $\mathbb{W}$ with $\mathcal{C}_s$
8: **Return**: $\mathcal{C}_s$
```

Performance Evaluation

Experimental Setup

Testbed

- **Public Blockchain**: Ethereum (Ropsten: PoW, Rinkeby: PoA).
- **Private Blockchain**: Hyperledger Fabric, Burrow.
- **CSPs**: 3 (testbed), 2–32 (Mininet).
- **Servers**: 4-core i5, 88-core Xeon.

Metrics

- Latency, resource consumption, gas costs, scalability.

Results and Discussion

Public Blockchain Performance

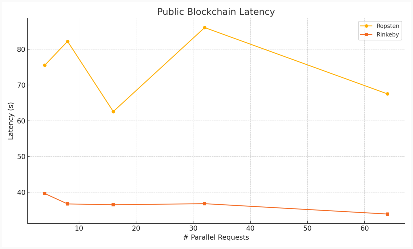


Figure 1: Public Blockchain Latency

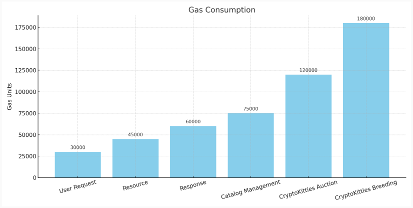


Figure 2: Gas Consumption

- Rinkeby (PoA) has lower latency than Ropsten (PoW).
- Gas costs comparable to CryptoKitties [4].

Private Blockchain Scheduling

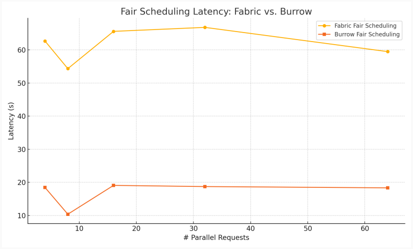


Figure 3: Scheduling Latency

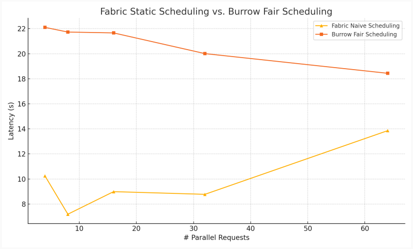


Figure 4: Static vs. Fair Scheduling

- Burrow outperforms Fabric due to fewer transaction failures.
- Static scheduling reduces Fabric's latency.

Scalability

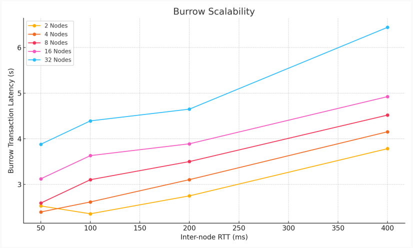


Figure 5: Burrow Scalability

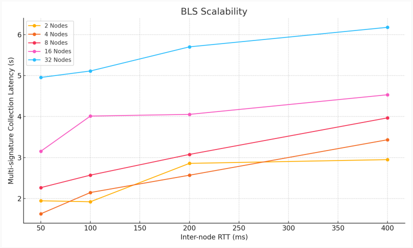


Figure 6: BLS Scalability

- Median propagation latency: 2.5s for 32 CSPs.
- BLS signature aggregation: 3.5s for 32 CSPs.

Resource Consumption

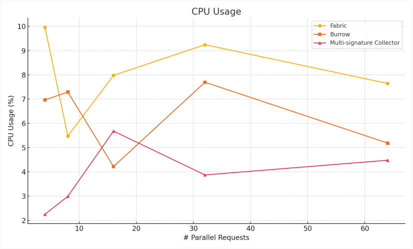


Figure 7: CPU Usage

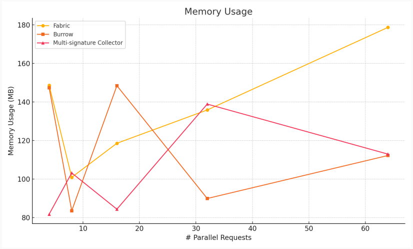
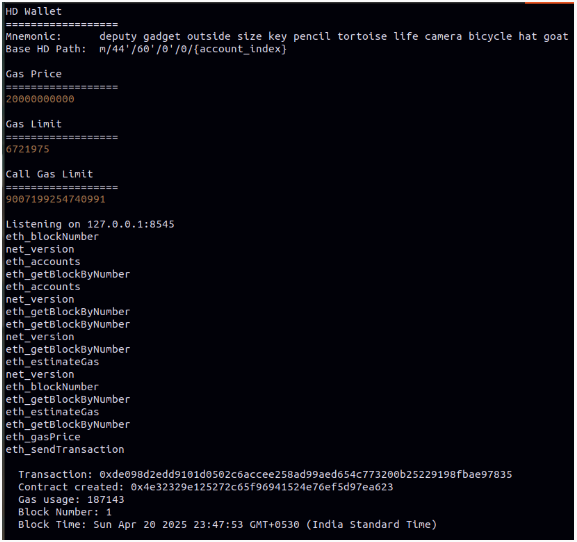


Figure 8: Memory Usage

- CPU usage < 10%, memory usage < 200 MB.

Detailed Results

HD Wallet Configuration



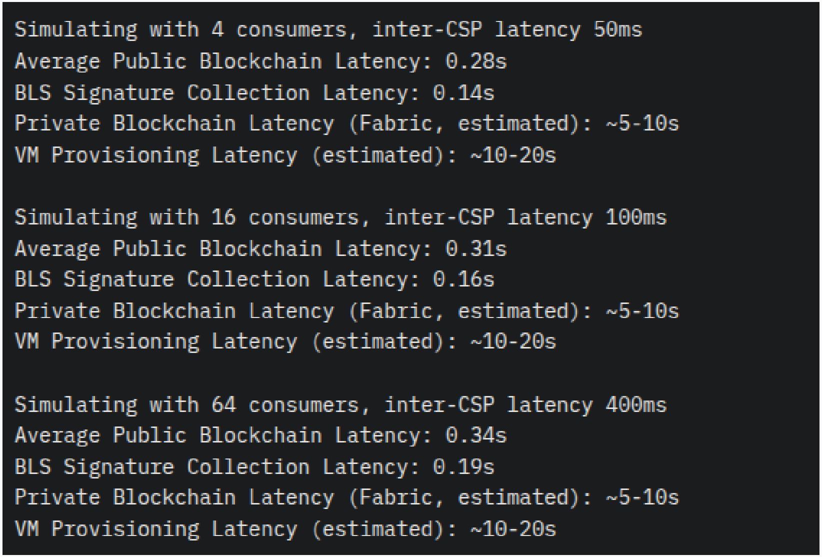
```
HD Wallet
==================
Mnemonic:      deputy gadget outside size key pencil tortoise life camera bicycle hat goat
Base HD Path:  m/44'/60'/0'/0/{account_index}

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 9: HD Wallet Setup Output

The HD wallet configuration screenshot shows the full 12-word mnemonic seed and the base derivation path `m/44’/60’/0’/0/{account_index}`, along with the gas parameters (20 gwei gas price, 6 721 975 gas limit). Below, the local node’s JSON-RPC endpoint on `127.0.0.1:8545` is confirmed, and methods like `eth_accounts` and `eth_sendTransaction` are listed.

This setup ensures that all accounts and transaction parameters remain deterministic across experiments. By fixing the derivation path and gas settings, deployments can be repeated reliably, which is critical for consistent latency and cost measurements.

End-to-End Latency Summary



```
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
```

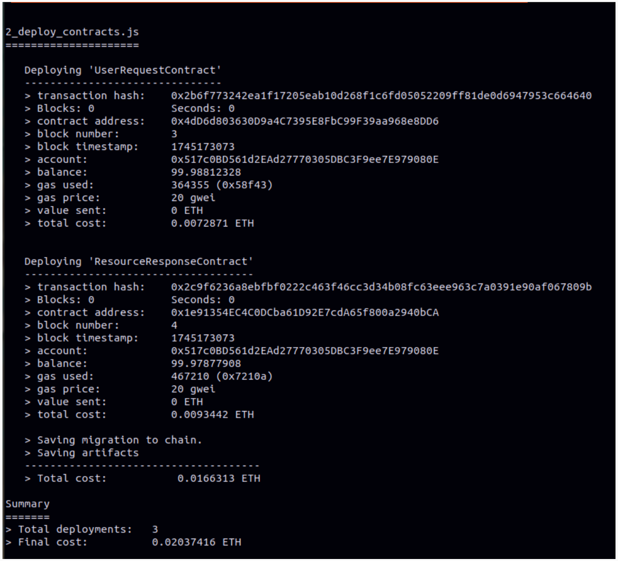
Figure 10: Latency vs. Consumer Load and Inter-CSP Delay

End-to-End Latency Summary

This summary plot presents overall response times for 4, 16, and 64 consumers at inter-CSP delays of 50 ms, 100 ms, and 400 ms. Public-chain latency remains under 0.35 s in all cases, whereas Fabric scheduling (5–10 s) and VM provisioning (10–20 s) dominate the end-to-end delay.

The data highlight that increasing consumer concurrency has marginal effect on the Ethereum interface but significantly magnifies private-infrastructure delays. The VM provisioning curve in particular suggests resource contention under high parallel load, motivating future optimisations or horizontal scaling.

Smart Contract Deployment Costs



```
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
```

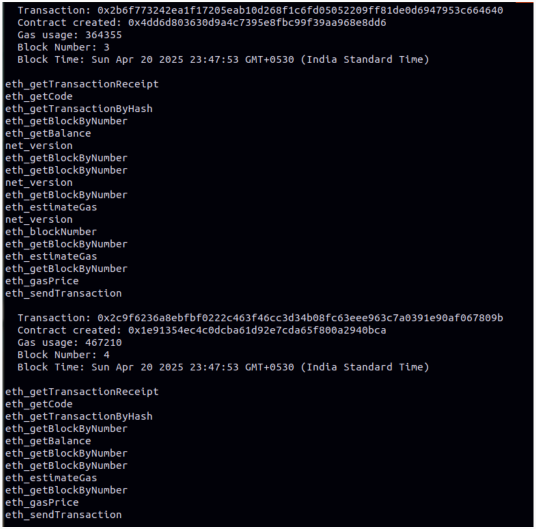
Figure 11: Gas Usage and ETH Cost per Contract

Smart Contract Deployment Costs

The deployment log shows two key contracts: `UserRequestContract` (364 355 gas ≈ 0.0073 ETH) and `ResourceResponseContract` (467 210 gas ≈ 0.0093 ETH), totalling about 0.0204 ETH. Transaction hashes, block numbers, and per-contract gas metrics are also recorded.

These figures demonstrate that richer contract functionality—multi-signature handling and encrypted data storage—directly increases gas consumption and economic overhead. Such cost profiling is essential when weighing mainnet vs. Layer-2 or private-chain deployment strategies.

JSON-RPC Call Trace



```
  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 12: RPC Sequence for Deployment Transactions

JSON-RPC Call Trace

This trace captures every JSON-RPC invocation—`eth_getTransactionReceipt`, `eth_getCode`, `eth_getBalance`, etc.—for both deployments, all confirmed at the same timestamp (Sun Apr 20 2025 23:47:53 GMT+0530). It illustrates deterministic, atomic block finality on the local testnet.

Logging each RPC call and observing uniform timestamps ensures a fully reproducible pipeline for performance measurements. This granularity is key to dissecting latency contributions across the blockchain interaction stack.

Conclusion

Conclusion

- **CollabFed**: Pioneering public-private blockchain interoperability.
- Key features:
  - Secure decentralized gateway.
  - *Consensus on Consensus* for request processing.
  - BLS signatures for verifiable responses.
  - Scalable cloud federation PoC.
- Applications: E-commerce, supply chains, healthcare.

Future Work

- **Protocol Optimization**: Explore PoS, HotStuff.
- **Security Enhancements**: Zero-knowledge proofs.
- **Diverse Applications**: DeFi, IoT.
- **Fault Tolerance**: Analyze network partitions, Byzantine behavior.

References i

References

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

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

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

References ii

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

[5] TradeLens. TradeLens: Digitizing global supply chains. `https://www.tradelens.com`, 2020.

[6] ArtChain. ArtChain: A blockchain-based art marketplace. *White Paper*, 2020.

[7] Tesseract Authors. Tesseract: Cross-chain interoperability framework. *arXiv preprint*, 2019.

[8] Xclaim Authors. Xclaim: Trustless cross-chain asset transfer. *IEEE Symposium on Security and Privacy*, 2019.

References iii

[9] E. Abebe et al. Trusted oracle-based data exchange in private blockchains. *IEEE Transactions on Blockchain*, 2020.

[10] M. Cash et al. Two-tier blockchain architecture for private consortia. *Journal of Distributed Ledger Technologies*, 2020.