On Design Issues and Architectural Styles for Blockchain-driven IoT Services

Chun-Feng Liao, Sheng-Wen Bao, Ching-Ju Cheng, and Kung Chen
Department of Computer Science
National Chengchi University
Taipei, Taiwan

***Abstract*—We can perceive the advent of smart living spaces attributed to the fast emerging of IoT (Internet of Things) technologies. By combining with the blockchain technology, many innovative business models can be brought into reality. This paper aims to report our recent progress in investigating the architectural issues for realizing blockchain-driven IoT services. In particular, we present and discuss four typical architectural styles for such services. A preliminary evaluation against different styles is also provided to compare the given styles.**

***Keywords*—Blockchain, IoT, Smart Contract, Ethereum**

I. Introduction

Recent years have seen increased attention being given to the study of blockchain[1]. Conceptually, blockchain is a kind of secured distributed database where the validity of data is verified by peers. The consensus of peers is achieved based on pre-determined policies (e.g. Proof-of-work or Proof-of-stake). Therefore, blockchain enables distributed peers to reach consensus in a trustless network without a centralized authority. Blockchain technologies facilitate many innovative business models that are not possible previously such as ride-sharing [2] and agri-food supply chain tracing [3]. Due to the highly distributed nature of IoT (Internet of Things) services, blockchain also attracts attentions of researchers in IoT domain. Blockchain serves as the billing layer so that it forms a marketplace of services among smart things [4]. A noteworthy example is Slock.it, which is a property sharing service based on its smart lock and the Ethereum[5] blockchain platform. Nevertheless, blockchain and IoT are both emerging technologies, the design issues and architectural styles of blockchain-driven IoT (B-IoT) services are still not well-explored. The purpose of this paper is therefore to report our recent progress in this direction. In this paper, we present design issues of B-IoT services. Also, four typical architectural styles for B-IoT are presented and discussed.

II. Design Issue and Architectural Styles

Let us begin by taking a look at the design considerations that are unique to B-IoT services:

1) The locations of blockchain endpoints: From a network's point of view, the blockchain is essentially a peer-to-peer network. A different implementation of blockchain has a different name for a peer in the blockchain network. In this paper, a peer in the blockchain is called a blockchain endpoint (also known as a node in many works of literature). In a blockchain network, endpoints keep track of blocks and are responsible for verifying transactions. In a proof-of-work blockchain, endpoints also generate new blocks through a competitive process called mining. Obviously, the place to deploy blockchain endpoints has a significant impact on the bandwidth, computation and space requirements and thus is an important architectural consideration.

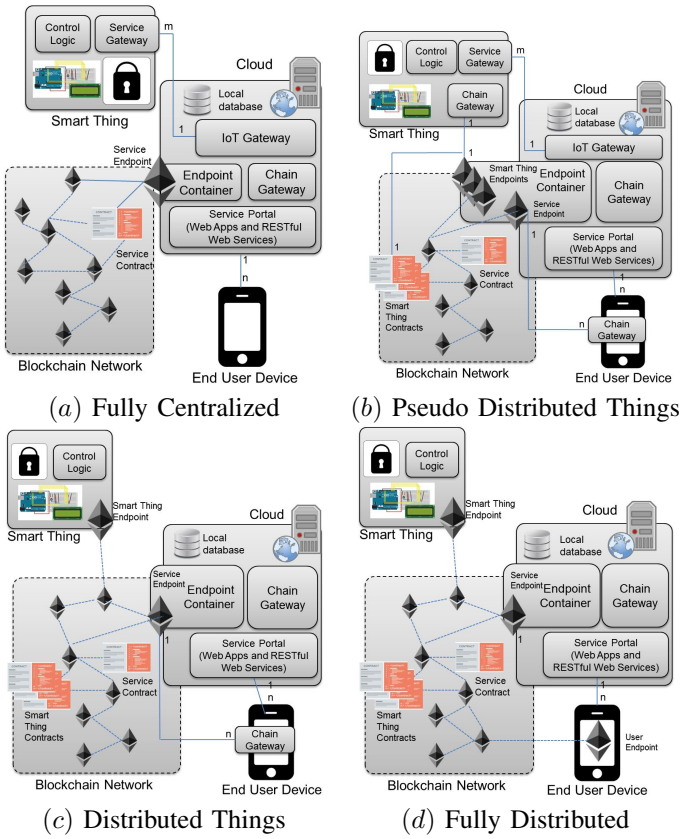
2) The distribution of business logic and data: In a B-IoT service, a useful alternative to implementing the business logic is the smart contract which refers to stored scripts in a blockchain. For example, the states of a contract are also verified and stored on the blockchain, so that many IoT services keep track of states of physical assets in the smart contract. This new alternative raises a new design consideration for developers, that is, which parts of logic and data are suitable for placing in the blockchain and which parts are not. As reaching consensus in blockchain usually takes more than few seconds, placing too many logic and data in the blockchain can lead to poor performance. On the contrary, the benefits of using blockchain are dissolved if we do not put logic and data onto the chain.

3) The mechanisms of cyber-physical integration: As mentioned, cyber-physical integration is an important issue in an IoT service. For example, before a user can use a rented smart thing, what is the objective of the payment? If each smart thing is correlated to a smart contract, then the payment can be realized by sending a transaction to the address of the contract. Another issue is how the service provider finds and controls things. For example, after the payment is confirmed, the renting service unlocks the rented device either by Websocket (off-chain) or by issuing a contract event (on-chain).

By taking the design considerations mentioned above into account, we can come up with four typical architectural styles for B-IoT services. To explain these architectures in a precise manner, we use a smart thing renting service as an illustrative scenario. In such scenario, users can search for the smart thing the rent and pay for it using smartphones. After the payment is confirmed, the smart thing is automatically unlocked. The thing is locked again after the lease time being expired.

One naive approach is shown in Fig.1a, where a user selects things to rent and pays on the Service Portal. Most program states including states of smart things are stored in the local database, and business logic is implemented as web applications and services. There are one blockchain endpoint and one smart contract which is used to record payment flow. The smart things do not aware of blockchain and are controlled by the vendor-specific approach. Such approach is called *Fully Centralized* as it keeps most of the logics and

978-1-5090-4017-9/17/$31.00 ©2017 IEEE



(a) Fully Centralized (b) Pseudo Distributed Things

(c) Distributed Things (d) Fully Distributed

Fig. 1: Architectural styles of Blockchain-driven IoT services

data in a centralize way. As a result, such design does not benefit from the advantages of the blockchain technology: robust and distributed consensus. To embrace the philosophy of blockchain more, the designer can create a contract for and to associate an endpoint with each smart thing. However, as mentioned, it is hard to deploy an endpoint on a smart thing because an endpoint can drain out resources of the device. One intermediate approach is the *Pseudo Distributed Things* architecture. As shown in Fig.1b, the endpoints for each node are deployed and managed in the cloud by the Endpoint Container component. Smart Things and the end user device can interact with blockchain via the Chain Gateway component. The implementation of Chain Gateway is usually platform dependent. For instance, in an Ethereum blockchain network, the Chain Gateway is realized using web3.js. This architecture is called Pseudo Distributed Things as the endpoints are physically located in the cloud and are logically attached to each smart thing.

Recently, Ethereum introduces a new concept called the light client which can perform the tasks of an endpoint in a low-capacity environment. The core idea is that each light client is assisted by some light servers, provided by endpoints on a more powerful machine so that the light client can perform partial tasks and keep partial data at a time. Such improvement allows us to place endpoints on the smart things, so that the *Distributed Things* architecture can be realized (see Fig.1c). Also note that in the Distributed Things architecture, IoT Gateway is unnecessary as the smart thing is now

TABLE I: A summary of architectural styles for blockchain-driven IoT services

Style name	Blockchain endpoint on	Thing identification	Thing control	Contracts	Minimal off-chain links
Fully Centralized	Cloud	Vendor specific	Websocket	1	$m+n$
Pseudo Distributed Things	Cloud	Contract address	Chain gateway	$1+m$	$m+n$
Distributed Things	Cloud and thing	Contract address	Chain event	$1+m$	n
Fully Distributed	Cloud, thing and device	Contract address	Chain event	$1+m$	0

n: number of users; m: number of smart things;

controlled directly by the corresponding smart contract. The architecture can be Fully Distributed (Fig.1d) if an endpoint is also deployed in the end user device. In the *Fully Distributed* architecture, the service provider does not implement payment logic. Instead, a user pays directly to the address of the smart contract associated with the thing one is going to rent.

Table I is a summary of several attributes of the architectural styles discussed above. It is worthy to note that if the style is more distributed, the less off-chain links it need. This observation indicates that ideally, one should design the system so that the architectural style is as close to Fully Distributed as possible. Because when there are more off-chain links, the system is less robust and less secure.

III. Conclusion

This paper discusses design issues and architectural styles of constructing B-IoT services. The presented architectural styles are useful for helping developers to make appropriate design decisions. Theoretically, a developer should design B-IoT services so that they are as close to the Fully Distributed architecture as possible. Nevertheless, one may find that other styles are more appropriate because of the efficiency or the cost (computation capability of smart things). Our next step is to investigate how these styles affect architectural attributes such as robustness, security, and efficiency.

Acknowledgment

This work is partially sponsored by Ministry of Science and Technology, Taiwan, under grant 105-2221-E-004-004.

References

[1] M. Swan, *Blockchain: Blueprint for a new economy*. "O'Reilly", 2015.

[2] Y. Yuan and F.-Y. Wang, "Towards blockchain-based intelligent transportation systems," in *Intelligent Transportation Systems (ITSC), 2016 IEEE 19th International Conference on*. IEEE, 2016, pp. 2663–2668.

[3] F. Tian, "An agri-food supply chain traceability system for china based on rfid & blockchain technology," in *Service Systems and Service Management, 13th International Conference on*. IEEE, 2016, pp. 1–6.

[4] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the internet of things," *IEEE Access*, vol. 4, 2016.

[5] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, vol. 151, 2014.