Towards Detecting Semantic Events on Blockchains

Abhisha Bhattacharyya[1], Rahul Iyer[1], and Kemafor Anyanwu[1]

North Carolina State University, Raleigh NC 27695, USA
{abhatt22, rahul_iyer, kogan}@ncsu.edu

Abstract. Blockchains maintain information about transactions creating and transferring digital assets. "Smart contracts" which are arbitrary, immutable code executable on a blockchain, can be used to implement complex business rules that govern transactional behavior as well as raise events which external applications can be notified about. However, the arbitrary and imperative nature of smart contracts and the events they raise, make their discoverability and reuse challenging. Further, events represented merely as function signatures limit the ability for more complex, semantic event detection which require richer representations such as semantic rules.
In this paper, we motivate and describe our efforts towards support for *semantic event detection on blockchains* that are founded on (i.) *a declarative, semantic data, transaction and event model* and (ii.) an *integrated blockchain-streaming application execution architecture*. We overview our selected implementation strategy that builds on the BigChainDB blockchain engine and Apache Kafka, a large-scale streaming platform.

Keywords: Semantic Events · Blockchains · Declarative transactions.

1 Introduction

Blockchains typically maintain information about transactions that create or transfer ownership of digital assets. To enable business applications with complex business logic for governing such transactions, the concept of "smart contracts" was introduced by blockchain platforms like Ethereum [3] and HyperLedger [1]. Smart contracts, which are arbitrary, immutable code executed on blockchains are also able to raise events which external applications can consume. Until very recently, limited interaction models were permitted with respect to events e.g. smart contracts cannot detect events raised by other smart contracts since they cannot read directly from the blockchain. Fortunately, a recent Ethereum extension Eventeum[2] now provides a connection between smart contracts and middleware layer via a message bus containing details of the event emitted. However, there are still major limitations with the state-of-the art in smart contracts and blockchain event models as illustrated using the following example.

Imagine a digital service marketplace that allows consumers to request for bids for specific services and service providers to offer service bids. This scenario can be useful in domains such as Manufacturing that are increasingly becoming more Cloud-oriented. In large-scale or bespoke manufacturing for example, it is common to need to outsource the manufacturing of some components. However, in high-capital domains, a significant degree of confidence and trust in capabilities and capacities of service providers is required by the service requestor. Some ideas on using the blockchain for maintaining historical records of bids, their fulfillment, performance, etc to be leveraged for mediating automatic agreements, has been proposed in [5]. These ideas have been explored using Ethereum smart contracts resulting in the following lessons learned:

Lack of Reusability: Each *request-for-bid - rfb* (and similarly bid) is a non-traditional blockchain transaction which would need to be implemented as a separate smart contract, even when rfbs have similarities or overlapping behavior. This is because program reuse is more challenging with imperative representations given that semantics is implicit in program behavior. The immutability of smart contracts also implies that if business conditions evolve even slightly, modifying the smart contract to capture new business logic is a major challenge.

Discoverability: In our example, a service provider would need to be aware of the existence of a smart contract requesting for a bid on services that they provide. This requires either (i.) some standardized token contract interfaces for representing specific protocols (analogous to ERC20/721) or (ii.) some registry of smart contracts indexed by standardized metadata. However, it is unlikely that a fixed set of function interfaces that comprise a token will be general enough to capture such protocols, given the variability in possible request and bid requirements (unless perhaps organized into very narrow vertical segments). Another approach that can be used to discover requests is via the events raised by smart contracts. Such events may be routed through a publish/subscribe platform like Apache Kafka (such as is done by Eventeum). However, subscribing to events requires apriori knowledge of event definition. Unfortunately, each smart contract raises events using a function signature of its choosing. Consequently, the same event could be represented using different signatures which cannot all be known apriori. Secondly, how to map such events into an appropriate publish-subscribe event detection model is unclear. This is because the semantics of an event may be implicit and complex. For example, a request for bid transaction that has parameters `Material` as *"PolyCarbonate"* and `Quantity` as *"10"* can be mapped to `"3D Printing"` event. Such terms and relationships would need to be captured in an ontology and semantic rule engine.

Our preliminary investigation leads us to the conclusion that there are two key requirements needed to enable detection of semantic events on blockchains: (i.) The use of a declarative and semantic data, transaction and event model that allows new transaction (e.g. bids) and event classes to be introduced in a straightforward manner. The semantic-awareness requirement allows reasoning about semantic relationships between transaction and event types. (ii.) A processing architecture that integrates blockchains with a semantic rule engine

and publish and subscribe mechanisms, enabling detection of complex semantic events. We are currently developing these ideas in the context of a project called SmartChainDB. SmartChainDB extends a blockchain platform BigChainDB [4] that has a semi-structured data model, enable significant flexibility in extensibility.

References

1. An introduction to hyperledger. Tech. rep., The Linux Foundation, https://www.hyperledger.org/wp-content/uploads/2018/07/HL_Whitepaper_IntroductiontoHyperledger.pdf
2. Listening to ethereum events with eventeum. https://kauri.io/article/90dc8d911f1c43008c7d0dfa20bde298/listening-to-ethereum-events-with-eventeum
3. A next-generation smart contract and decentralized application platform. Tech. rep., Ethereum, https://github.com/ethereum/wiki/wiki/White-Paperethereum
4. Bigchaindb 2.0 the blockchain database. Tech. rep., BigchainDB GmbH, Berlin, Germany (May 2018), https://www.bigchaindb.com/whitepaper/bigchaindb-whitepaper.pdf
5. Angrish, A., Craver, B., Hasan, M., Starly, B.: A case study for blockchain in manufacturing:fabrec: A prototype for peer-to-peer network of manufacturing nodes. Procedia Manufacturing **26**, 1180–1192 (2018)