2016 13th Working IEEE/IFIP Conference on Software Architecture

The Blockchain as a Software Connector

Xiwei Xu
NICTA, Sydney, Australia
CSE, UNSW, Sydney, Australia
Xiwei.Xu@nicta.com.au

Cesare Pautasso
Faculty of Informatics
University of Lugano (USI)
via Buffi 13, 6900 Lugano, Switzerland
c.pautasso@ieee.org

Liming Zhu
NICTA, Sydney, Australia
CSE, UNSW, Sydney, Australia
Liming.Zhu@nicta.com.au

Vincent Gramoli
NICTA, Sydney, Australia
University of Sydney, Australia
Vincent.Gramoli@sydney.edu.au

Alexander Ponomarev
and An Binh Tran
NICTA, Sydney, Australia
Alexander.Ponomarev@nicta.com.au
AnBinh.Tran@nicta.com.au

Shiping Chen
CSIRO, Sydney, Australia
Shiping.Chen@csiro.au

***Abstract*—Blockchain is an emerging technology for decentralized and transactional data sharing across a large network of untrusted participants. It enables new forms of distributed software architectures, where components can find agreements on their shared states without trusting a central integration point or any particular participating components. Considering the blockchain as a software connector helps make explicitly important architectural considerations on the resulting performance and quality attributes (for example, security, privacy, scalability and sustainability) of the system. Based on our experience in several projects using blockchain, in this paper we provide rationales to support the architectural decision on whether to employ a decentralized blockchain as opposed to other software solutions, like traditional shared data storage. Additionally, we explore specific implications of using the blockchain as a software connector including design trade-offs regarding quality attributes.**

***Index Terms*—Blockchain; Architecture connector; Design; Trade-off**

I. Introduction

Blockchain is an emerging technology that enables new forms of distributed software architectures, where components can find agreements on their shared states for decentralized and transactional data sharing across a large network of *untrusted* participants without relying on a central integration point that should be *trusted* by every component within the system .

The blockchain data structure is a timestamped list of blocks, which records and aggregates data about transactions that have ever occurred within the blockchain network. Thus, the blockchain provides an immutable data storage, which only allows inserting transactions without updating or deleting any existing transaction on the blockchain to prevent tampering and revision. The whole network reaches a consensus before a transaction is included into the immutable data storage. The next writer of new records on the immutable data storage is decided via different mechanisms, for example, Proof-of-work or Proof-of-stake [24].

The first generation of blockchain is a public ledger for monetary transactions with very limited capability to support programmable transactions. A typical type of applications is cryptocurrency [24]. Cryptocurrency is a digital currency that is based on peer-to-peer network and cryptographic tools. Cryptocurrencies are low-cost and inherently independent of any centralized authority to transfer virtual money or issue new units of money. New units of money are issued by the users of the cryptocurrency through mining. The virtual money can be transferred among peer-to-peer users without going through a trusted authority to purchase goods and services in real world. Bitcoin is the first and most widely used cryptocurrency.

The second generation of blockchain became a generally programmable infrastructure with a public ledger that records computational results. Smart contracts [20] were introduced as autonomous programs running across the blockchain network and can express triggers, conditions and business logic to enable complicatedly programmable transactions. Smart contracts are more versatile than simple currency transactions.

The design of a blockchain-based system has not yet been systematically explored, and there is little understanding about the impact of introducing the blockchain in a software architecture. In this paper, we discuss our experience obtained from applying the blockchain into a number of projects, which resulted in operational prototypes we built using readily available blockchain techniques. The prototypes included in this paper are 1) a decentralized trading market for data sharing, and 2) a platform for participating organisations to securely negotiate and store sensitive data values, which represents a scenario of secure data exchange and negotiation.

Based on this experience, from an architectural perspective, according to the taxonomy of software connectors [16], we propose to consider the blockchain as a novel kind of software connector, which should be considered as a possible decentralized alternative to existing centralized shared date storage. Such view helps us make explicitly important architectural considerations on the resulting quality attributes of the applications. We found that using the blockchain as a software connector could improve information transparency and traceability. However, the mining mechanism increases the communication latency, which might cause poor user

978-1-5090-2131-4/16 $31.00 © 2016 IEEE
DOI 10.1109/WICSA.2016.21

experience. Likewise, the amount of data that can be stored on the blockchain is very limited, thus making it important to decide which data (or meta-data) should be stored on-chain vs. off-chain.

The paper proceeds by introducing background information about the blockchain in Section II, followed by discussing blockchain from an architecture perspective in Section III. Section IV compares the blockchain with existing software connectors. Section V discusses the detailed architecture of our prototypes using blockchain as a software connector. Section VI enumerates the lessons learned from our experience, before Section VII concludes the paper.

II. Blockchain

A. Background

Initially, the blockchain was the key technique behind Bitcoin [19]. The blockchain is a public ledger maintained by all the nodes within the cryptocurrency network. The blockchain stores all the transactions that have ever occurred in the cryptocurrency system. Later, the concept was generalized to a distributed ledger that exploits the blockchain to verify and store transactions without needing cryptocurrency or tokens [27].

The blockchain network does not rely on any central trusted authority, which has the power to control the system, like traditionally centralized banking and payment systems. Instead, trust is achieved as an emergent property from the interactions between nodes within the network. In this paper, we use *blockchain* to refer to the data structure replicated on the nodes and *blockchain network* to refer to the infrastructure composed of a decentralized peer-to-peer network of nodes.

Blocks and transactions are the two essential elements making up the blockchain. Seen as a data structure, the blockchain is an ordered list of blocks. Blocks are the containers aggregating transactions. Every block is identifiable, and linked back to its previous block in the chain.

Transactions represent state transitions with ownership information, which could include new data records and transfer of control among participants. The transactions in cryptocurrencies are the data structures that encode the monetary value being transferred between accounts. More generally, such as in Ethereum, the transactions are a set of identifiable data packages that store monetary value, code, and/or parameters and results of function calls. The integrity of the transactions is ensured by cryptographic techniques.

Once created, the transaction is signed with the signature of the transaction's initiator, which indicates the authorization to spend the money, create the contract, or pass the data parameters associated with the transactions. If the signed transaction is properly formed, it is valid and contains all the information needed to be executed.

The transaction is sent to a node connected to the blockchain network, which knows how to validate the transaction. The invalid transactions are discarded, while the valid transactions are propagated to another three to four other connected nodes, which will further validate the transactions and send them to their peers untill the transaction reaches every node in the network.

This flooding approach guarantees that a valid transaction will reach the whole network within few seconds. The senders do not need to trust the nodes they use to broadcast the transactions, as long as they use more than one to ensure that the transaction propagates. The recipients do not need to trust the senders either because the transactions are signed and contain no confidential information or credentials such as private keys.

When a transaction reaches a mining node, it is verified and included in a block, which is propagated to the network. The block is chained into the blockchain once the whole network reaches a consensus. Once recorded on the blockchain and confirmed by sufficient subsequent blocks, the transaction becomes a permanent part of the public ledger and is accepted as valid in principle by all nodes within the blockchain network.

Cryptocurrencies	
Bitcoin [19]	https://bitcoin.org/
Peercoin	http://peercoin.net/
Colouredcoins	http://coloredcoins.org/
Omni	http://www.omnilayer.org/
Nxt	http://nxt.org/
Smart contract platforms	
Etheruem	https://www.ethereum.org/
Counterparty	http://counterparty.io/
Ledger platforms	
Factom	http://factom.org/
Ripple	https://ripple.com/
Eris	https://erisindustries.com/
MultiChain	http://www.multichain.com/
Enigma	http://enigma.media.mit.edu/

Table I: Examples of blockchain applications and platforms

B. Blockchain Applications and Platforms

Table I gives some examples of blockchain platforms that use the blockchain at the core of their architecture.

1) Cryptocurrency: Cryptocurrency uses cryptography to control the monetary issuance and secures the transaction. The first cryptocurrency, Bitcoin, created in 2009, is still the most widely-used cryptocurrency [1]. Bitcoin allows developers to add 40 bytes of arbitrary data to a transaction, which can be permanently recorded on the blockchain. Thus, the blockchain of Bitcoin has been used to register asset and ownership other than monetary transactions, like in Ascribe[1].

Some cryptocurrencies are overlay networks on Bitcoin, for example, coloured coins, which taints a subset of Bitcoin to represent and manage real-world assets. Other overlay networks completely define new transaction syntax, such as Omni and Counterparty. There are also cryptocurrencies that have their own blockchains built from scratch, such as Nxt. Please refer to [18], [3] and [27] for more comprehensive surveys on the state-of-art of existing cryptocurrencies.

[1]Ascribe — https://www.ascribe.io/

2) Smart contract: Smart contract is the most important element in the second generation of blockchains, which enables a generally programmable infrastructure. The smart contract is deployed and executed on the blockchain network and can be used by the components connected to the blockchain to reach agreements and solve common problems with minimal trust.

There are platforms that allow end users to build self-executing contracts on the Bitcoin blockchain network, for example, smartcontract[2]. The smart contract can still be updated after being submitted and before being propagated to the network. However, smart contracts on the Bitcoin blockchain network are very simple due to the limited expressiveness of the corresponding scripting language, which does not support complex control flow.

Ethereum, as a blockchain-based platform, views smart contract as their first-class element. Ethereum has built its own blockchain from scratch with a built-in Turing-complete script language for writing smart contracts. Counterparty has recreated Ethereum smart contract platform on Bitcoin[3]. The smart contract has been used to enable programmable transactions and machine-to-machine communication in IoT (Internet-of-Things), for example, ADEPT (Autonomous Decentralized Peer-To-Peer Telemetry) project of IBM [10].

III. The Blockchain Connector

A. Software Connector

Software connectors are the fundamental building blocks of software interactions [16]. A connector is an interaction mechanism for the components. Connectors include pipes, repositories, and sockets. For example, middleware can be viewed as a connector between the components that use the middleware [6]. Connectors in distributed systems are the key elements to achieve system properties, such as performance, reliability, security, etc. Connectors provide interaction services, which are largely independent of the functionality of the interacting components [26]. The *services* provided by a software connector could be classified into four categories: *communication*, *coordination*, *conversion* and *facilitation*. *Communication* services transfer data among components while *coordination* transfers control among components. *Conversion* services adjust the interactions to allow components that have not been exactly tailored for each other to establish interactions. *Facilitation* services help to support and optimise components' interactions.

B. Overview

Fig. 1 gives an overview of the blockchain playing the role of software connector. The blockchain is a complex, network-based software connector, which provides communication, coordination (through transactions, smart contracts and validation oracles) and facilitation services [16]. The validation oracle facilitates component coordination within the network

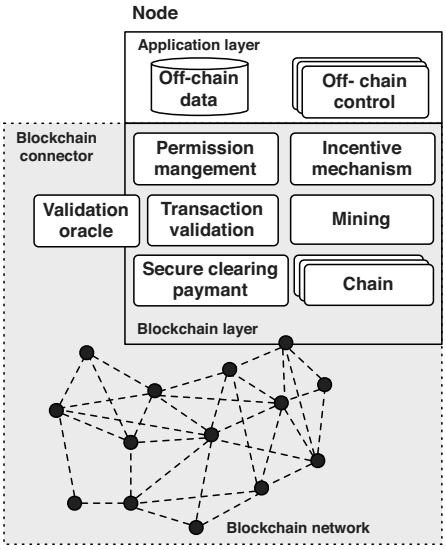

Figure 1: Overview of blockchain as connector

using external, independently managed state. Other facilitation services include cryptography-based secure clearing payment, mining, transaction validation, incentive mechanisms, and permission management.

Every node in the blockchain network has two layers, namely, application layer and blockchain layer. Part of the application is implemented inside the blockchain connector in terms of smart contracts. The part of application outside the blockchain connector might host off-line data and application logic, and interact with the blockchain through transactions. Table II shows some design decisions developers need to consider when using blockchain as a connector and summarizes the corresponding impact on quality attributes.

One of the main architectural decisions for software connector is that what functionality is implemented in the connector and what is implemented in the component. In the case of blockchain, this decision concerns which data and computation should be placed on-chain or kept off-chain (Application Design Decision 1 in Table II). While the blockchain provides a trust-less network that can verify partial computational results and provide agreements on the outcomes of transactions, the amount of computational power and data storage space available on the blockchain network remains limited.

Another decision concerns the access scope of the blockchain: public, private or consortium/community [4] (Application Design Decision 2 in Table II). Most of the cryptocurrencies are built on top of public blockchains, which can be accessed and mined by anyone with Internet access. Using a public blockchain results in better information transparency and audit-ability, but sacrifices information privacy. Consortium blockchain is used across multiple organizations. The consensus process of a consortium blockchain is controlled by authorized nodes. The right to read the blockchain may be public, or restricted to the participants of the blockchain

[2]Smartcontract — http://www.smartcontract.com/

[3]http://counterparty.io/news/counterparty-recreates-ethereums-smart-contract-platform-on-bitcoin/

Table II: Design decisions and quality attribute trade-offs

Blockchain Design Decision 1
Mechanisms of improving transaction processing rate Larger block size; Off-chain transactions; Smaller transaction without signature; Scalable protocol
Blockchain Design Decision 2
Mechanisms of selecting the next block included in the blockchain Proof-of-work, Proof-of-stake, Proof-of-burn, Proof-of-retrievability
Application Design Decision 1
Scope: on-chain Enable verification of computational result, limited computation power and data storage Examples: Metadata (V-A), Negotiable value (V-B). **Scope: off-chain** More computation power and data storage, less cost, additional trust required Examples: Raw personal data (V-A), Sensitive information (V-B)
Application Design Decision 2
Public chain Information transparency, growth potential to larger scale, trustworthy, existing user base Examples: V-A **Private chain** Easier management, better privacy Examples: Consortium blockchain (V-B)
Application Design Decision 3
Single chain Easier chain management and permission management, harder data management and isolation Examples: V-A, V-B. **Multiple chains** Information isolation, harder chain management and permission management
Application Design Decision 4
External Validation oracle Introduce a third party trusted by the whole network Examples: Arbitrator (V-A) **Internal Validation oracle** Periodically injecting external state into the blockchain might introduce latency issues. The source of external state also needs to be trusted.
Application Design Decision 5
Permissionless vs. Permissioned blockchain Trade-offs: Performance, cost, censorship, reversibility, finality, flexibility in governance Permissions: Read/Join network, submit transaction, mine, create assets Example: Permissioned (V-A, V-B)

network. A private blockchain's write permission is kept to one organization. Using consortium and private blockchains requires a permission management component to authorize the participants within the network. There are many platforms that support building consortium chains and private chains, for example, Multichain and Eris.

Additionally, a blockchain-based system can maintain a unique chain to record all types of transactions together or maintain multiple chains to isolate information of separate parties or of separate concerns, for example, using one chain to store transactions, and using a separate chain to store access control information (Application Design Decision 3 in Table II).

Challenges of public blockchain Scalability is one of the main criticisms of public blockchains. Currently, the public blockchains, like Bitcoin and Ethereum, can only handle on average 3-20 transactions per second, while the mainstream payment service, like VISA, can handle on average 2000 transactions per second. There are works trying to improve the scalability (Blockchain Design Decision 1 in Table II). Bitcoin plans to increase its block size from 1MB to 8MB to allow miners to include more transactions into one block. Bitcoin lightening network [21] moves some of the transactions off-chain. A multisignature transactions is established between two participants as a micropayment channel to transfer value offchain. Once both sides wish to close the micropayment channel and finalize the value transfer, a transaction is submitted to the global Bitcoin blockchain. Segregated witness[4] proposes to remove the signatures from transactions to reduce the size of transactions, thus, one block could contain more transactions. Bitcoin-NG [8] decouples Bitcoin's blockchain operation into two planes: leader election and transaction serialization. Once a leader is selected randomly, it is entitled to serialize transactions until the next is selected.

Another concern of using blockchain is that all the information on the blockchain is publicly available to all the participants within the network, especially the information on the public blockchain, which is publicly accessible by everyone. Cryptography is the only way to preserve data privacy.

Besides, if a public blockchain is used, running computations on the blockchain costs actual money. Thus, applications are not supposed to deploy all computations and store all data on the blockchain. A common practice we also observed in our projects is to keep the big and private raw data off-chain, and stores the meta-data on-chain.

C. Communication Service

Communication service is a primary block of component interaction, which transfer data among components. Fig. 2 shows the internal structure of a node within the blockchain network. Components use blockchain as a mediator to transfer data. There are two ways to store data on the blockchain. One is to add data into transactions, like Bitcoin; the other is to add data into contract storage, like Ethereum. Both ways store data through submitting transactions to the blockchain, which may contain the information of money transfer together with some arbitrary data. After the transaction is included in the blockchain, the data becomes publicly accessible to the components within the network.

Some blockchain platforms provide an API and/or tools to access and filter the historical transactions. Ethereum suggests to cache all transactions to prevent the blockchain network from being under heavy stress due to frequent queries. The authors of MultiChain also plan to establish a bridge between its blockchains and regular relational databases in its future versions. Using the ordinary database indexing techniques, the historical transactions can be analyzed more efficiently.

Other than transactions, blocks also contain the state of the whole system after applying those transactions, In Bitcoin,

[4]https://github.com/bitcoin/bips/blob/master/bip-0141.mediawiki

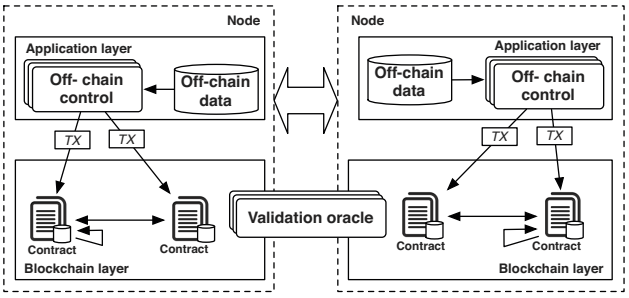

Figure 2: Interaction between applications and blockchain

the state is the collection of coins of all the accounts that have not been spent yet. In Ethereum, the state of the system is the changes of the whole contract storage. In Ethereum, every contract has its own storage where only the contract can write to. The contract storage can be viewed as a flexible key-value data store. The data stored in contract storage can be updated through sending transactions to the corresponding contract with new value. The contract has an address, which is used to query the contract storage. In the block, the state is stored in a tree data structure. For example, Bitcoin uses a Merkle tree whereas Ethereum uses a Patricia tree. Similarly as the transactions, the state of contract storage can be queried through API. By default, the query returns the current state.

D. Coordination Service

Different components of the architecture can coordinate their computations through the blockchain. To do so, it is possible to submit transactions to smart contracts to invoke the functions defined in the smart contracts, or use a validation oracle to sign transactions, the outcome of which depends on the external state.

As shown in Fig. 2, the control of the application flows through transactions initiated from externally owned accounts and transferred among contract accounts. Contracts behave like autonomous agents that live in the execution environment of the blockchain network. Contracts are instantiated by submitting transactions with the source code of the contracts to the blockchain network. A contract defines a set of functions. At the function level, the contract runs the code of a function when receiving a transaction calling the function with its required parameters. At the contract level, a contract could create a new contract by sending a transaction. The contract can also kill itself. The contract cannot receive any transactions after killing itself, but the source code of the contract cannot be removed from the blockchain. The source code is permanently stored within the transaction that creates the contract.

Validation oracle The execution environment of blockchain is a closed environment, which is not allowed to import external states through polling external servers. To address this limitation, the concept of *validation oracle* is introduced to evaluate conditions that cannot be expressed within blockchains.

A validation oracle is a mechanism that facilitate component coordination within the network using external state. Validation oracle is part of the blockchain connector, but can be independent of the blockchain network (Application Design Decision 4 in Table II). When a validation of transactions depends on some external state, the validation oracle is requested to validate the transaction and sign the valid transaction. This will block the progress of the transaction until a condition over the external state is verified by the validation oracle.

If the validation cannot be automated, a human arbitrator can validate transactions and sign valid transactions. If the validation can be automated, an automated arbitrator, could periodically pull the value of the variables from contract storage as state of the application to validate the transactions. However, both ways introduce a externally trusted party again. The last way is to inject the external state into the blockchain through periodically updating the value of variables within the contract storage. The last way can cause time delay between the external state changes and the change being injected into the blockchain.

In Bitcoin, an automated validation oracle can be implemented as a server outside the blockchain network, which has its own key pair. When a transaction requires external state to be validated, the validation oracle is requested to sign the transaction on-demand. The logic of validating transaction is defined by the user. Thus, the validation oracle signs a transaction when the user-defined expression on the server is evaluated to be true [2]. To reduce the required trust, Orisi[5] runs a set of independent validation oracles. Orisi allows the participants involved in a contract to select a set of oracles they both are comfortable using before initiating the contract, and then sign a contract requiring a certain number of the validation oracle signatures.

E. Facilitation Services

1) Transaction validation: The mechanism to validate transactions is specific to blockchains. Generally, the transactions are validated by being re-executed by the node that receives the transactions. For example, in Bitcoin, the validation of transactions relies on two scripts, including a *locking* script in the output of a transaction that specifies the conditions to spend the coins referred by the transaction, and an *unlocking* script that satisfies the conditions placed on a transaction output by the *locking* script. When a transaction is validated, the unlocking script in each input is executed alongside the corresponding locking script to see if it satisfies the spending condition. The transaction is valid if the result of executing scripts is "TRUE", which means the *unlocking* script has succeeded in resolving the spending condition. If the result is not "TRUE" after executing the combined script, the transaction is invalid.

2) Mining mechanism: Mining is a process, in which some nodes within the blockchain network aggregate transactions into blocks. These nodes are called *miners* in the blockchain network. Once a new block is generated by a miner, the miner propagates the block to the blockchain network. And the new

[5]Orisi: Distributed Bitcoin oracles — http://orisi.org/

block is included into the public blockchain after the whole network reaching consensus.

There are different mechanisms to select the miner as the next author to update the ledger (Blockchain Design Decision 2 in Table II). In Bitcoin, the miner is chosen at random through "Proof-of-work". "Proof-of-work" is a piece of data that is very costly to produce but easy to be verified. Producing a "Proof-of-work" is a random process with low probability. Thus, the miners in Bitcoin network compete to generate the "Proof-of-work" through burning their CPU time. The first miner to find the "Proof-of-work" is the potential next author of the blockchain. The difficulty of the work is adjusted to generate a new block every 10 minutes. However, proof of work largely limits the capacity of processing transactions.

"Proof-of-stake" is an alternative mechanism, which grants mining rights to participants in proportion to their holding of the currency within the blockchain network. For example, the miners in Peercoin blockchain network need to prove the ownership of a certain amount of currency to mine blocks. "Proof-of-stake" blockchains provide protection from a malicious attack because executing an attack would require the attackers to own large amount of currency, which is very expensive. Besides, the miners owning a large stake most probably won't attack the system, for example, through double spending. In long term, such attacks will decrease the value of the cryptocurrency and the value of their stake.

The "Proof-of-burn" process used in Counterparty blockchain involves destroying Bitcoins and generating proportionally XCPs (coins used in Counterparty). More recently, Permacoin proposes a modification to Bitcoin [17], which uses "Proof-of-retrievability" to re-purpose Bitcoin's mining resource to distributed storage of archival data. This approach provides additional incentives to contribute resources to the network.

3) Secure clearing payment: Blockchain provides a service of trusted peer-to-peer payment through cryptography. Every transaction is associated with the public key of its initiator. The transaction can be broadcast to the blockchain network only after the initiator signing the transaction with the corresponding private key. Thus, the authenticity is enforced by the key pairs. The transaction validation checks if new extra money created from the blockchain network after a specific transaction.

4) Permission management: Blockchains could be classified into permissioned blockchains and permissionless blockchains (Application Design Decision 5 in Table II). The service of permission management is provided by a permissioned blockchain network.

The participants of a permissionless blockchain networks is either pseudonymous or anonymous, like Bitcoin and Ethereum. Using anonymous validators takes the risk of Sybil attack, where the attacker gains a disproportional amount of influence on the system. For example, in Bitcoin, any participant with a sufficient share of computational power is able to change the records in the blockchain without respecting jurisdictional boundaries and therefore undermine financial sanctions and seizure of assets [7]. Besides, the "Proof-of-work" process, used as a Sybil protection mechanism, is costly and wasteful.

By contrast, permissioned blockchain networks, like Ripple and Eris Industries, are more congruent with traditional banking systems and can provide more utility to financial institutions [25]. In permissioned blockchain networks, the identity of the validators or even the participants is whitelisted through some types of KYC (Know Your Customer) procedure, which is a widely used method of managing identity in traditional finance. It means that the participants of the system require legal identities in real world to validate transactions. Thus, permissioned blockchains are able to legally host off-chain assets in the real world, while permissionless systems cannot.

Other than the permission of validation, basic permissions like joining the network, submitting transactions, mining, and creating assets can be also managed by the permission management service. Once joining a blockhain network, the participant inherently has the read permission on the blockchain because all the information recorded is publicly available. The permission information can be stored on-chain as well.

There are trade-offs between permissioned and permissionless systems including transaction processing rate, cost, censorship, reversibility, finality [25] and the flexibility in changing and optimizing the network rules.

5) Economic incentive: Every blockchain introduces economic incentives, reputation and rating mechanisms for miners to validate transactions and generate blocks and participants to be honest.

For example, in Bitcoin, the miners have two incentives to mine blocks, including the reward of generating new blocks and the transaction fees associated with transactions being aggregated into the blocks. Ethereum also charges computation fee for the miner to execute the smart contracts. Enigma has a fixed price for storage, data retrieval, and computation within the network. Besides, a node is required to submit a security deposit to join the network. If a node is found to lie, its deposit will be split among the other honest nodes.

IV. Comparison with Other Connectors

A. Centralized, Shared Data Store

Shared data stores, like key-value stores, export a basic Create/Read/Update/Delete (CRUD) interface. The blockchain is an append-only data store as it does not support update but rather supports the creation of new transactions. Any changes/updates on contract states are appended to the blockchain as new transactions. An analogy with this so-called ledger in data stores is the concept of log where data items get appended but never deleted or updated. This immutability-of-stored-information property is the key to the traceability of the relevant assets recorded on the blockchain.

Traditional shared data stores use different strategies to improve sustainability and throughput, and reduce latency, such as master-slave replication, and multi-master replication. Blockchain provides a more sustainable data storage because the data is duplicated on every node within the blockchain

network. But the throughput of some blockchains is not comparable with shared data store due to the latency caused by mining.

Traditional shared data stores have their own consensus protocols to synchronize replicas [11] in a fully trusted environment, such as 2-Phase Commit and Paxos. The consensus protocol of blockchain is aimed to tolerant Byzantine Generals' Problem [13], in which components of the system aim at reaching agreement among themselves to process correct operations despite a faulty component. The comparison of the consensus protocols used for blockchains and for general distributed systems is detailed in Section IV-B.

Besides, blockchain is able to validate the consistency of transactions based on rules attached with the transactions in terms of smart contract. Such rules can be applied on the whole blockchain, for example, to prevent double-spending problem through checking new extra money created during a spending transaction.

B. Replicated State Machine

Replicated state machine [22] is a general method to implement a fault-tolerant service with a distributed system. To cope with failures, it replicates the service at several servers and coordinates the service requests issued by the clients. Similarly, the blockchain uses distribution not to depend nor rely on any single entity.

State machine replication typically relies on a consensus protocol that takes as an input the requests of the components and decides upon one of these requests [12]. In the case of a distributed locking service, the consensus will guarantee that only one particular client acquires a lock, while multiple clients requested it concurrently. Blockchain also features a consensus protocol to ensure that among multiple conflicting proposed transactions, only one gets approved, preventing for example a double spending of the same coins.

For reaching a consensus on a particular transaction request, the replicated state machine requires sufficiently many votes. Replicated state machine rely on quorums of voters [15] that stem from the concept of weighted votes [9]. Typical blockchain implementations also requires sufficiently many votes. In Ripple, sufficiently many votes are obtained when a minimum of nodes in a unique node list have voted whereas in Bitcoin sufficiently many votes are obtained when a sufficiently complex challenge (Proof-of-work) is solved.

A replicated state machine supports *communication* by transmitting data among components. Components can store and retrieve information that will persist despite failures. The state machine replication guarantees that the information stored by one component gets replicated and delivered to another components upon requests even when some failures occur.

To address arbitrary failures or Byzantine failures [13], replicated state machines exploit security mechanisms. The sender of a message is typically authenticated with public-key cryptography so that the encryption with the sender private key serves as a signature to whoever decrypts the message with the corresponding public key. The digital signature resulting from public-key cryptography is also used in blockchains to preserve the ownership of coins. Collision-resilient hash functions help verifying the integrity of the message. They take the content of the message and produce a digest. This digest once sent encrypted allows the receiver to observe that the signed message was not altered. As an example, Bitcoin uses the SHA256 whereas the early replicated state machine tolerating Byzantine failures [5] used the AdHash solution based on MD5.

Finally, a replicated state machine totally orders the requests from components. It controls concurrency by scheduling requests issued by components and thus serves as a *facilitation* connector. This total order is also the key property of the blockchain. In Bitcoin, each block contains the hash of the previous block according to this total order, hence allowing to audit preceding transactions by backtracking the blockchain up to the genesis block. To maintain this total order and to prevent the chain from becoming a tree, miners always append blocks to the first chain of maximal length they hear of and all transactions that are part of forked blocks in shorter branches are simply discarded.

V. Project Retrospective

A. Data Monetization

Our first project is a platform to support the scenario of data monetization in which the data owners increase the value of their data through trading their data sets with data consumers. We consider two use cases. In one case, the data providers publish their data sets on the platform and the data consumers could select data sets from one or more data owners to do analytics for different purposes, and pay the data owners according to the value of their data sets. In the other case, the data consumers first post their analytics jobs with the price information on the platform, after which data owners could browse the list of analytics jobs and select the jobs based on the conditions defined in the offer.

The platform can be used in different business scenarios, for example, trading personal data produced by individuals. This scenario is inspired by [14], which discusses an economy of micropayment based on the Web to compensate people for originally creative work they post on the Web. Thus, personal data is treated as private property that can be traded.

Another possible scenario is the data analytics across organizations. The organizations doing data analytics pay the organizations who provide the data. The amount of money is calculated based on the value of the data set. For example, in an analytics based on two data sets from two different insurance companies, the data set from a company with larger number of customers is more valuable than the one from a company with smaller number of customers. Thus, the insurance company, which provides more valuable data set gets more money from the organization doing the analytics using the data sets.

Fig. 3 shows the architecture of the platform. The platform provides mainly three functions, including data trading,

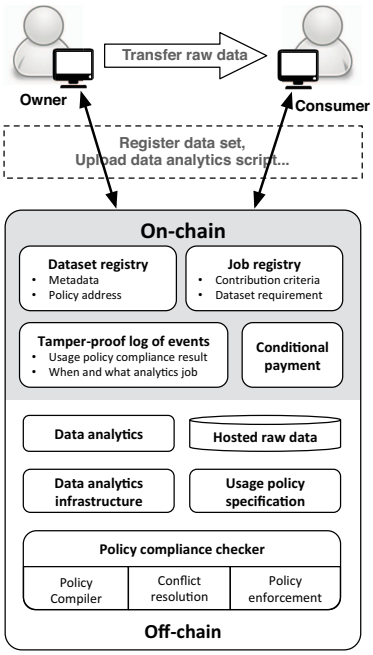

Figure 3: Architecture of data monetization platform

compliance checking of user-defined usage policy and data analytics. The policy compliance checking and data analytics is off-chain functionality, the technical detail of which is out of the scope of this paper.

The blockchain in this project allows the communication and facilitates the interactions between data owners and data consumers through running a set of smart contracts, logging events in an immutable data storage and providing a conditional payment infrastructure.

On the blockchain, there is a data set registry implemented as a smart contract, which stores all the data sets registered on the platform and allows data owners to register a new data set on the blockchain. The new data set is registered through calling the data set registry contract to create a data set contract, which stores the hash of the data set to allow consumers to check the integrity of the off-chain data. The metadata of the data set, like the description and the size of the data set, and a pointer to the corresponding user-defined usage policy is stored off-chain.

Similarly, we use another smart contract to implement a job registry that stores the list of the existing analytics jobs on the platform and allows data consumers to add new analytics jobs. Every analytics job is a contract, which defines the requirement of the requested data sets and the criteria to measure the contribution of an involved data set, for example, the size of the data set, the publish date of the data set, and the coverage of the data set etc. A more comprehensive value model is out of the scope of this project. The trading and negotiation logic are implemented in smart contracts as well.

Blockchain provides a tamper-proof log of events that ever occurred in the platform, including both on-chain and off-chain events/activities/data. On-chain events/activities include registering, trading and negotiating. Off-chain data include results of usage policy compliance checking and the information of analytics job, such as processing time, the data sets involved, and the monetary value eventually paid to each of the data owner. Besides, the blockchain is inherently a payment infrastructure that supports conditional payment. In our case, for example, the payment is triggerd before the analytic job starts. The amount of money is calculated by the smart contract according to the contribution criteria associated with the analytics job and metadata of the involved data set.

Users, as data owner or data consumer, interact with the smart contracts running on the blockchain. In this platform, due to its size, the raw data is stored and transferred off-chain. This reflects current practices of popular Web applications which allow users to download the data associated with their accounts, for example, Google takeout service[6].

One issue of this kind of marketplaces is how to verify that the data being sold complies with the owner's description. In our platform, we introduce reputation and rating mechanisms for data owners to be honest. A similar ongoing industrial project, called Slur[7], is an anonymous marketplace for trading secret information. Slur introduces an additional role, called *Arbitrators*, that validates the data. When the buyers claim that the content does not match the seller's description, slur randomly selects several arbitrators to evaluate the content. The arbitrators are paid for their effort.

B. Organizations Sharing Sensitive Data

Another project we are working on is a platform for participating organizations to securely negotiate and store sensitive data values (such as prices, delivery dates, or legal contracts). The architecture of this project is given in Fig. 4. This scenario requires secure data exchange and negotiations. Some details had to be omitted and generalised due to Intellectual Property reasons.

The users could log on to the platform via federated access. There are negotiation templates stored in a central place with specific value fields in the template to be negotiated. The sensitive information is still kept inside the organizations where the data was originated, and thus not available to other organizations using the same platform or stored in any centralised third party platform. The negotiation can be initiated, negotiated and signed through a web application or a mobile application.

As a part of the platform, blockchain is used to facilitate the negotiation by using smart contracts, and store the different versions of sensitive data produced during negotiation. One smart contract is used to represent one negotiation. The negotiation is initiated by a participant from an existing template by selecting initial values for the negotiable variables. All the values produced during the negotiation are included in blockchain. Since the information on blockchain is publicly available to all users, the value is encrypted before being

[6]https://www.google.com/settings/takeout
[7]http://slur.io

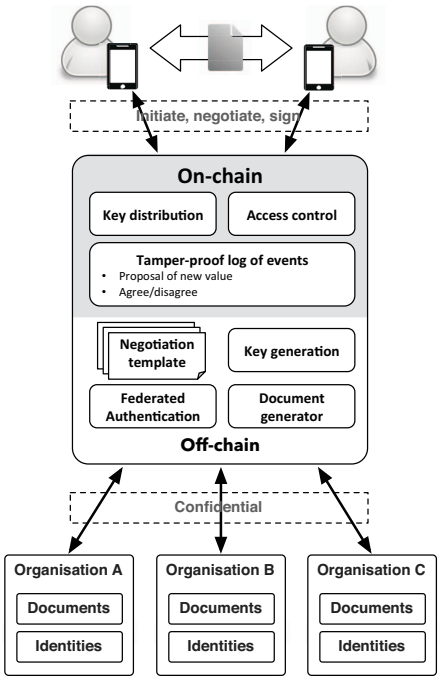

Figure 4: Overview of the legal platform

included into the blockchain. When a negotiation is created by an involved participant, our platform generates a secret key associated with the negotiation, which is used to encrypt the value of the negotiable variables before adding the information into blockchain. Then a smart contract is generated to facilitate this negotiation. The smart contract 1) implements the negotiation process, 2) has access-control management to restrict the access to the negotiation, and 3) distributes the secret key of the contract by encrypting it with the participant's public key, which is his/her blockchain address, and allows the participant to retrieve his/her encrypted contract secret key. Once a participant gets the encrypted contract secret key, he/she decrypts it with his/her private key. With the contract secret key, the participant can query the encrypted value of the negotiable variable and decrypt it. The retrieval of the negotiable variable and decryption is transparent to the end user.

The negotiation is done peer-to-peer and may require manual user intervention. Every activity, such as proposing a new value, agreeing or disagreeing on a value, are included in the blockchain as different versions of the negotiation. Once an agreement is reached and signed by all the involved participants, the negotiation is finalized, a digital document of the negotiation is generated. The digital document is stored internally in the organisation. To bind the digital document and the corresponding smart contract, the address if the smart contract is included in the digital documents, and then the hash of the digital document is included in the smart contract. After the binding, the smart contract could be killed to avoid further interaction and modification.

In this platform, the blockchain prevents tampering and enforces integrity and auditabilty of the sensitive data. A consortium blockchain or public blockchain can be selected in this scenario since the privacy of the data is preserved through cryptography.

VI. Discussion

Lesson: scalability and performance The performance of public blockchain is very limited. As mentioned earlier, public blockchains can only process 3-20 transactions per second. The average transaction processing rate we calculated from the whole blockchain (1020156 blockchains at 18/02/2016 00:21:12 GMT) is 1.7 transactions per block, and the average transaction processing rate from the latest 100000 blocks (920156-1020156) is 3.4 transactions per block. The average mining time is 17 seconds.

We conducted a small experiment to test the performance of a private blockchain, and compared the result with the public blockchain. We built a Ethereum private blockchain and created 50 accounts in the genesis block. We issued simple transactions which transfer 0.001 ether from one account to another. The sender and the recipient of the transactions were chosen randomly from the 50 accounts. During the experiment, we found a bug in Ethereum that causes many transactions failed to be included. The issue was reported and confirmed as legitimate[8]. After fixing the issue, the performance of the private chain became much better than the public chain. The number of transactions included into one block was around 15000 transaction on average, and the mining time was around 41 second on average. Thus, the transaction process rate was around 366 transactions per second.

Lesson: Privacy Public blockchains do not guarantee data privacy. Also permissionless blockchains cannot preserve privacy of the data because anyone could join the blockchain network without permission, and all the data on the blockchain is visible to all participants. Thus, for scenarios like the legal contract platform, a permissioned blockchain is more appropriate, which can allow developers to explicitly grant permissions to the participants. Besides, the information on blockchain might need to be encrypted to preserve privacy. In this case, the key needed to be generated and stored off-chain. Thus, the blockchain doesn't have enough information that can be used by the components without permissions to access the sensitive data.

Lesson: Trusted third-party Using external state does not always introduce the need for trusting an additional party. For example, in the licence renewal scenario, the government is a trusted party anyway, thus, we use government as an validation oracle that injects external state into the blockchain.

Lesson: Incentives If a blockchain-based system has computation ran off-chain or data stored off-chain, an additional economic incentive is required for the participants to be honest. Incentives for miners may include rewards, transaction

[8]https://github.com/ethereum/go-ethereum/issues/2139

fees, computation fees, or data storage fees. Incentives for participants to be honest can involve: security deposits, or reputation and rating mechanisms used in our first project.

Lesson: Reducing cost The applications on top of the blockchain could reduce the transactions being included into blockchain. For example, establishing micropayment channel, which only submit the transaction once being closed by either party. The transient state does not need to be included into blockchain, for example, not all the activities during negotiation are worth to be included into blockchain. To reduce the number of submitted transactions, an alternative design is to only record the different value of negotiable variables and the final voting result of the value rather than record every single voting activity.

Lesson: Data and contract management If the data to be stored by the application is associated with the state of the contract processing it, the data will be discarded once the functionality of the contract is updated through uploading a new version of the contract to the blockchain. To address this problem, we suggest to separate the computation from the data in dedicated smart contracts.

Once deployed on the blockchain, the smart contract is always "running" and responding to requests. We suggest to kill the contract explicitly once the functionality of the contract is not used to avoid further interaction and unnecessary cost.

Lesson: Off-chain data Store We stored meta-data on-chain to be publicly accessible, and kept the raw private data off-chain. For example, we put the hash of personal data on-chain, but transfer the raw data off-chain.

Due to the limited size of the data store provided by the blockchain [23], an off-chain data store is necessary for some applications. There are existing platforms providing a data layer on top of the blockchains, such as Factom, which stores only the hash of the the private data and small amounts of public data in their own blockchain. Factom also anchors the Bitcoin blockchain every 10 minutes to be more secure. Distributed data storage, like IPFS[9], DHT (Distributed Hash Table) are also sometime used in combination with the blockchains to build decentralized applications.

VII. Conclusions

In this paper we have presented our experience from using the blockchain in several projects. The blockchain provides communication and coordination services through transactions, validation oracles and smart contracts, and specific facilitation services, including permission management, cryptography-based secure payment, transaction validation, mining and incentives. We have compared the blockchain to related software connectors such as the shared data store and the replicated state machine, highlighting the most important theoretical differences. Based on the practical project experience we have distilled important design decisions implied by the choice of introducing a blockchain in the architecture and discussed the corresponding trade-offs.

[9]IPFS — https://ipfs.io/

Acknowledgments

NICTA is funded by the Australian Government through the Department of Communications and the Australian Research Council through the ICT Centre of Excellence Program.

References

[1] Crypto-currency market capitalizations. http://coinmarketcap.com/.
[2] bitcoinwiki. Contract. https://en.bitcoin.it/wiki/Contract#Example_7:_Rapidly-adjusted_.28micro.29payments_to_a_pre-determined_party.
[3] J. Bonneau, A. Miller, J. Clark, A. Narayanan, J. A. Kroll, and E. W. Felten. Sok: Research perspectives and challenges for bitcoin and cryptocurrencies. In *the 36th IEEE Symposium on Security and Privacy (SP2015)*, pages 104–121, May 2015.
[4] V. Buterin. On public and private blockchains. https://blog.ethereum.org/2015/08/07/on-public-and-private-blockchains/.
[5] M. Castro and B. Liskov. Practical byzantine fault tolerance. In *Proc. of OSDI*, pages 173–186, 1999.
[6] P. Clements, F. Bachman, L. Bass, D. Garlan, J. Ivers, R. Little, R. Nord, and J. Stafford. *Documenting Software Architectures: Views and Beyond*. Addison-Wesley, 2003.
[7] EBA. Eba(european banking authority) opinion on "virtual currencies". 2014.
[8] I. Eyal, A. E. Gencer, E. G. Sirer, and R. van Renesse. Bitcoin-ng: A scalable blockchain protocol. In *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*, Santa Clara, CA, Mar. 2016. USENIX Association.
[9] D. K. Gifford. Weighted voting for replicated data. In *Proceedings of the seventh ACM symposium on Operating systems principles*, pages 150–162. ACM Press, 1979.
[10] IBM. Device democracy saving the future of the internet of things. 2015.
[11] B. Kemme and G. Alonso. Database replication: a tale of research across communities. *Proceedings of the VLDB Endowment*, 3(1-2):5–12, 2010.
[12] L. Lamport. The part-time parliament. *ACM TOCS*, 16(2):133–169, 1998.
[13] L. Lamport, R. Shostak, and M. Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, July 1982.
[14] J. Lanier. *Who Owns the Future?* Simon and Schuster, 2013.
[15] D. Malkhi and M. Reiter. Byzantine quorum systems. In *Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing*, STOC '97, pages 569–578, 1997.
[16] N. R. Mehta, N. Medvidovic, and S. Phadke. Towards a taxonomy of software connectors. In *Proc. of ICSE*, pages 178–187, June 2000.
[17] A. Miller, A. Juels, E. Shi, B. Parno, and J. Katz. Permacoin: Repurposing bitcoin work for data preservation. In *IEEE Symposium on Security and Privacy*, May 2014.
[18] M. Morisse. Cryptocurrencies and bitcoin: Charting the research landscape, August 2015.
[19] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. https://bitcoin.org/bitcoin.pdf.
[20] S. Omohundro. Cryptocurrencies, smart contracts, and artificial intelligence. *AI Matters*, 1(2):19–21, Dec. 2014.
[21] J. Poon and T. Dryja. The bitcoin lightning network: Scalable off-chain instant payments. 2016.
[22] F. B. Schneider. Implementing fault-tolerant services using the state machine approach: A tutorial.
[23] P. Snow, B. Deery, J. Lu, D. Johnston, and P. Kirby. Business processes secured by immutable audit trails on the blockchain. 2014.
[24] M. Swan. *Blockchain: Blueprint for a New Economy*. O'Reilly, US, 2015.
[25] T. Swanson. Consensus-as-a-service: a brief report on the emergence of permissioned, distributed ledger systems. 2015.
[26] R. N. Taylor, N. Medvidovic, and E. M. Dashofy. *Software Architecture: Foundations, Theory, and Practice*. Wiley, 2009.
[27] F. Tschorsch and B. Scheuermann. Bitcoin and beyond: A technical survey on decentralized digital currencies. *IACR Cryptology ePrint Archive*, 2015:464, 2015.