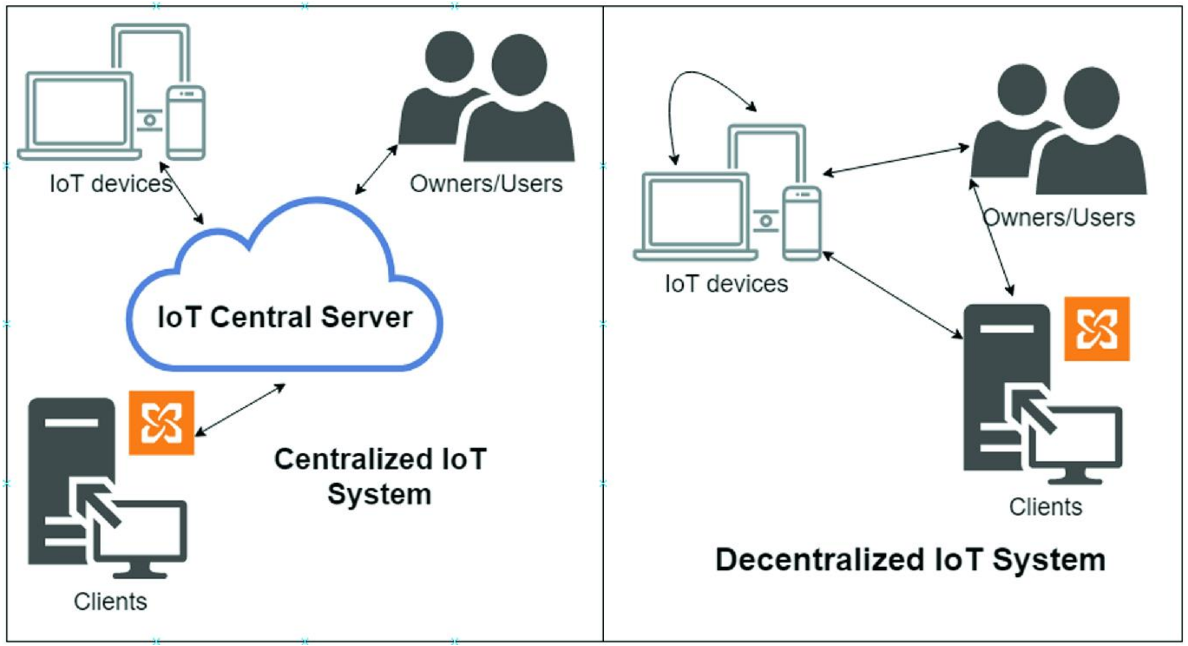
Non-standard applications of blockchain technology - blockchain for enterprise applications

Bc. Matúš Revický

Supervisor: doc. RNDr. Jozef Jirásek, PhD.

Problem Statement I. (motivation)

- Traditional databases are susceptible to attacks, where an attacker can **permanently change or delete data** inside the database.
- Assumption that devices that are connected trust each other
- Hard to know which device to trust in the network.
- Need for a solution that can store data about every transaction in an environment where participants do not fully trust one another, and the data needs to be resistant to cyber-attacks.
- **Desired solution: Involved parties have access to data that can be trusted**.



Goals and motivation

Review relevant literature about applications of blockchain technology not only in cryptocurrencies.

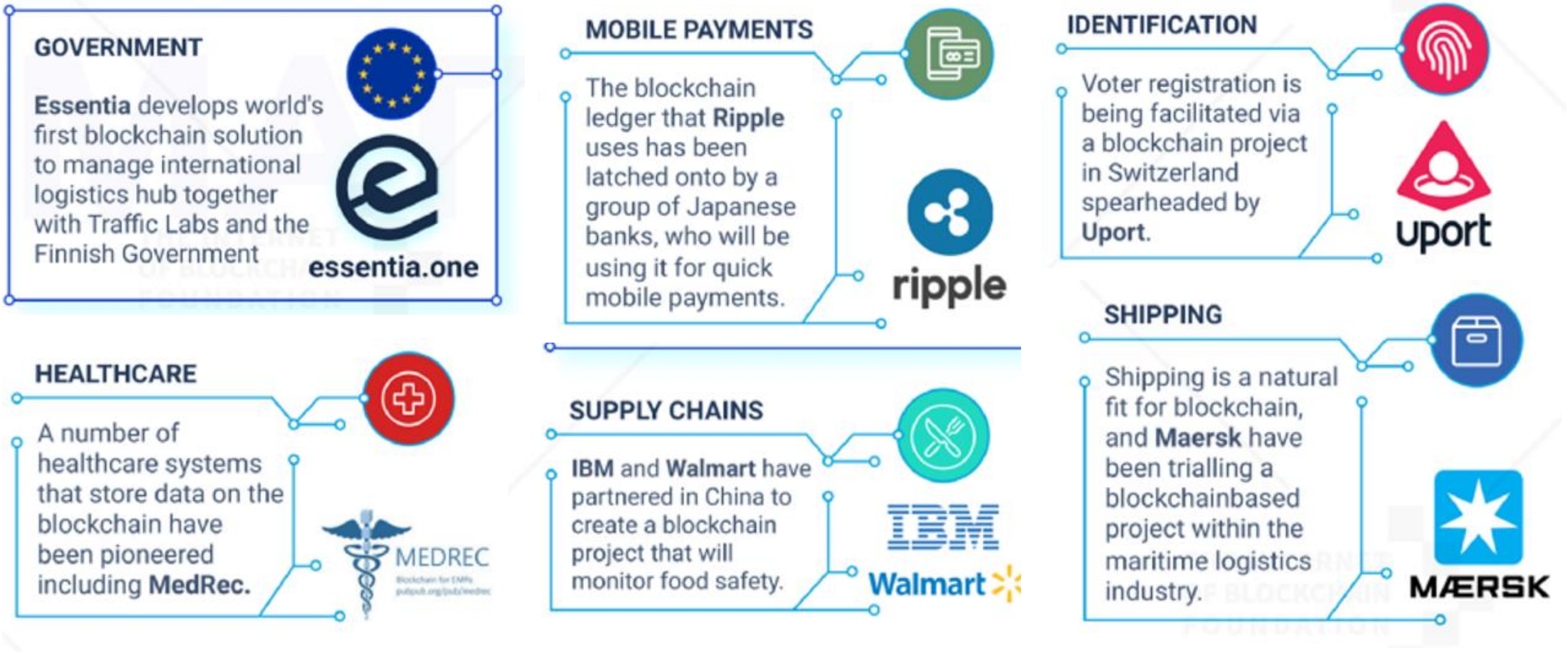
Evaluate capabilities of existing blockchain frameworks.

Design of decentralized system that can utilize IoT devices by employing a viable blockchain implementation.

Implement and evaluate a proof of concept in a real life scenario that could benefit from utilizing IoT devices.

Evaluate performance of developed solution.

Blockchain - Real world uses cases



GOVERNMENT

Essentia develops world's first blockchain solution to manage international logistics hub together with Traffic Labs and the Finnish Government

essentia.one

HEALTHCARE

A number of healthcare systems that store data on the blockchain have been pioneered including **MedRec.**

MEDREC

MOBILE PAYMENTS

The blockchain ledger that **Ripple** uses has been latched onto by a group of Japanese banks, who will be using it for quick mobile payments.

SUPPLY CHAINS

IBM and **Walmart** have partnered in China to create a blockchain project that will monitor food safety.

IDENTIFICATION

Voter registration is being facilitated via a blockchain project in Switzerland spearheaded by **Uport**.

SHIPPING

Shipping is a natural fit for blockchain, and **Maersk** have been trialling a blockchainbased project within the maritime logistics industry.

Problem I. - Review relevant literature about applications of blockchain technology not only in cryptocurrencies

Aspect	Public chains	Enterprise chains
Confidentiality	No	**Yes**
Anonymity	No	**Yes**
Membership	Permissionless	**Permissioned via voting, KYC, usually under an enterprise blockchain.**
Identity	Anonymous	**Known users**
Consensus	PoW/PoS	**BFT**
Finality	Mostly probabilistic	**Requires immediate/instant finality.**
Transaction speed	Slower	**Faster (usually, should be).**
Scalability	Better	**Not very scalable, usually due to consensus choice. Usually a much smaller number of nodes compared to public chains.**
Regulatory compliance	Not usually required	**Required at times.**
Trust	Fully decentralized	**Semi-centralized and managed via consortium and voting mechanisms.**
Smart contracts	Not strictly required; for example, in the Bitcoin chain	**Strictly required to support arbitrary business functions.**

Problem I. - review relevant literature about applications of blockchain technology not only in cryptocurrencies

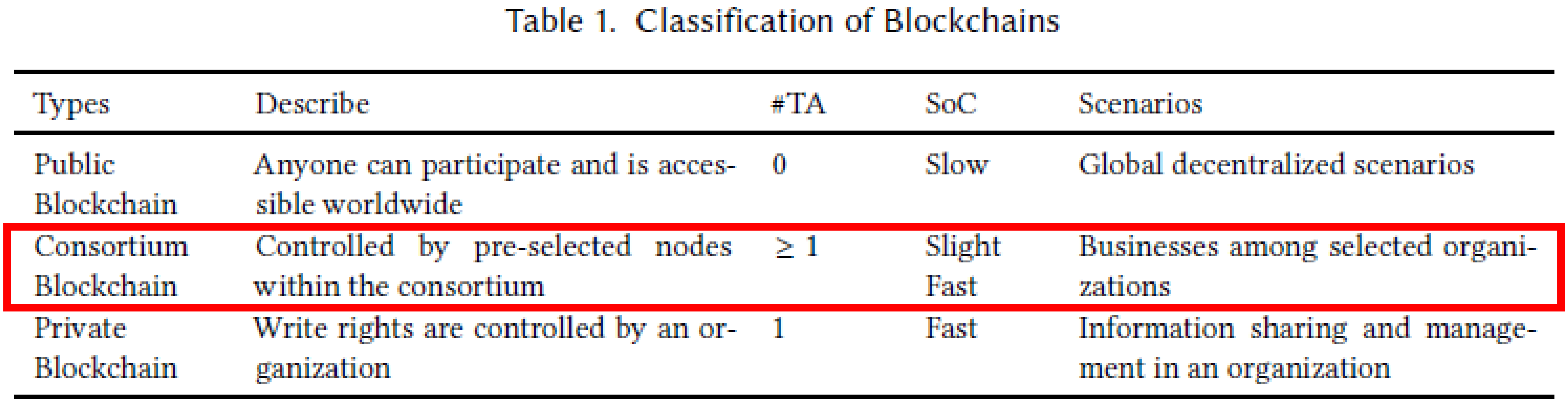
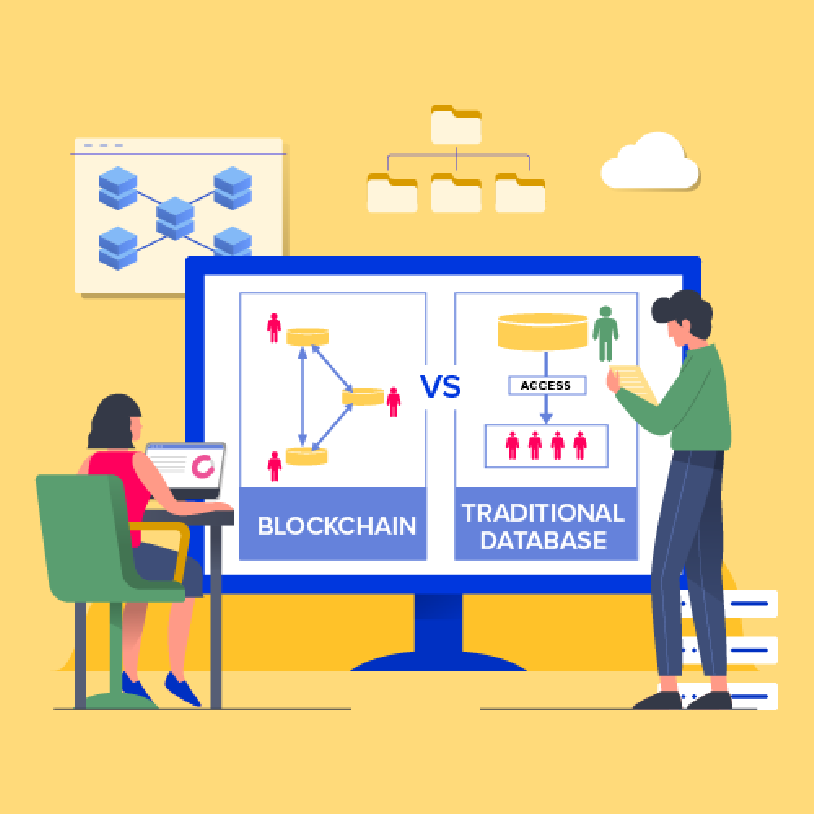


Table 1. Classification of Blockchains

Types	Describe	#TA	SoC	Scenarios
Public Blockchain	Anyone can participate and is accessible worldwide	0	Slow	Global decentralized scenarios
Consortium Blockchain	Controlled by pre-selected nodes within the consortium	≥ 1	Slight Fast	Businesses among selected organizations
Private Blockchain	Write rights are controlled by an organization	1	Fast	Information sharing and management in an organization

Problem II. - Evaluate capabilities of existing blockchain frameworks

Feature/Framework	Quorum	Fabric	Corda
Performance (TPS)	700	**560**	600
Consensus mechanism	Pluggable multiple Raft, IBFT, PoA	**Pluggable Raft, unoffiacial SmartBFT**	Pluggable, notarybased
Tooling	Rich enterprise tooling	**SDKs**	Rich enterprise tooling
Smart contract language	Solidity	**Golang/Java/Javascript/Typescript**	Kotlin/Java
Access control	Enterprise-grade permissioning mechanism	**Membership service provides/certificate based**	Doorman service/KYC. Certificate based
Implementation language	Golang, Java	**Golang**	Kotlin
Node membership	Smart contract and node software managed	**Via membership service provider**	Node software managed using configuration files, certificate authority controlled
Member identification	Public keys/ addresses	**PKI-based via membership service provider, supports organization identity**	PKI-based, support organization identity
Cryptography used	SECP256K1, AES, CURVE25519 + XSALSA20 + POLY13050, PBKDF2, SCRYPT,	**SECP256R1**	ED255519, SECP256R1, RSA – PKCS1
Smart contract runtime	EVM	**Sandboxed in Docker containers**	Deterministic JVM
Upgradeable smart contract	Possible with some patterns, not inherently supported	**Allowed via upgrade transactions**	Allowed via administrator privileges and auto update allowed under administrative checks
Tokenization support	Flexible—inherited from public Ethereum standards	**Programmable**	Corda token SDK



Where blockchain could provide value

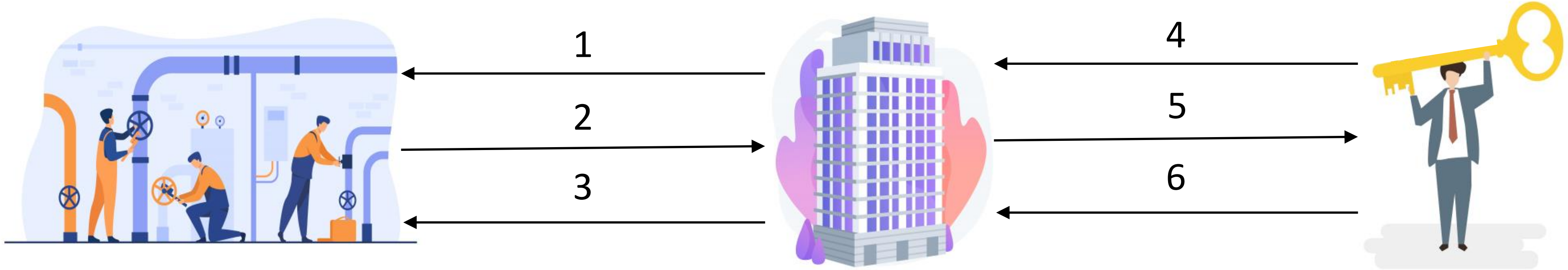
Situations that favor the use of blockchain technology include

- **when multiple parties are sharing and updating shared data,**
- **there is a need for reliable records,**
- **there are intermediaries that add costs, and/or**
- **there is a lack of trust between involved parties.**

Some of the **blockchain's key advantages** include disintermediation, improved product traceability, increased transparency of transaction histories, as well as enhanced security of records regarding fraud and unauthorized activities.

Design of decentralized system that can utilize IoT devices by employing a viable blockchain implementation

1. The property manager requests registration of main heat meter for the whole building from the heating distributor.
2. The heating distributor accepts registration request of the main heat meter.
3. The property manager regularly updates main meter value.
4. The final consumer requests registration of local heat meter.
5. The property manager accepts registration of local heat meter.
6. The final consumer regularly updates own local meter value.



Heating distributor **Property management company** **Final consumer**

Design of decentralized system that can utilize IoT devices by employing a viable blockchain implementation.

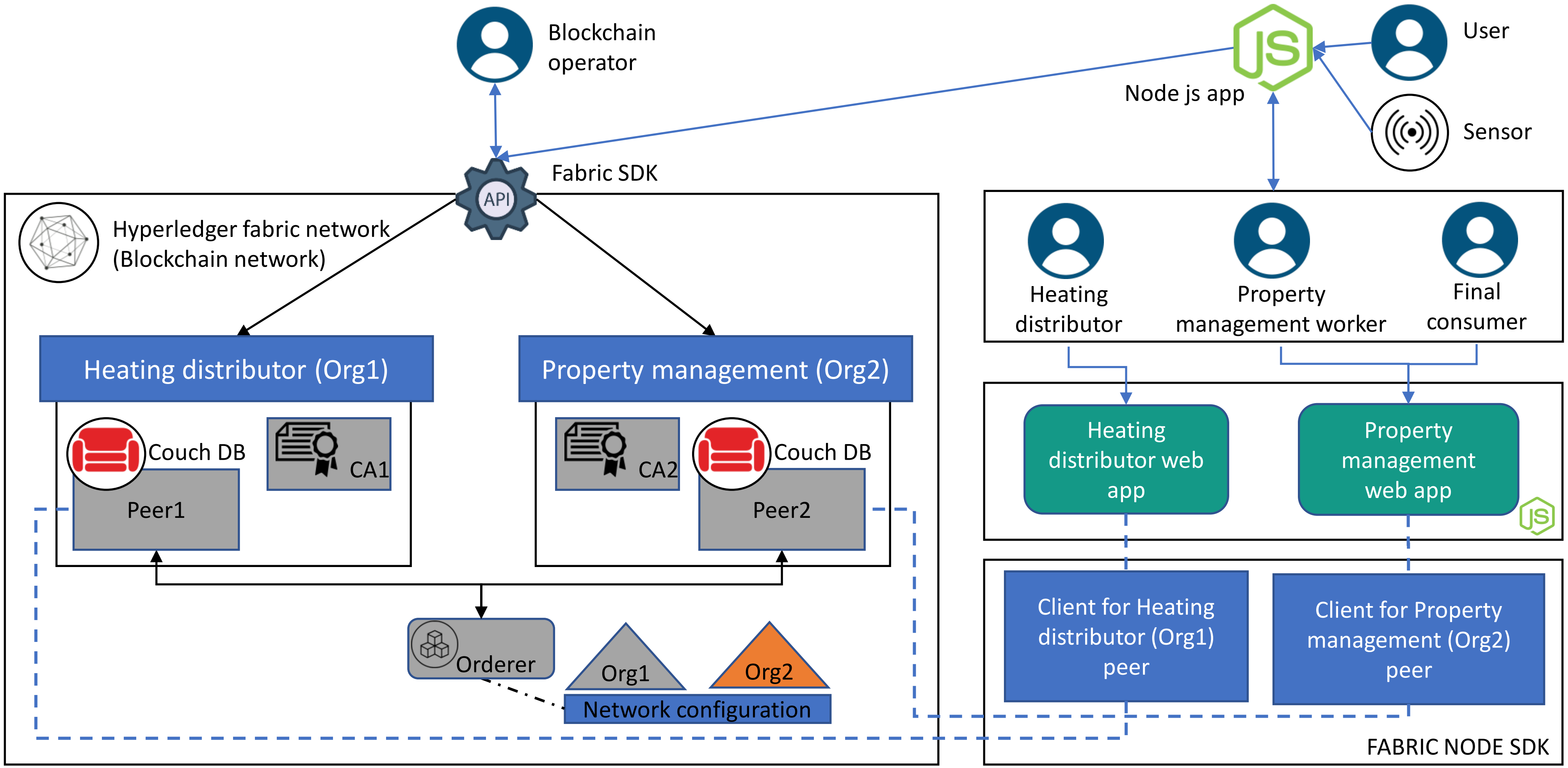
Asset type	Asset attributes
Meter	meterID, consumerMSP, approverMSP, value, status

Data type	Data attributes	
Meter	Request and acceptance status:	• by heating distributor and housing cooperative respectively • by final consumer and property manager respectively
	Current value(heat consumed):	• by final consumer and property manager

The capabilities and restrictions of each role:

- Only property manager may request main heat meter registration for the entire apartment building.
- Only heating distributor may accept main heat meter registration request.
- …

High level architecture

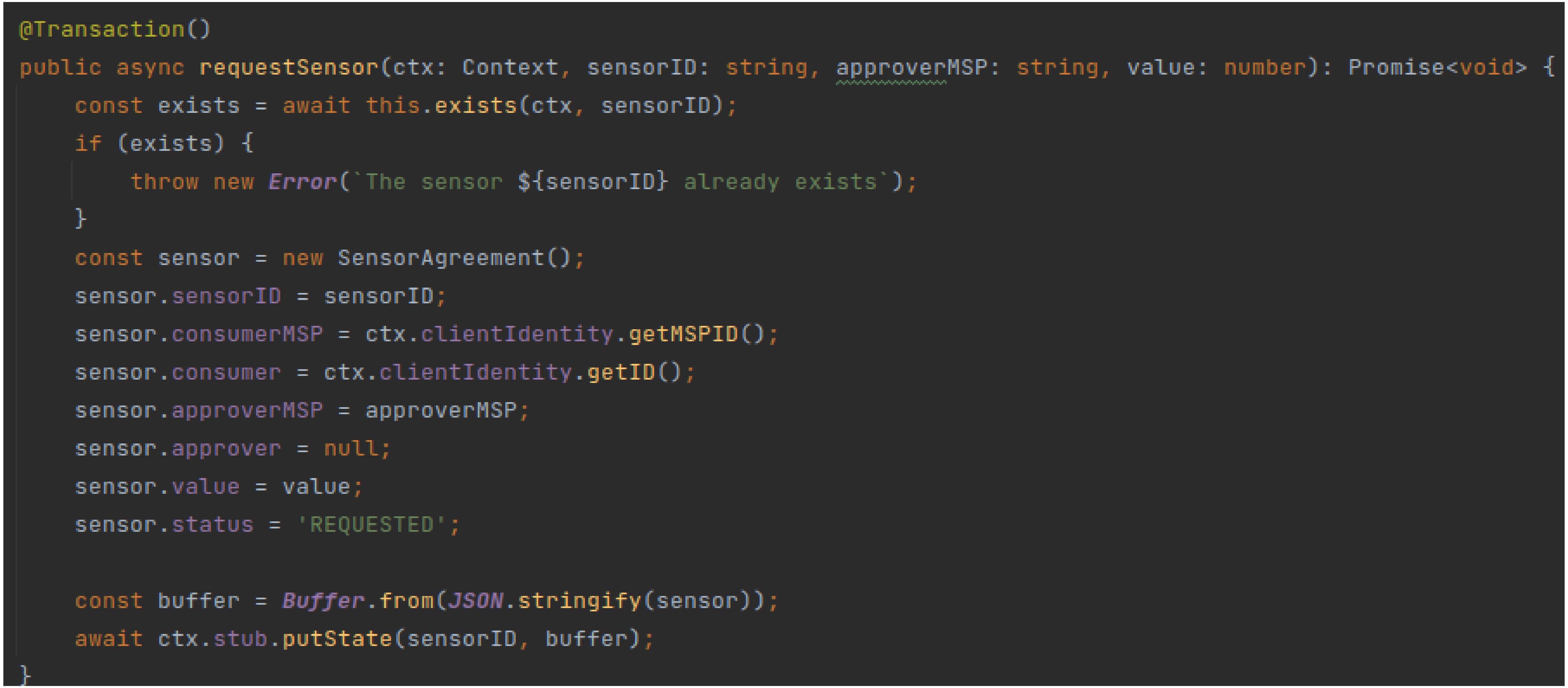


Blockchain operator
Node js app
User
Sensor
Fabric SDK
API
Hyperledger fabric network (Blockchain network)
Heating distributor (Org1)
Property management (Org2)
Couch DB
CA1
Peer1
CA2
Couch DB
Peer2
Orderer
Org1
Org2
Network configuration
Heating distributor
Property management worker
Final consumer
Heating distributor web app
Property management web app
Client for Heating distributor (Org1) peer
Client for Property management (Org2) peer
FABRIC NODE SDK

High level architecture – local development

IMAGE	COMMAND	NAMES
hyperledger/fabric-tools:2.3	"/bin/bash"	cli
hyperledger/fabric-peer:2.3	"peer node start"	peer0.org1.example.com
hyperledger/fabric-peer:2.3	"peer node start"	peer0.org2.example.com
couchdb:3.1.1	"tini -- /docker-ent…"	couchdb0
hyperledger/fabric-orderer:2.3	"orderer"	orderer.example.com
couchdb:3.1.1	"tini -- /docker-ent…"	couchdb1
hyperledger/fabric-ca:latest	"sh -c 'fabric-ca-se…"	ca_orderer
hyperledger/fabric-ca:latest	"sh -c 'fabric-ca-se…"	ca_org2
hyperledger/fabric-ca:latest	"sh -c 'fabric-ca-se…"	ca_org1

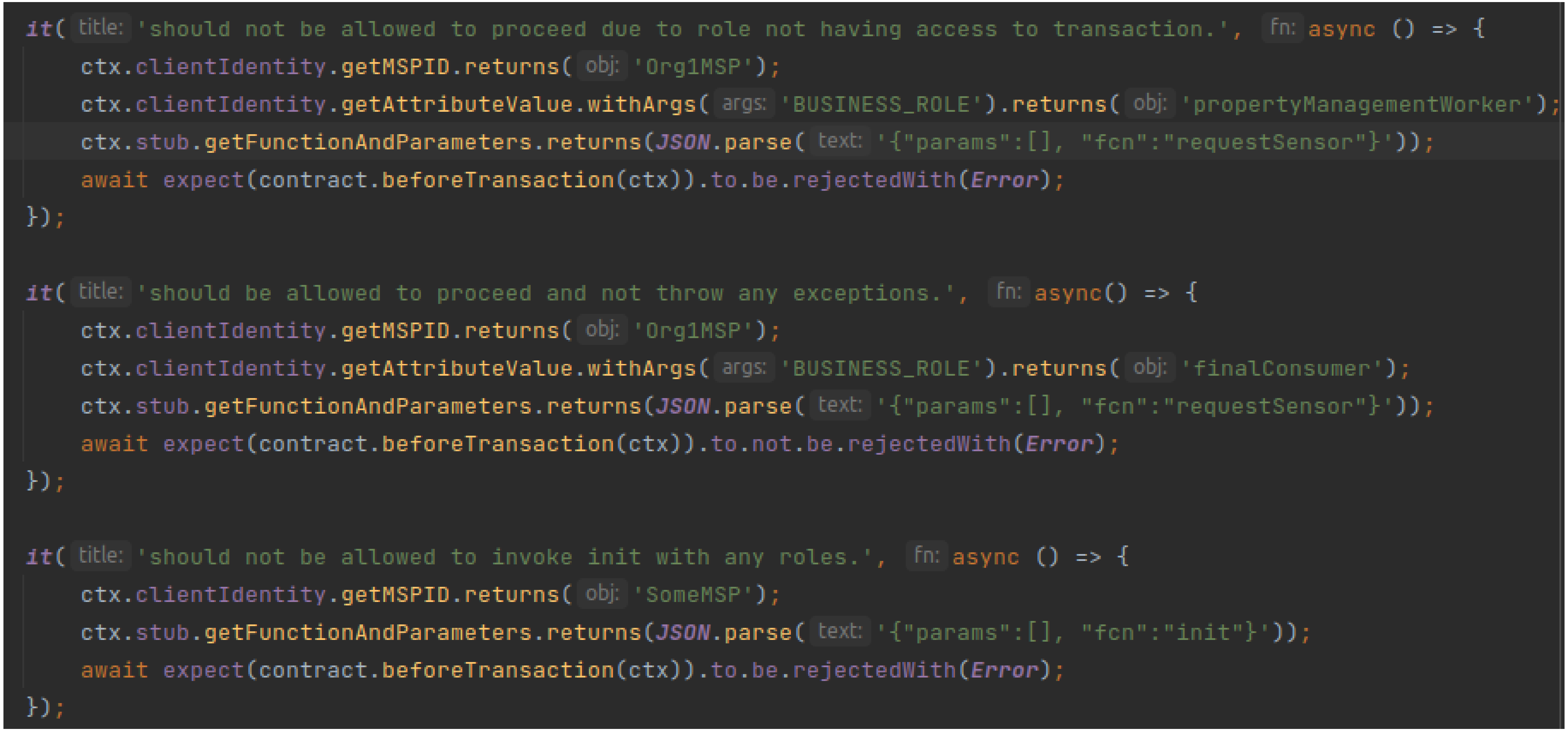
Implementation – smart contracts



```
@Transaction()
public async requestSensor(ctx: Context, sensorID: string, approverMSP: string, value: number): Promise<void> {
    const exists = await this.exists(ctx, sensorID);
    if (exists) {
        throw new Error(`The sensor ${sensorID} already exists`);
    }
    const sensor = new SensorAgreement();
    sensor.sensorID = sensorID;
    sensor.consumerMSP = ctx.clientIdentity.getMSPID();
    sensor.consumer = ctx.clientIdentity.getID();
    sensor.approverMSP = approverMSP;
    sensor.approver = null;
    sensor.value = value;
    sensor.status = 'REQUESTED';

    const buffer = Buffer.from(JSON.stringify(sensor));
    await ctx.stub.putState(sensorID, buffer);
}
```

Implementation – smart contracts unit tests

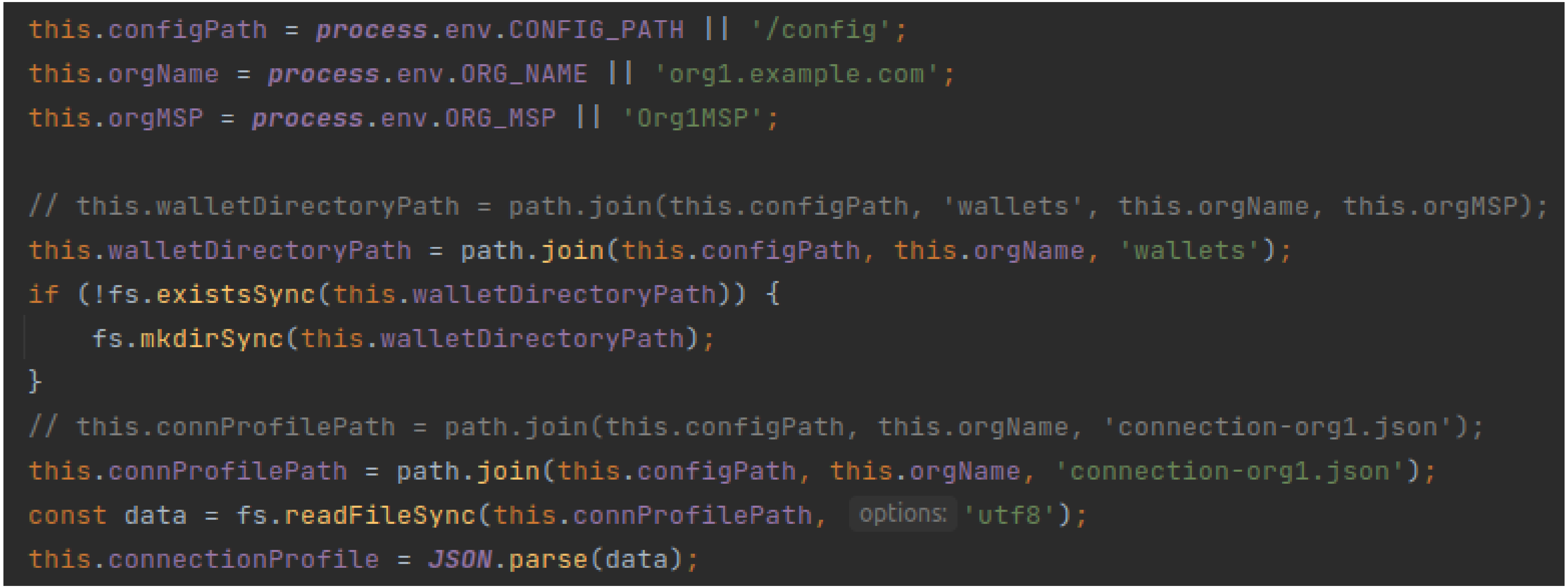


```
it( title: 'should not be allowed to proceed due to role not having access to transaction.', fn: async () => {
    ctx.clientIdentity.getMSPID.returns( obj: 'Org1MSP');
    ctx.clientIdentity.getAttributeValue.withArgs( args: 'BUSINESS_ROLE').returns( obj: 'propertyManagementWorker');
    ctx.stub.getFunctionAndParameters.returns(JSON.parse( text: '{"params":[], "fcn":"requestSensor"}'));
    await expect(contract.beforeTransaction(ctx)).to.be.rejectedWith(Error);
});

it( title: 'should be allowed to proceed and not throw any exceptions.', fn: async() => {
    ctx.clientIdentity.getMSPID.returns( obj: 'Org1MSP');
    ctx.clientIdentity.getAttributeValue.withArgs( args: 'BUSINESS_ROLE').returns( obj: 'finalConsumer');
    ctx.stub.getFunctionAndParameters.returns(JSON.parse( text: '{"params":[], "fcn":"requestSensor"}'));
    await expect(contract.beforeTransaction(ctx)).to.not.be.rejectedWith(Error);
});

it( title: 'should not be allowed to invoke init with any roles.', fn: async () => {
    ctx.clientIdentity.getMSPID.returns( obj: 'SomeMSP');
    ctx.stub.getFunctionAndParameters.returns(JSON.parse( text: '{"params":[], "fcn":"init"}'));
    await expect(contract.beforeTransaction(ctx)).to.be.rejectedWith(Error);
});
```

Implementation – Fabric Node SDK



```
this.configPath = process.env.CONFIG_PATH || '/config';
this.orgName = process.env.ORG_NAME || 'org1.example.com';
this.orgMSP = process.env.ORG_MSP || 'Org1MSP';

// this.walletDirectoryPath = path.join(this.configPath, 'wallets', this.orgName, this.orgMSP);
this.walletDirectoryPath = path.join(this.configPath, this.orgName, 'wallets');
if (!fs.existsSync(this.walletDirectoryPath)) {
    fs.mkdirSync(this.walletDirectoryPath);
}
// this.connProfilePath = path.join(this.configPath, this.orgName, 'connection-org1.json');
this.connProfilePath = path.join(this.configPath, this.orgName, 'connection-org1.json');
const data = fs.readFileSync(this.connProfilePath, options: 'utf8');
this.connectionProfile = JSON.parse(data);
```

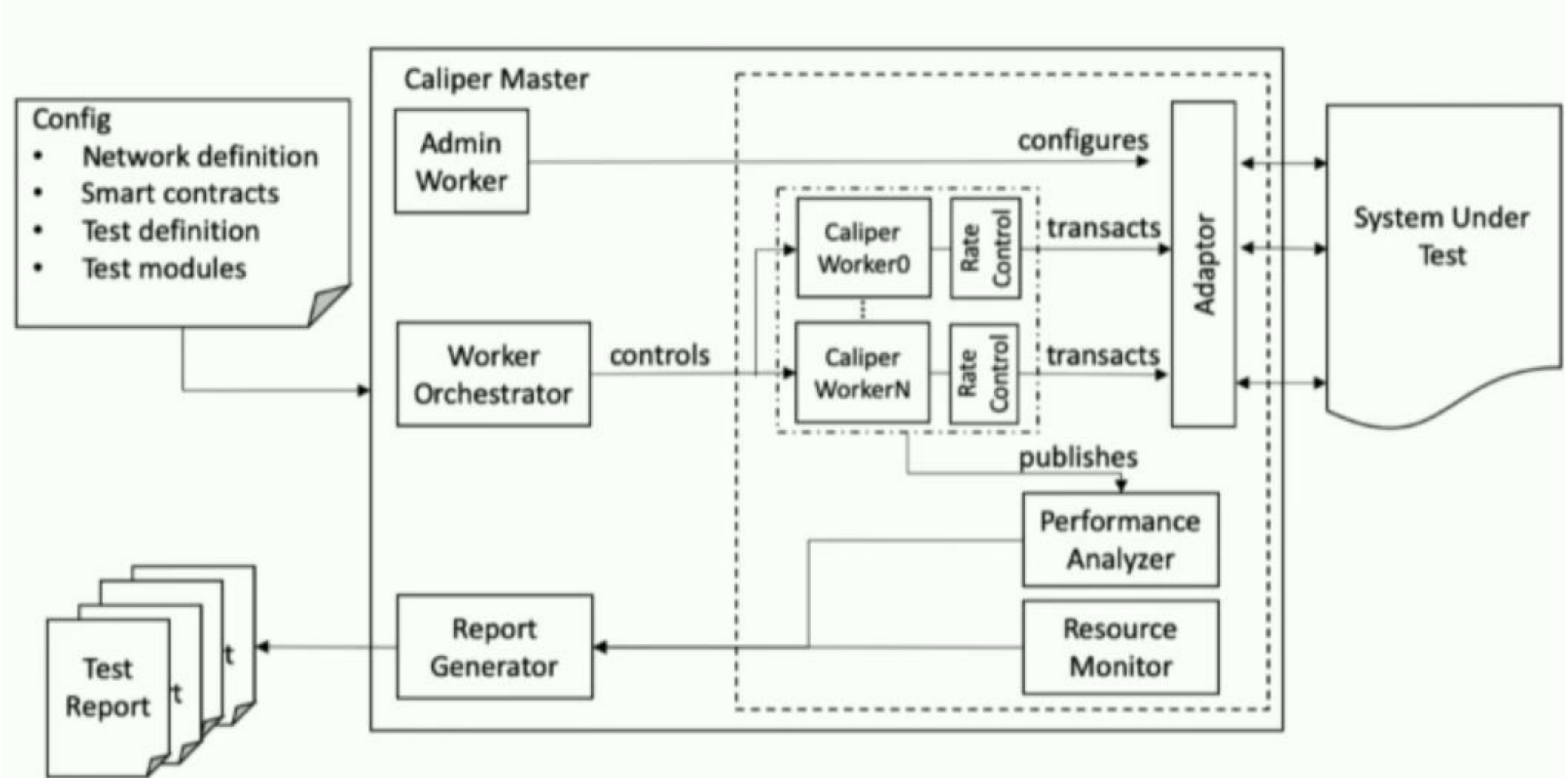
Use case

```
{
  "_id": "sensor5",
  "_rev": "1-96ef88e5ab0ac754686f5e37cd925ef8",
  "approver": null,
  "approverMSP": "Org1MSP",
  "consumer": "x509::/OU=client/CN=finalconsumer::/C=US/ST=North Carolina/L=Durham/O=org1.example.com/CN=ca.org1.example.com",
  "consumerMSP": "Org1MSP",
  "sensorID": "sensor5",
  "status": "REQUESTED",
  "value": 0,
  "~version": "CgMBBgA="
}
```

```
{
  "_id": "sensor5",
  "_rev": "2-1b9fc04d2946ff6ee06a6229472e246c",
  "approver": "x509::/OU=client/CN=propertymanagementworker::/C=US/ST=North Carolina/L=Durham/O=org1.example.com/CN=ca.org1.example.com",
  "approverMSP": "Org1MSP",
  "consumer": "x509::/OU=client/CN=finalconsumer::/C=US/ST=North Carolina/L=Durham/O=org1.example.com/CN=ca.org1.example.com",
  "consumerMSP": "Org1MSP",
  "sensorID": "sensor5",
  "status": "ACCEPTED",
  "value": 0,
  "~version": "CgMBBwA="
}
```

```
{
  "_id": "sensor5",
  "_rev": "4-007cdebb0a079f09bf03c3c913669089",
  "approver": "x509::/OU=client/CN=propertymanagementworker::/C=US/ST=North Carolina/L=Durham/O=org1.example.com/CN=ca.org1.example.com",
  "approverMSP": "Org1MSP",
  "consumer": "x509::/OU=client/CN=finalconsumer::/C=US/ST=North Carolina/L=Durham/O=org1.example.com/CN=ca.org1.example.com",
  "consumerMSP": "Org1MSP",
  "sensorID": "sensor5",
  "status": "ACCEPTED",
  "value": 25,
  "~version": "CgMBCQA="
}
```

Performance testing - Caliper local



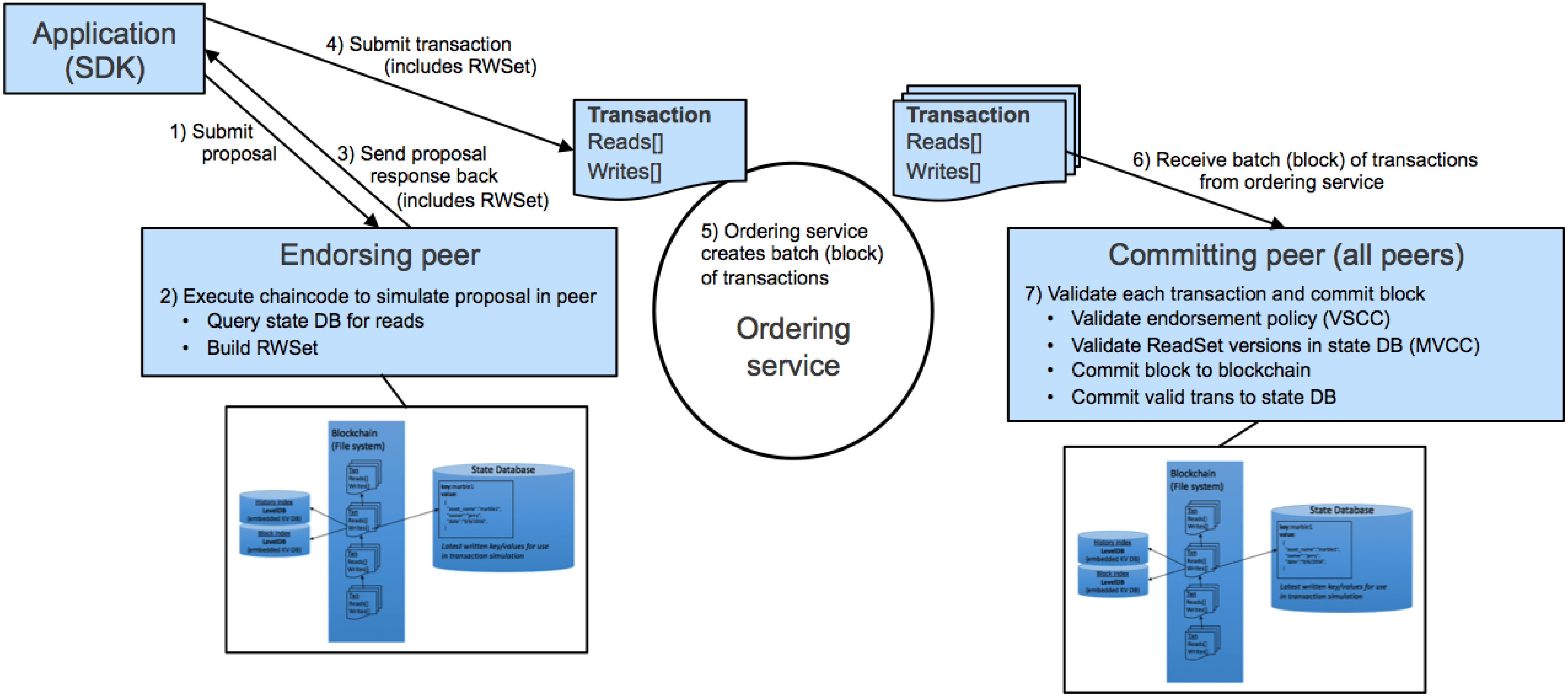
Implementation – smart contracts unit tests

Name	Succ	Fail	Send Rate (TPS)	Max Latency (s)	Min Latency (s)	Avg Latency (s)	Throughput (TPS)
updateSensorFixedLoad	41	197	21.6	11.64	0.54	5.64	13.3

```
warn: [TransactionEventHandler]: strategyFail: commit failure for transaction "bd4…956": Error: Commit of transaction bd4…956 failed on peer peer0.org1.example.com:7051 with status MVCC_READ_CONFLICT
```

Name	Succ	Fail	Send Rate (TPS)	Max Latency (s)	Min Latency (s)	Avg Latency (s)	Throughput (TPS)
createSensorFixedLoad	225	0	15.8	16.94	0.83	10.69	10.3
readSensorFixedLoad	6169	0	208.6	1.30	0.01	0.43	208.5
updateSensorHighThroughputFixedLoad	268	0	24.9	12.37	0.49	7.55	14.7

Productivity, Scalability, and Level of Trust



Application (SDK)

1) Submit proposal

3) Send proposal response back (includes RWSet)

4) Submit transaction (includes RWSet)

Transaction
Reads[]
Writes[]

Endorsing peer

2) Execute chaincode to simulate proposal in peer

- Query state DB for reads
- Build RWSet

5) Ordering service creates batch (block) of transactions

Ordering service

Transaction
Reads[]
Writes[]

6) Receive batch (block) of transactions from ordering service

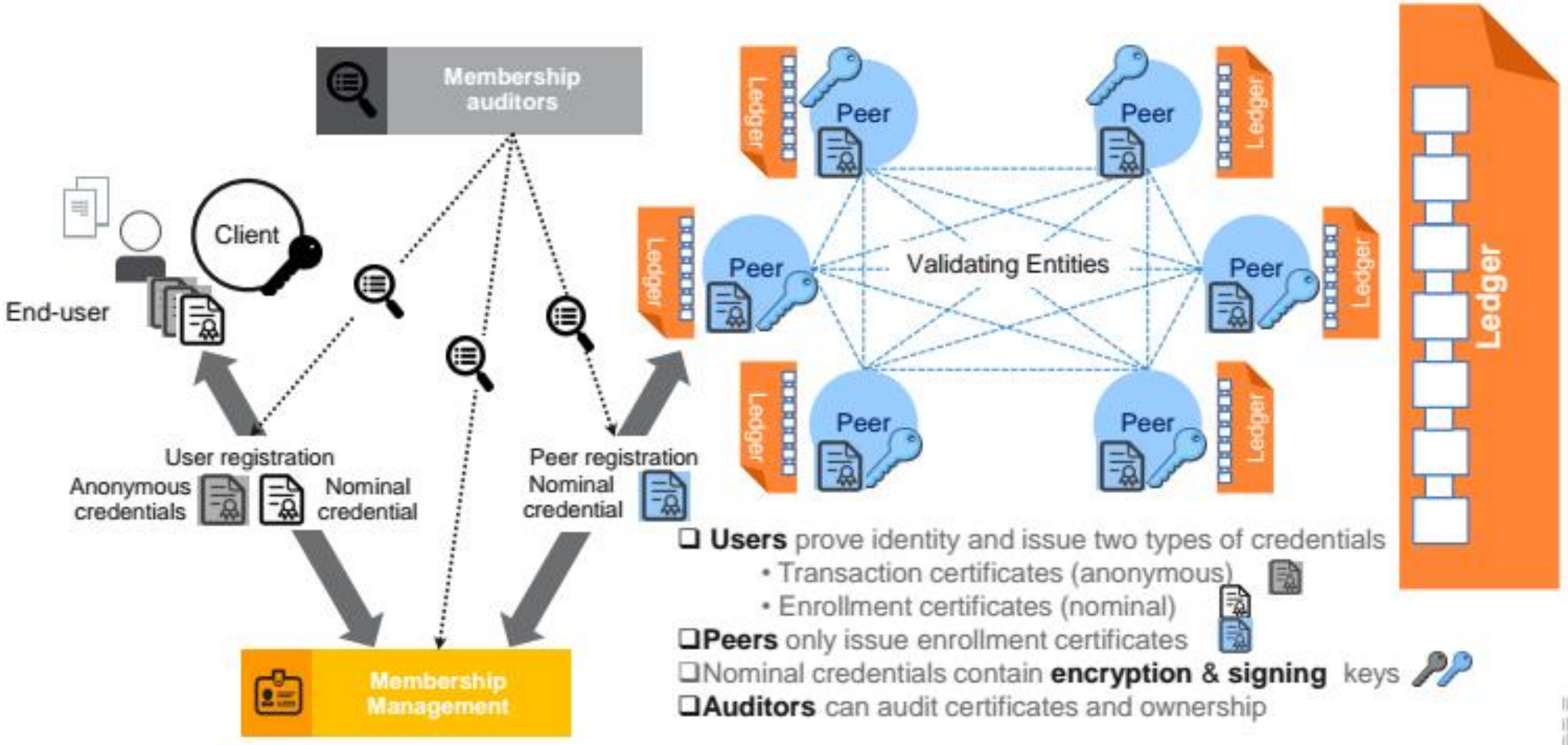
Committing peer (all peers)

7) Validate each transaction and commit block

- Validate endorsement policy (VSCC)
- Validate ReadSet versions in state DB (MVCC)
- Commit block to blockchain
- Commit valid trans to state DB

Key concepts

Membership



- ❑ **Users** prove identity and issue two types of credentials
  - Transaction certificates (anonymous)
  - Enrollment certificates (nominal)
- ❑**Peers** only issue enrollment certificates
- ❑Nominal credentials contain **encryption** & **signing** keys
- ❑**Auditors** can audit certificates and ownership

Zdroje

1. R. Wattenhofer: The Science of the Blockchain, CreateSpace IndependentPublishing Platform, 2016, ISBN-13: 978-1522751830

2. M. Swan: Blockchain: Blueprint for a New Economy, O'Reilly Media, 2015, ISBN-13: 978-1491920497

3. D. Tapscott , A. Tapscott : Blockchain Revolution: How the Technology Behind Bitcoin Is Changing Money, Business, and the World, Portfolio, 2016,ISBN-13: 978-1101980132

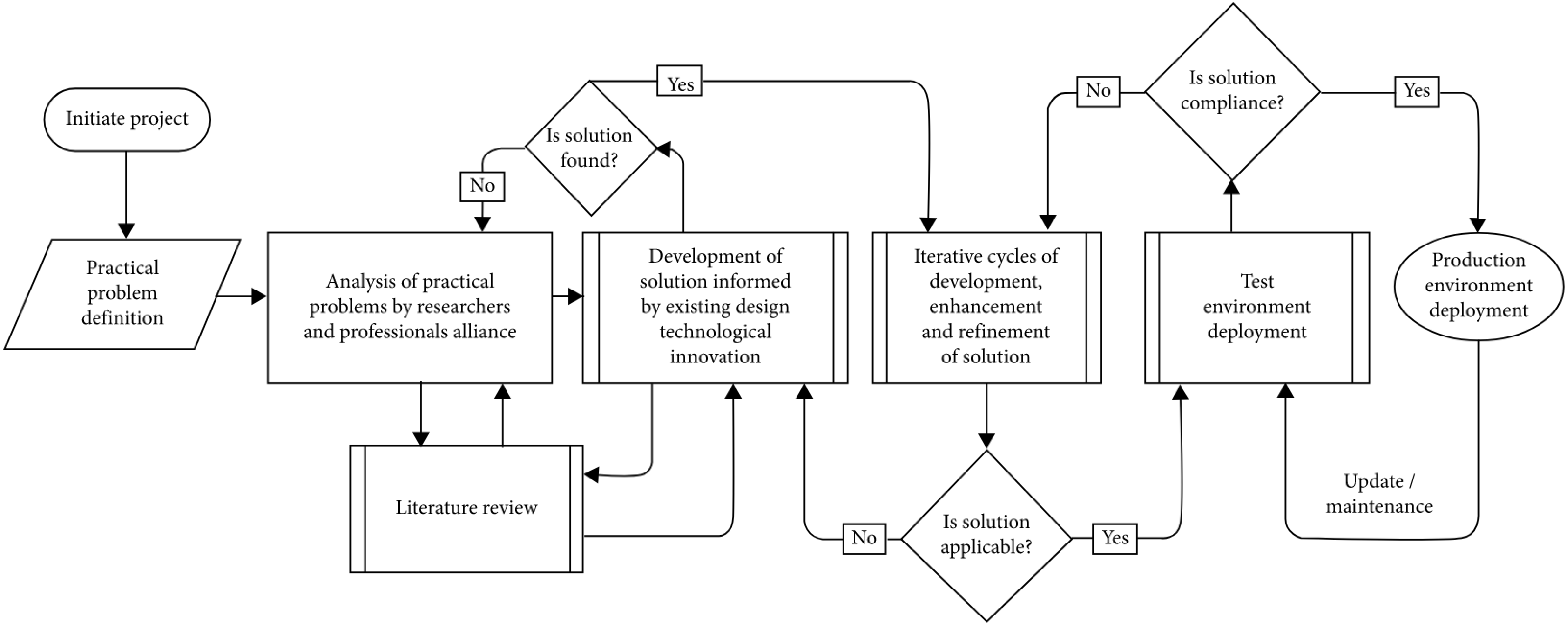
4. Chris Dannen: Introducing Ethereum and Solidity; Foundations of Cryptocurrency and Blockchain Programming for Beginners, Apress, Berkeley, 2017, ISBN 978-1-4842-2534-9, https://doi.org/10.1007/978-1-4842-2535-6

5. Xiaoqi Li, Peng Jiang, Ting Chen, Xiapu Luo, Qiaoyan Wen, A Survey onthe security of blockchain systems, In Future Generation Computer Systems, 2017 , ISSN 0167-739X, https://doi.org/10.1016/j.future.2017.08.020.

6. Steve Huckle, Rituparna Bhattacharya, Martin White, Natalia Beloff, Internet of Things, Blockchain and Shared Economy Applications, In Procedia Computer Science, Volume 98, 2016, Pages 461-466, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2016.09.074.

Thank you for your attention

Methodology


Initiate project
Practical problem definition
Analysis of practical problems by researchers and professionals alliance
Literature review
Development of solution informed by existing design technological innovation
Is solution found?
No
Yes
Iterative cycles of development, enhancement and refinement of solution
Is solution applicable?
No
Yes
Test environment deployment
Is solution compliance?
No
Yes
Production environment deployment
Update / maintenance