Anatomy, Concept, and Design Space of Blockchain Networks

Nguyen Khoi Tran
The University of Adelaide
Adelaide, Australia
nguyen.tran@adelaide.edu.au

M. Ali Babar
The University of Adelaide
Cyber Security Cooperative Research Centre
Adelaide, Australia
ali.babar@adelaide.edu.au

***Abstract*—Blockchain technologies have been increasingly adopted by enterprises to increase operational efficiency and enable new business models. These enterprise blockchain applications generally run on dedicated blockchain networks due to regulations and security requirements. The design process of these networks involves many decisions and trade-offs that impact security, governance, and performance of applications that run on them. The challenge is further exacerbated by the lack of a common architecture and concept map to communicate about blockchain networks, as blockchain technologies tend to use different concepts and architecture. This paper presents a concept map, an anatomy and the principal dimensions of the design space of blockchain networks. We applied the proposed design space in a case study about designing and deploying a blockchain network for an ad-hoc IoT infrastructure. We found that the design space brought structure to the design process and the analysis of design alternatives. The presented concept map, anatomy and design space are intended to help improve the blockchain network design practice and lay a foundation for future research on the design process and deployment automation of blockchain networks.**

***Index Terms*—Blockchain; Architecture; Design Space; Concept; Design**

I. Introduction

Enterprise adoption of blockchain technology is on the rise. On July 3, 2019, HSBC bank conducted a blockchain-based Letter-of-Credit (LC) transaction between Vietnam and Korea. This blockchain-based international transaction was the first between two countries and the 7th globally. The use of Blockchain technology reduced the transaction processing time from 5 - 10 days to merely 24 hours [1]. According to a survey conducted by Deloitte in 2019, 53% of the participating global enterprises stated that that blockchain "will be critical, in our top 5 strategic priorities" and 40% were willing to invest $5 millions or more in new blockchain initiatives in the following twelve months [2].

Intuitively, a *blockchain* is a data structure that comprises blocks of records. Every block contains a cryptography hash of an immediate ancestor block. These hashes form a chain from the first to the last block and impose a chronological order on blocks' content. This chain also helps to detect tampering, as changes in a previous block would render all the following hashes incorrect. To prevent one party from covering its tampering by recalculating the hashes, a blockchain is replicated and maintained by a network of mutually mistrusting peers (i.e., a *blockchain network*). These peers follow a *blockchain protocol*, which governs the process of validating and appending transactions to the blockchain (i.e., *mining*). Some blockchain protocols also execute user-defined codes, which is generally called *"smart contracts"* [3].

Enterprise blockchain applications generally run on dedicated blockchain networks that enterprises deploy and operate. The rise of enterprise blockchain application has increased the demand for knowledge, processes, and techniques for designing and deploying blockchain networks. The existing software architecture research on blockchain has been focusing on the smart contracts [4] and the ways software systems interact with them [5], [6]. From a network perspective, the existing research has been focusing on comparative analyses of blockchain protocols (e.g., [7]), which covers only one aspect of blockchain network design and deployment. Thus, there is a gap for *a design space that captures and characterizes design decisions regarding networking and protocol that underpin blockchain networks*. This design space could inform architects of decisions to be made and possible choices. It could also facilitate further studies on design decisions of the existing blockchain networks to form guidelines, rules and design techniques.

A design space is a mean of codifying software design knowledge. Intuitively, it is a *n-dimensional Cartesian space*, whose dimensions represent architectural design decisions, and values on those dimensions are alternative design choices. Complete system designs are points in the space. Design spaces provide a foundation for analyzing systems to identify good and bad combinations of design choices. Architects can use design space as a guide to identify the required decisions, their alternatives, and their interactions [8], [9]. In the context of a blockchain network, a design space is useful not only for day-to-day practice but also for understanding and analyzing emerging blockchain protocols and platforms, which all claim superior security and efficiency.

This paper outlines *a design space for blockchain networks*. This design space contains 19 dimensions with 51 alternative choices, which form two clusters: network and protocol. The network cluster guide architects through the decisions to design and deploy a network to run a blockchain. The protocol cluster concerns with the blockchain protocol deployed on a network. Whilst most architects would not develop new

blockchain protocols, the protocol cluster helps them to understand the architecture of off-the-shelf protocols and make more informed decisions. As most of the interesting design spaces in practice are too rich to be captured in their entirety, we do not present this as a final and definitive design space. Instead, we intend to capture the principal decisions to guide the practice and advance the understanding of blockchain network design.

The main contributions of this paper are:

- A concept map of blockchain that defines and relates key concepts of blockchain to support a good understanding of blockchain networks and the design space
- An anatomy of blockchain network that lays a foundation for forming and understanding the design space
- A multi-dimensional design space of blockchain network to guide the design and deployment of blockchain networks
- A case study in which we applied the design space to design and deploy a blockchain network on an ad-hoc Internet of Things (IoT) infrastructure.

This paper is organized as follows. Section II presents a concept map of blockchain. Section III presents the internal structure of a blockchain network via a replicated state-machine perspective. Section IV details the dimensions of the blockchain network design space and the method that we applied to reach these results. Section V presents a case study where the design space was applied. Section VI discusses related work and Section VII concludes the paper.

II. Blockchain Concepts

A. What is blockchain?

Blockchains are complex and multi-facet distributed systems. Their underlying technologies interweave decades of innovations across different research fields [10], and their application areas are still being discovered. Therefore, it is hard to capture their essence in a self-sufficient definition. Therefore, we instead provide a brief account of how blockchains and their descriptions evolved.

The blockchain community commonly agrees that blockchain technologies can be organized into three generations. First-generation blockchains, epitomized by Bitcoin [11], focus on providing immutable transaction ledgers to enable cryptocurrency. Therefore, they are generally defined as *a distributed, append-only database* that is controlled by multiple parties who do not necessarily trust each other [12]. First generation blockchains embed the rules and structure of ledger's content in their protocol. Therefore, they have limited programmability, meaning that new blockchain protocols are required for new applications.

Ethereum [3], launched in 2015, introduced programmability to blockchains. It allows users to deploy their business logic as pieces of code, called "smart contracts", on a blockchain. These "smart contracts" enjoy an integrity guarantee by hash-chaining and decentralization similarly to other ledger data. Their execution correctness is guaranteed by the fact that all participants who operate a blockchain network run smart contracts and validate their output in parallel. Due to the use of smart contract, these blockchains are generally defined as *replicated, deterministic state machines that transit between states by transactions* [3]. Ethereum and its derivatives are prime examples of second-generation blockchains.

As the awareness about BC and its potentials improved, enterprises became increasingly interested in the technology. However, public blockchains such as Ethereum are not compliant with enterprise requirements such as Know Your Customer (KYC), Anti Money Laundering (AML). Thus, new BC platforms that target enterprise use cases were invented, such as JP Morgan's Quorum, Corda, and Hyperledger Fabric. These BC platforms introduce access control, privacy-preservation, and protocol optimizations. Some BC platforms such as Fabric also introduced architectural innovations to blockchain protocols, making them more modular to support further optimisations [13]. Due to these advances, we consider these blockchains as "generation 2.5". For simplicity, we will prefer to these BC as enterprise BC in this paper.

The blockchain community has not quite reach a consensus on the third generation of blockchain. It appears that scalability in terms of the number of transactions per second (TPS) and the energy consumption are chief concerns that are driving the research towards third-generation blockchains.

B. Blockchain Concept Map

Due to its fast development and pollination between many research fields, blockchain has a large number of concepts which architects must understand to design and communicate about blockchain networks. Moreover, some concepts might appear overlapping and ambiguous. To address this challenge, we propose a concept map that captures common terminologies that appear in software engineering activities related to blockchains. Figure 1 depicts this concept map.

A blockchain *network* consists of blockchain *nodes* which belong to one of three variants: *full, lightweight, and remote*. This classification is based on the data that their role in the network. Full nodes can be further organized into mining- and non-mining-nodes. Detailed descriptions of these nodes are available in Section III. *Blockchain Network Developers* are responsible for designing and deploying blockchain networks.

The purpose of a blockchain network is to run *blockchain clients*. These clients implement a *blockchain protocol*, which governs the maintenance and update of a blockchain. This data structure stores transaction *ledgers* and can also host *smart contracts*. A blockchain platform such as Hyperledger Fabric comprises a blockchain client and utilities to operate and manage it. *Blockchain Developers* are responsible for these components.

Distributed Applications (DApp) are software applications that use on-chain ledgers and smart contracts to store data and implement business logic. *Blockchain Application Developers* design and develop DApp.

III. Anatomy of a Blockchain Network

This section outlines the anatomy of blockchain networks. We start by presenting a model of operation of a blockchain

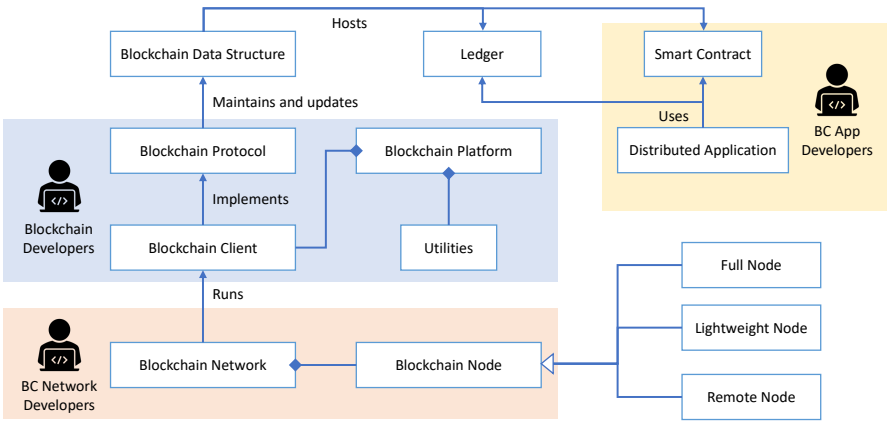

Fig. 1. A map of blockchain concepts and developers

network. Then, we specify software modules that implement the operation and computing nodes that host those modules.

A. Blockchain as a State Machine

We adopt the replicated state-machine model of second-generation blockchains, outlined in the Ethereum Yellow Paper [3], Corda White Paper [14] and the decomposition of consensus protocol presented in [12] as the basis of anatomy. Specifically, we consider a blockchain to be *a replicated, deterministic, transaction-based state machine that runs on a peer-to-peer network*. This state machine can be expressed in terms of its state transition function δ (Eq. 1), where S stands for the set of all states and TX denotes the set of all transaction.

$$\delta : S \times TX \Rightarrow S \tag{1}$$

As a transaction-based state machine, a blockchain moves between states by processing transactions. Formally, given a current state s_i and a transaction tx, the next state of a blockchain is $s_{i+1} = \delta(s, tx)$. Being deterministic means a blockchain produces only one s_{i+1} for each input (s_i, tx). As a consequence, all blockchain nodes that apply the same sequence of transactions to the same genesis state s_0 would arrive at the same final state.

A blockchain's *consensus protocol* ensures that all blockchain nodes arrive at the same final state. This can be done by ordering the transactions broadcasted to blockchain nodes or by selecting among the next states that blockchain nodes come up with and broadcast (i.e., "Nakamoto consensus"). Consensus protocols operate on different assumptions about the faultiness and maliciousness of participants. These assumptions have significant impacts on a network's security, resilience, and performance. For instance, Bitcoin uses a "Proof-of-Work" (PoW) consensus protocol which assumes that all participants are anonymous and untrustworthy. Therefore, it requires any participant who broadcasts a block to solve a costly cryptography puzzle to prevent a party from flooding the network with fake blocks.

Most blockchains also include a *validator* to filter incoming transactions based on their syntactic and semantic correctness before delivering them to nodes via a consensus protocol. Together, a replicated state machine, a consensus protocol, and a validator form a blockchain protocol.

It should be noted that a lot of blockchain literature tend to use "consensus protocol" as an umbrella term to cover all processes that happen under-the-hood of a blockchain. While this abstraction is helpful to gain an initial understanding of blockchains, we argue that architects would benefit from a more nuanced model that clearly distinguishes activities within these systems. For instance, based on such a model, Hyperledger Fabric blockchain introduced modularity into blockchain protocols [13]. Even if architects do not work on protocol design, a detailed model of blockchain's operation is still beneficial as it would help them analysing and selecting protocols for their networks.

B. Module Structure

Based on the operations outlined in the model and extensions made by enterprise blockchains, we identified eight modules of a blockchain network. The upper portion of Figure 2 depicts these modules and their interactions.

State Machine: A state machine embodies a blockchain protocol. It is characterized by the model of the state S and the model that user-defined logic – smart contract – is integrated into the state transition. Common state models are Unspent Transaction Output (UTXO), Account, and Versioned Key-Value Pairs. Common account models are on-chain and installed. These models represent alternative design choices of a blockchain protocol. Thus, we will discuss them further in Section IV.

Virtual Machine: Virtual machines allow a blockchain's state machine to run on various computing nodes with different hardware architecture and software configurations. For instance, Bitcoin uses a simple stack-based machine that is deliberately Turing-incomplete [11]. Ethereum uses a modified stack-based machine called Ethereum Virtual Machine (EVM)

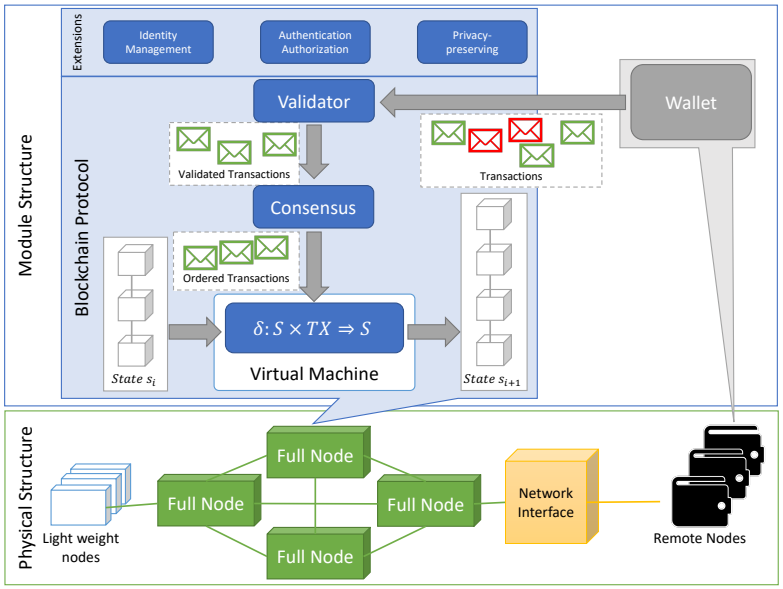

Fig. 2. Anatomy of a Blockchain Network

that allows support Turing completeness. Java Virtual Machine and container technologies, such as Docker, have also appeared in the existing blockchain platforms.

Consensus: The consensus module implements consensus protocols to enable ordered delivery of transactions to state machines. The choice of consensus protocol reflects the fault model that a blockchain network assumes. We will discuss consensus protocols further in Section IV.

Validator: This module implements a transaction validation scheme. At the lowest validation level, a validator only checks whether a transaction has appeared in a previous block. This validation can be done without access to transaction content using transactions' hashes. Higher validation levels are syntactic and semantic validation. These levels require both transaction content and blockchain content.

Wallet: A wallet is a software application that allows end-users to interact with a blockchain network. It stores private keys that control blockchain accounts and creates blockchain transactions. Two key approaches to design wallets are non-deterministic and hierarchical deterministic (HD). A thorough discussion of the wallet is beyond the scope of this paper. Interested readers can find more details in chapter 5 of [15].

The remaining three modules model the extensions made by enterprise blockchain platforms.

Identity Manager: Regulations such as Know Your Customer (KYC) forbids anonymity in enterprise blockchain networks. The Identity Manager module issues and manages certificates to link real-world identities with blockchain accounts. X.509 is the most commonly used standard.

Authentication and Authorisation: This module captures the authentication and authorization mechanisms used by enterprise blockchain networks. For instance, Hyperledger Fabric relies on digital certificates issued by a certificate authority.

Privacy Preserving: This module captures mechanisms to preserve the confidentiality of network participants and transactions. These mechanisms lie at the core of the extensions made by enterprise blockchain platforms. Privacy preservation generally requires both protocol extensions and off-chain components that leverage those extensions.

C. Physical Structure

The physical structure describes computing nodes and networks that host the presented software modules. Even though blockchain networks are peer-to-peer, blockchain nodes are not equal. The lower portion of Figure 2 depicts different types of nodes and their relationships.

Full Nodes: A full blockchain node holds an entire blockchain. Thus it can verify transactions and execute smart contracts independently. A blockchain network is maintained and operated by full nodes. The number of full nodes reflects a network's degree of decentralization. A trade-off of full nodes is that they are resource-intensive. For instance, by September 2018, hosting a full Ethereum node requires 80 to 100GB of fast storage [15].

Lightweight Nodes: A lightweight node holds block headers instead of the entire blockchain. These headers contain a cryptographic representation of the corresponding blocks in the form of a Merkle root. With the support of a full node, a lightweight node can use the Merkle root to validate whether a transaction belongs to a blockchain. Without a full node, however, a lightweight node cannot operate. In other words, lightweight nodes trade independence and security for reduced resource consumption.

Remote Nodes: A remote node does not store blockchain data nor validate transactions. Instead, it hosts the wallet software, which allows users to create, sign, and broadcast transactions. It interacts with blockchain networks via network interfaces.

Network Interface: A network interface node hosts an API for remote nodes to interact with a blockchain network. This node can be embedded in a full or lightweight blockchain node. Alternatively, it can be offered by a third party, such as Infura[1].

Networking-wise, blockchain nodes generally connect using TCP/IP connections and use gossip protocols to broadcast messages. For instance, transaction propagation in an Ethereum network is as follows [15]:

1) A client broadcasts its transaction to some blockchain nodes.
2) Nodes validate and broadcast the transaction to their peers who have not received the transactions.
3) Peer nodes repeat the process.

After several minutes, all full nodes should have received the transaction.

IV. Dimensions of the Design Space

A. Method

We analyzed the documentation and operational guides of four prominent blockchain platforms – Ethereum, Hyperledger Fabric, Quorum, and Corda – to identify dimensions and choices that make up a design space of blockchain network. We chose these blockchain platforms because they have wide adoption, and most of them include extensions for enterprise use cases. These features make them adequately representative to analyze and develop a design space for blockchain networks.

We analyzed blockchain network architecture on a network- and a protocol-level. Protocol wise, we analyzed whitepapers and other technical descriptions of the chosen protocols to extract their module structures. We then mapped the modules onto the module structure presented in Section III to identify alternative design choices for each module and amend missing modules to the structure.

Network wise, we analyzed the instructions to deploy a private blockchain network using the chosen platforms and identified decisions that impact the architecture of the network. To demonstrate our method, we present a walk-through example of identifying design decisions based on the process to deploy a private Ethereum network using its Go-based client (`geth`):

1) Building a genesis block of the blockchain network. Architects can choose between Proof-of-Work (PoW) and Proof-of-Authority (PoA) consensus protocol in this step.
2) Disseminating the genesis block to computing nodes that would participate in the network. Architects have to decide on how to map blockchain nodes to network participants and where to deploy those nodes.
3) Setting up a mechanism for nodes to discover each other. Architects have to choose between building a static network where each node is given full network topography from the beginning or building a dynamic network where nodes interact with a registry (i.e., "bootnode") to find others.
4) Setting up and starting mining to start the network. Architects have to decide on who has the right to mine (i.e., adding data to the network).

Based on this network deployment process, we identified five design decisions: consensus protocol, nodes-to-participants mapping, node deployment, node discovery mechanism, and mining right distribution. By applying this process to all chosen blockchain protocols and find the union of their design decisions, we came up with the design space of a blockchain network.

B. Protocol Design Decisions

Protocol decisions group consists of eight dimensions, which are further organized into three sub-groups (Table I).

State-machine decisions: This group covers state model, smart contract model, validator, and virtual machine of a blockchain protocol. Virtual machine and validator decisions are self-explanatory based on Section III.

State Models describe the form of the state information maintained on a blockchain. Common choices are Unspent Transaction Output (UTXO), Account, and Versioned Key-Value Pairs. In the *UTXO model*, each transaction contains addressable inputs and outputs. Every input is an output of a prior transaction. The state of a UTXO blockchain is, therefore, a list of transaction outputs that have not been used. In the *Account model*, a blockchain holds a list of accounts that can be debited and credited. Ethereum and its derivatives utilize this model. In the *Versioned Key-Value Pairs model*, a blockchain holds $(key, values, version)$ tuples, where $version$ is a monotonically increasing integer counter. Hyperledger Fabric uses this model. The choice of state model can determine how easy to understand a protocol is.

Smart Contract Models describe how smart contracts are integrated into the state transition process. *On-chain contracts* are stored directly on a blockchain. Each contract controls an account, from which it receives invocations. *Installed contracts*, on the other hand, are stored on blockchain nodes for future invocation. Some enterprise blockchains, including Corda and Hyperledger Fabric, use installed contracts. This model allows for more control over the placement of smart contracts, which can be used as a privacy-preserving mechanism. On-chain contracts, on the other hand, allows for a more decentralized operation and integrity guarantee.

Consensus-related Decisions: Decisions in this group reflect assumptions of architects about the threat and failure model of a network. When network participants are known and can

[1]https://infura.io

TABLE I
PROTOCOL DECISIONS OF A BLOCKCHAIN NETWORK

Group	Decision	Choices
State-machine	State Model	UTXO, Account, Key-value
	Smart Contract Model	None, On-chain, Installed
	Virtual Machine	EVM, JVM, Docker
	Validator	None, Uniqueness, Syntax-only, Semantic
Consensus	Consensus Protocol	Byzantine Fault and Sybil Tolerant, Byzantine Fault Tolerant, Crash-tolerant
Enterprise extentions	Identity Model	None, X.509 certificate
	Authentication and Authorisation	None, Digital Certificate
	Privacy Preserving	None, State segmentation, Private Channel

only fail by crashing (i.e., stop responding), a blockchain network can utilize *crash-fault tolerant* consensus protocols such as Raft. When participants are known but can behave unexpectedly, the operating environment becomes a Byzantine environment, and blockchain networks must use *Byzantine Fault Tolerant (BFT)* consensus protocols. When participants are anonymous and Byzantine, the network must also protect itself against Sybil attacks in which an attacker controls multiple identities at the same time to alter the consensus. *Proof-of-Work (PoW), Proof-of-Stake (PoS)*, and their derivatives are common solutions for this problem. These protocols mitigate Sybil attacks by making state broadcasts costly either by forcing participants to expense their resources in solving a cryptography problem (PoW) or their wealth in securing a stake (PoS).

Enterprise extensions: This group covers decisions on identity model, authentication and authorization mechanism, and privacy-preserving mechanism. The first two are self-explanatory based on Section III.

Privacy-preserving mechanisms ensure that blockchain content is visible only to the targeted recipients. One approach is to divide blockchain data into encrypted subsets while keeping a blockchain network intact. Quorum uses this option. An alternative design is to segment a blockchain network into multiple subnets that serve different groups of participants. These subnets can operate on shared computing and network resources. Users of this design are Hyperledger Fabric and Corda.

C. Network Design Decisions

Network design decisions consist of 11 dimensions, divided into five groups (Table II).

Full nodes decisions: The first decision of this group focuses on the mapping between full nodes and network participants. Options include one-to-many, one-to-one, many-to-one, and public. The public option is used in public networks where architects do not control the deployment of full nodes. One-to-one and many-to-one mapping indicate that each blockchain participant maintains at least one full node to keep a local copy of the blockchain data and participate in the network operation. One-to-many mapping makes sense in case many network participants do not have the authorization nor resources to maintain a full node and therefore opt to share a full node. Taken to the extreme, one can design a "blockchain network" with only one node, shared by all participants. Merits of this design are up to the debate.

The second decision is about the deployment locations of full nodes. Choices include hosting nodes on the same cloud, different public clouds, a hybrid infrastructure consisting of public and private clouds, or solely on networked computers.

The third decision is concerns with the distribution of mining rights. This decision determines the degree of decentralization and the governance of a network. Options include all, one, M-out-of-N, and public. The public option is used in public networks where architects leave the mining decision to participants.

Lightweight nodes decisions: The second group of dimensions concern with lightweight nodes. As we mentioned previously, lightweight nodes trade the security and independence of full nodes for a reduction in resource requirement. Decisions in this group include the number of lightweight nodes and their deployment locations. These decisions are highly dependent on the operating context of a network. Therefore, no generalization of the choices of these decisions can be drawn based on the current practice.

Remote nodes decisions: The third group comprises two decisions concerning remote nodes. As we mentioned previously, remote nodes are software clients through which end-users interact with a blockchain network. The first decision regarding remote nodes is about assigning them to network participants. Options include one-to-many, one-to-one, and many-to-one and public, similarly to the full node's mapping. The second decision is about where remote nodes are deployed. Remote nodes can be standalone software systems or embedded into end-users' applications using platform-provided libraries such as Web3[2].

Network interface decisions: The fourth group designs the interface between blockchain networks and remote nodes. This group consists of two decisions: type and deployment of network interfaces. We identified three interface types: Remote Procedure Call (RPC), Inter-process Communication (IPC), and Serverless functions. Ethereum and its derivatives

[2]https://github.com/ethereum/web3.js/

TABLE II
NETWORK DECISIONS OF A BLOCKCHAIN NETWORK

Group	Decision	Choices
full node decisions	full node – Participant Mapping	One-to-many, one-to-one, many-to-one, public
	Full node deployment	Same cloud, multiple clouds, hybrid clouds, networked computers
	Mining distribution	All, one, M-out-of-N, public
Lightweight node decisions	Number of lightweight node	Context dependent
	Lightweight node deployment	Context dependent
Remote note decisions	Remote node – Participant Mapping	One-to-many, one-to-one, many-to-one, public
	Remote node deployment	Standalone, Embedded in app
Network interface decisions	Interface type	RPC API, IPC API, Serverless functions
	Interface deployment	Standalone, Embedded in full node, Embedded in Lightweight node
Node communication decisions	Peer communication protocol	Gossip-based, Unicast
	Node discovery	Static peer list, Dynamic via gossip protocols, Dynamic via registries

offer both RPC and IPC APIs. Both types allow a remote node to send instructions to a blockchain node. However, IPC requires both nodes to reside on the same host. Thus, IPC is more secure but less convenient comparing to RPC. Some cloud-based blockchain services offer serverless functions as wrappers around these APIs to further abstract away the complexity of blockchains.

The second decision of this group focuses on the deployment of network interfaces. One option is to deploy network interfaces on standalone servers. Alternatively, interfaces can be hosted directly by full nodes or lightweight nodes of a blockchain network.

Node communication decisions: The first decision concerns with the protocol that nodes use to message others. The existing blockchain protocols use gossip-based broadcast and unicast. Many blockchain protocols use gossip-based broadcast. For instance, Ethereum uses a gossip protocol called Ethereum Wire Protocol (ETH)[3] to exchange blockchain information between its nodes. Hyperledger Fabric also implements a gossip data dissemination protocol[4]. The second decision related to node communications is the mechanism for nodes to discover each other. Three alternatives are static network specification, dynamic discovery via gossiping, and dynamic discovery via registry nodes. Some blockchain protocols, such as Ethereum, support all three options.

V. CASE STUDY: BLOCKCHAIN-AT-EDGE

We applied the proposed design space in the design and deployment of a real-world blockchain network and analyzed its impact on the design process. The authors participated in the case study as both architects and observers. We maintained a journal of thoughts, reasoning, and decisions throughout the project. We analyzed the records at the end to identify design decisions, their rationale, and their relation to the design space. We also conducted a performance analysis of the resulting blockchain network to quantify the quality of the design. In this section, we present the context of the blockchain network that we designed in the case study. Then, we present a walk-through analysis of the design process. Finally, we discuss the strengths and weaknesses of the design space that we observed when applying it to a complex network.

A. Blockchain-at-Edge

Information superiority leads to more timely and informed decisions, which differentiate between success and perish in critical missions such as emergency responses. Information systems have been applied to create and maintain information superiority since the 1990s. In recent years, there has been an increasing trend of integrating IoT devices to information systems to contribute real-time, fine-grained information about operating environments.

This case study focused on an ad-hoc IoT infrastructure that can be deployed in-situ to monitor an operating environment. This infrastructure is expected to operate autonomously. It also cannot rely on the communication infrastructure at the deployment site, as it might be down in the event of an emergency. Figure 3 depicts the ad-hoc IoT infrastructure that we used in the case study. It has been deployed in an emergency response exercise conducted in South Australia in December 2019. This infrastructure includes:

- *Wireless sensor nodes* (TI CC2650) which collect and transfer environmental data wirelessly to gateways
- *Gateways* which are battery-powered, single-board computers (Raspberry Pi 3)
- *Aggregator* which is a small-form-factor computer (Intel NUC Hades Canyon) responsible for coordinating gateways and archiving their sensor data

The demand for integrity assurance and provenance of sensing data drove the adoption of blockchain in the ad-hoc IoT infrastructure. A blockchain network could provide both secure storage and code execution to implement various integrity assurance schemes. As Internet uplink is not guaranteed to the IoT infrastructure, this blockchain network must run directly on the infrastructure instead of remote servers. Our focus in this case study is to design and deploy this blockchain network.

[3]https://github.com/ethereum/devp2p/blob/master/caps/eth.md
[4]https://hyperledger-fabric.readthedocs.io/en/release-1.4/gossip.html

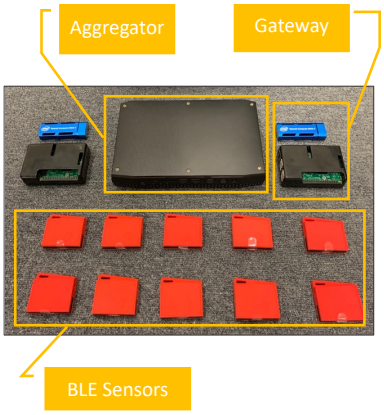

Fig. 3. An ad-hoc IoT infrastructure

B. Design Process

We started the design by identifying the requirements and participants of the blockchain network. Then, we decided on the protocol to run the network. Finally, we decided on various network-aspects outlined in the design space.

Requirements: Based on the operating context and available hardware of the ad-hoc IoT infrastructure, we identified four requirements:

1) Supporting smart contracts
2) Allowing nodes to continue adding to the blockchain when they are disconnected from each other
3) Consuming minimal CPU time and memory, as some nodes are battery-powered
4) Supporting cross-operation between multiple instruction set architectures (ISA)

Network Participants: Determining participants of a blockchain network is crucial to the design process because most network-related design decisions require knowing who participants are. We decided to consider gateways and aggregators as participants. The rationale is that these nodes handle the routing and storage of sensor data. In the event of network segmentation, they would continue operating and thus need to interact with the blockchain network. Making these nodes network participants offers the optimal trade-off between the level of detail and complexity. An alternative is to make every sensor node a participant. This design increases the complexity while not contributing much because sensors still rely on gateways to route sensor data and interact with the blockchain.

Protocol Decision: The first and fourth requirements dictate that the blockchain protocol of our network must support smart contracts and run on both AMD64 architecture of the aggregator and ARM-v7 architecture of the gateways.

We considered Ethereum, Corda, Quorum, and Fabric as choices for the blockchain protocol. All of them support smart contracts. Except for Corda, we found evidence and support for running Ethereum, Quorum, and Fabric on ARM-v7 architecture. However, in a performance test where we deploy these protocols on the gateways, we found that a Fabric-based network took a significant amount of time to start and had a low throughput. Thus, we were left with Ethereum and Quorum. Driven by the familiarity of the development team with Ethereum, we chose Ethereum as the protocol for our blockchain network.

Ethereum offers two choices for the consensus protocol: Proof-of-Work (PoW) and Proof-of-Authority (PoA). Because participants are known in the ad-hoc IoT infrastructure, the blockchain network does not need protection against Sybil attack. Thus, we chose PoA as the consensus protocol. A side effect of this decision is that our network requires substantially less computing resources to run, which works well with the third network requirement.

Network Decisions: *Lightweight node or full node?* The third requirement, namely minimizing CPU time and memory consumption, makes the use of lightweight nodes on gateways appear inevitable. However, lightweight nodes depend on full nodes to validate and process transactions. Thus, using lightweight nodes reduces the autonomy of gateways in the event of network segmentation, which is against the second network requirement. Moreover, thanks to the PoA consensus protocol, we can reduce the resource consumption of full nodes substantially by eliminating their intensive puzzle-solving tasks. Therefore, we decided to use only full nodes in our network.

Who can have full nodes? Who can mine? Based on the second network requirement, we decided that every participant (i.e., gateways and aggregator) maintains one full node and participates in the mining process. This design allows nodes to have access to the complete record of data and continue operating in the event of network segmentation.

Where are remote nodes? Who runs them? Clients who rely on remote nodes to interact with the blockchain network include software agents on gateways that collect sensor data and network monitoring utilities. We decided to embed the functionality of remote nodes directly within clients instead of offering standalone remote nodes to keep the architecture and codebase simple. We opted to use the Web3 client library of Ethereum to implement these capabilities.

What and where are network interfaces? Remote nodes interact with the blockchain network via interfaces. We opted to use both inter-process call (IPC) API and JSON remote procedure call (RPC) API of Ethereum as interfaces. We decided to use only IPC on gateways to avoid exposing them to security risks associated with RPC. For the aggregator node, we opted to use both IPC and RPC as this device can run more powerful firewall programs.

How do nodes communicate? We used the default protocol of Ethereum, called Ethereum Wire Protocol (ETH). This protocol operates on top of the RLPx transport protocol, which is a TCP-based transport protocol.

C. Resulting Design

By applying the stated decisions, we designed and deployed a blockchain-enabled ad-hoc IoT infrastructure depicted in

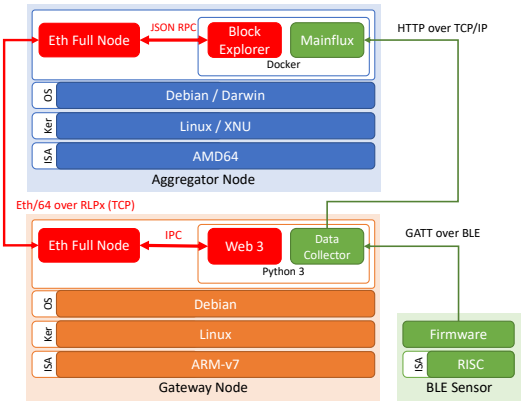

Fig. 4. Architecture of the studied blockchain-enabled IoT infrastructure

Figure 4. We used Raspberry Pi 3B as gateway nodes. We deployed an in-house Python program that uses the Linux Bluetooth protocol stack (`BlueZ`) to connect with and retrieve data from sensors over Bluetooth Low Energy (BLE). This Python program also interacts with a local Ethereum full node via IPC, using the Web3Py library[5].

For the aggregator node, we used a small-form-factor computer equipped with a quad-core AMD64 CPU, 8GB of memory, and onboard wireless interfaces. We deployed a Mainflux[6] server to ingest sensor data. We also deployed a Block Explorer system to visualize and monitor blockchain's content. Finally, to monitor the performance metrics of all nodes in the infrastructure, we used a Prometheus server[7].

Blockchain wise, we used Go-Ethereum version 1.9.7 with a custom genesis file that enables the PoA protocol. On this blockchain network, we deployed a smart contract to store sensor data.

We measured the resource consumption of gateways with and without the blockchain network to assess its effects. Overall, the overheads caused by the blockchain network were minimal. The most substantial change was the disk write rate, which increased by nearly seven-folds to an average of 20.8 kbps. Memory consumption increased from 192.36MB to 249.25MB (average). Network traffic through the wireless interface also risen from 58.0 kbps (max) to 62.7 kbps (max). However, the CPU load does not show a noticeable difference, averaging between 7.25% and 7.36% in both scenarios. Based on these figures, we concluded that the designed network satisfied the requirements.

D. Discussions

We made the following observations throughout the case study. First, the design space gives structure to the design process and the analysis of trade-offs. By listing all principal decisions, the design space provides a checklist that allows architects to go through the design process systematically, decision by decision. The design space also provides a concise model to describe and communicate candidate network designs. The design space was also invaluable when we conducted a trade-off analysis between design alternatives. It highlighted options while keeping us focused on the particular design decision being analyzed. Such a focus can be challenging to achieve when we work with a system with multiple moving and interlocking parts like a blockchain network.

Second, technology familiarity plays a critical role in weighting design options. For instance, we discovered that most popular open-source blockchain platforms support smart contracts and run on ARM architecture, with a varying amount of effort. The tie-breaker that tipped the scale towards Ethereum was the familiarity of the development team with the technology. We hypothesized that the complexity of blockchain platforms and the lack of standardized concepts and network anatomy made the knowledge and skill of developers not immediately transferable between platforms. Thus, it was a risk mitigation strategy to go with the familiar platform.

The design space is not without limitations. First, the interaction between decisions that has not been captured explicitly in the design space. For instance, the decision to use the PoA consensus protocol mitigated the resource consumption issue of full nodes. This decision made the deployment of full nodes on gateways feasible and desirable. Missing implicit interactions among dimensions is an inherent problem of design spaces. Future research could map a large number of blockchain networks to the design space to uncover correlations and interactions among its dimensions empirically.

The second limitation is that some blockchain networks do not align strictly with the design space. For instance, Hyperledger Fabric decomposes mining-enabled full nodes further into peers that run the state machine and orderers that run the consensus protocol. By providing a more fine-grained model, Fabric opens more design decisions for architects, which have not been captured by the design space. Future research could extend the design space with platform-specific dimensions.

The participation of the authors, as architects, presents a threat to the use case's validity. The reason is that we might not face the same challenges as architects who learn and apply the design space for the first time, due to our familiarity with the design space. We mitigated this threat by constraining our design process to use only concepts reported in this paper and publicly available documentation of blockchain platforms to avoid widening the gap. We also limited the scope of our analysis to how the design space modifies the design process instead of how it objectively and quantitative improves the process. Thus, we believe that our observations are still applicable to other projects after architects have familiarised themselves with the design space. Future research could focus on further experimental and empirical studies to draw more conclusions on the impacts of the design space.

[5]https://web3py.readthedocs.io/en/stable/
[6]https://mainflux.readthedocs.io/en/latest/
[7]https://prometheus.io

VI. Related Work

The research relevant to the architectural design of blockchain networks and applications can be organized into three groups based on their focus: *protocol, network, and system*. The protocol-centric research analyzes blockchain protocols and assesses their impacts on quality attributes of a blockchain network. For instance, Dinh et al. [7] modeled and compared 21 blockchain protocols on their application areas, smart contract execution environment, smart contract language, data model, and consensus protocol. Network-centric research assesses the impact of network protocols on the quality attributes of blockchain networks. For instance, Nguyen et al. [16] experimented on the effects of transmission delay on a Hyperledger Fabric network. They introduced an artificial network delay between two nodes in Sorbonne and Heidelberg and calculated the offset between their states. They found that one node can lag for more than one hour behind the others when the network delay is 3.5 seconds. Beyond this point, the network failed. These results are essential to the outlined design space because they provide architects more information to make decisions. The proposed design space helps to form a foundation to incorporate these results into a comprehensive design framework.

The system-centric research studies the integration of blockchain networks into software systems to identify architectural patterns, tactics, and design processes [4]–[6]. Xu et al. [5] proposed to treat blockchain as a type of software connector that enables decentralized and trustless interactions between distributed software components. By treating blockchain as a software connector, architects can make explicit architectural considerations. This idea laid a foundation for software architecture research on blockchain applications, such as the design process [6] and architectural patterns [4]. Our design space, outlined in this paper, does not pose as an alternative to these results, but to complement them, as we target the blockchain networks that host these applications, instead of the application themselves.

VII. Conclusions

Before designing a blockchain network, it is valuable to explore and understand the design decisions to be made and their choices. Such knowledge, captured in a design space, would help developers to systematize their decision making in the design process and trade-off analyses. This design space would also serve as a standard lexicon to describe and evaluate the existing and new blockchain networks. In this paper, we have described the design space of a blockchain network in terms of two dimension-clusters: protocol and network. For each dimension, we have identified choices based on the architectural descriptions and deployment instructions of prominent blockchain platforms, including Ethereum, Quorum, Corda, and Hyperledger Fabric. To support architects to understand and apply the design space, we have also outlined a concept map and anatomy of blockchain networks. Finally, we have applied the design space in designing and deploying a blockchain network for an ad-hoc IoT infrastructure and observe its impact on the design process.

In our future work, we plan to conduct a large scale mapping of existing blockchain protocols and networks onto the design space to generate a large dataset for detecting patterns and hidden interactions among dimensions. We also plan to investigate model-driven engineering approaches to (semi-) automatically transform the proposed design space to platform-specific spaces and deploy the resulting network.

Acknowledgements

The work of M. Ali Babar was supported in part by the Cyber Security Research Centre Limited, whose activities are partially funded by the Australian Government's Cooperative Research Centres Programme.

References

[1] HSBC, "Hsbc executes the first live blockchain letter-of-credit transaction in korea and vietnam," 2019. [Online]. Available: https://www.about.hsbc.com.vn/-/media/vietnam/en/news-and-media/

[2] Deloitte, "Deloitte's 2019 global blockchain survey," *Deloitte Insights*, 2019. [Online]. Available: https://www2.deloitte.com/insights/us/en/topics/understanding-blockchain-potential/global-blockchain-survey.html

[3] G. Wood, "Ethereum: A secure decentralized generalized transaction ledger," *Ethereum Yellow Paper*, 2014.

[4] X. Xu, C. Pautasso, L. Zhu, Q. Lu, and I. Weber, *A Pattern Collection for Blockchain-Based Applications*, 2018.

[5] X. Xu, C. Pautasso, L. Zhu, V. Gramoli, A. Ponomarev, A. B. Tran, and S. Chen, "The blockchain as a software connector," in *2016 13th Working IEEE/IFIP Conference on Software Architecture (WICSA)*, 2016, Conference Proceedings, pp. 182–191.

[6] X. Xu, I. Weber, M. Staples, L. Zhu, J. Bosch, L. Bass, C. Pautasso, and P. Rimba, "A taxonomy of blockchain-based systems for architecture design," in *2017 IEEE International Conference on Software Architecture (ICSA)*. IEEE, 2017, Conference Proceedings, pp. 243–252.

[7] T. T. A. Dinh, R. Liu, M. Zhang, G. Chen, B. C. Ooi, and J. Wang, "Untangling blockchain: A data processing view of blockchain systems," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 7, pp. 1366–1385, 2018.

[8] T. G. Lane, "Studying software architecture through design spaces and rules," 2000.

[9] Y. Brun, R. Desmarais, K. Geihs, M. Litoiu, A. Lopes, M. Shaw, and M. Smit, *A Design Space for Self-Adaptive Systems*. Springer, 2013, pp. 33–50.

[10] A. Narayanan and J. Clark, "Bitcoin's academic pedigree," *Communications of the ACM*, vol. 60, no. 12, pp. 36–45, 2017.

[11] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Bitcoin Whitepaper*, 2008.

[12] C. Cachin and M. Vukolić, "Blockchain consensus protocols in the wild," *arXiv preprint arXiv:1707.01873*, 2017.

[13] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. D. Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, S. Muralidharan, C. Murthy, B. Nguyen, M. Sethi, G. Singh, K. Smith, A. Sorniotti, C. Stathakopoulou, M. Vukoli, S. W. Cocco, and J. Yellick, "Hyperledger fabric: A distributed operating system for permissioned blockchains," in *Proceedings of the Thirteenth EuroSys Conference*. 3190538: ACM, 2018, Conference Proceedings, pp. 1–15.

[14] R. G. Brown, "The corda platform: An introduction," *Corda Whitepaper*, 2018.

[15] A. Antonopoulos and G. Wood, *Mastering Ethereum: Building Smart Contracts and Dapps*. O'Reilly Media, Incorporated, 2018. [Online]. Available: https://books.google.com.au/books?id=SedSMQAACAAJ

[16] T. S. L. Nguyen, G. Jourjon, M. Potop-Butucaru, and K. Thai, "Impact of network delays on hyperledger fabric," *arXiv preprint arXiv:1903.08856*, 2019.