LEDGER
ledgerjournal.org

ISSN 2379-5980 (online)
DOI 10.5195/LEDGER.2019.173

PROCEEDINGS ARTICLE

A Decentralized Context Broker Using Byzantine Fault Tolerant Consensus

Aswin Karthik Ramachandran Venkatapathy,*† Michael ten Hompel‡

Abstract. A context broker is a reliable message-relaying service used to connect devices by integrating all device protocols and communication methods, and reliably transporting messages while isolating data from other application service layers and networking complexities. A highly scalable decentralized context broker stack is composed of three layers—starting with a peer-to-peer network connecting a byzantine fault-tolerant (*i.e.*, blockchain-based) consensus protocol—and it manages the communication using a web-socket streaming protocol as interface to other applications. This paper presents such a concept for a decentralized context broker stack for intercommunication between heterogeneous materials handling systems, and deploys the stack as proof-of-concept using ROS-based robots in a logistics scenario.

1. Introduction

Industry 4.0 and the concept of a socially networked industry requires that the entities in an industry are networked and are able to communicate between each other to autonomously collaborate for performing tasks.[1] dezCom is an architecture for **de**centrali**z**ed **com**munication and messaging. For a truly decentralized industrial process, the communication between the entities that take part in the process should also take place in a decentralized manner. Even though the execution algorithm is decentralized in nature, when running on a centralized communication architecture, such as Message Queuing Telemetry Transport (MQTT) or Orion context broker, it is not truly decentralized. Therefore, the effort to abstract the communication layer with a pure decentralized communication framework is implemented and presented in this article as the dezCom communication stack. Decentralization using a blockchain not only increases the availability of the broker but also increases the security of data in a trustless network. There are many implementations of decentralized robots based on blockchain technology.[2,3] In this paper we contribute by developing the first-ever decentralized industrial supply chain messaging system based on a mining-less blockchain where emphasis is given to messaging using assets in a socially networked industry. We begin with an overview of context brokers in section 2. The dezCom communication stack, described in section 3, is demonstrated in the proof-of-concept deployment in section 4. Finally, in section 5, the results for dezCom with use cases is summarized.

* 166Lf7mVS45uy8SN3THcnMciyFnuYzayUd

† A. K. Ramachandran Venkatapathy (aswinkarthik.ramachandran@tu-dortmund.de) is Research Associate of Chair for Material Handling and Warehousing at TU Dortmund, Germany.

‡ M. ten Hompel (michael.tenhompel@tu-dortmund.de) is Professor of Chair for Material Handling and Warehousing at TU Dortmund.

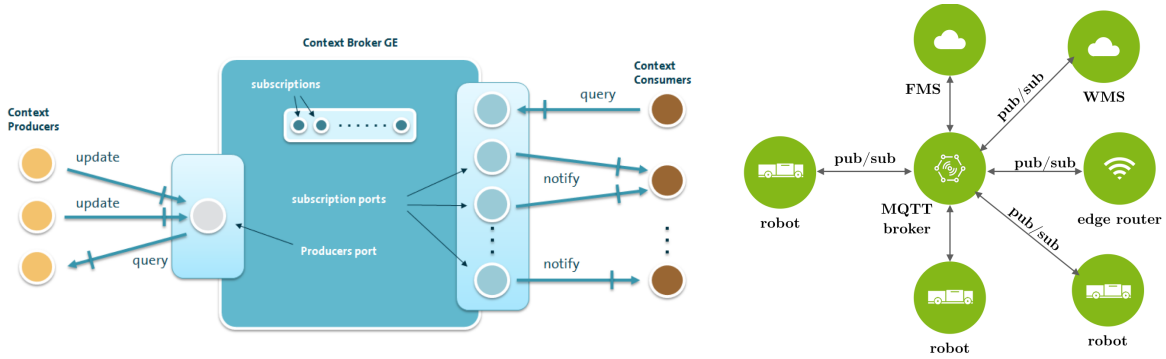

Fig. 1. Context brokers communication flow Fiware (left),[4] MQTT stack as deployed (right)[5]

2. Context Broker Overview

The two types of context brokers currently used in industry are the Orion Context broker from the Fiware foundation and the MQTT broker. The Orion Context broker allows the user to manage the entire life-cycle of context information including updates, queries, registrations, and subscriptions which are also used in dezCom with transactions on assets approach.[4] The Fiware Catalogue contains a rich library of components (Generic Enablers) with reference implementations that allow developers to put functionality into effect such as a connection to IoT or Big Data analysis, making programming much easier,.[4] Whereas in dezCom, we try to keep the communication interface generic with web sockets allowing other applications to interface freely. Even though Fiware is the only available multipurpose context broker, the architecture is centralized (as shown in figure 1) with a single point of failure, with every interfaced entity having a web-server listening on a port for REST endpoint calls for every communicated message. MQTT is an ISO standard (ISO/IEC PRF 20922) publish-subscribe-based messaging protocol designed for connections with remote locations where a ”small code footprint” is required or the network bandwidth is limited.[5] MQTT is primarily a messaging system where context information can be embedded in the messages. It offers a generic communication interface and transports the messages to subscribers with the help of a central MQTT broker.

2.1. Requirements—The requirements for a decentralized context broker must include (but not be limited to) the following:

Throughput message fan-out—a small number of data producers (publishers) need to frequently send data to a much larger group of consumers (subscribers).

Addressing and discovery—sending data to specific application instances, devices, or users.

Load balancing with N-way scalability—where applications produce a large volume of work items or requests and dynamically use a scalable pool of workers.

Location transparency—applications need to scale to a very high number of instances spread out geographically, and with intrinsic modularity in the applications, specific endpoint-configuration information for applications to understand the endpoints.

Fault tolerance and trust architecture—the application needs to be highly resilient to network or other outages with the application executed by diverse entities of the process supply chain to allow anyone to take part in the communication.

The above mentioned widely-used systems do not meet the requirements except for high throughput message fan-out *i.e.*, if the server with the central instance is unreachable, the whole

ISSN 2379-5980 (online)
DOI 10.5195/LEDGER.2019.173

network communication is disrupted.

3. dezCom: A Decentralized Context Broker

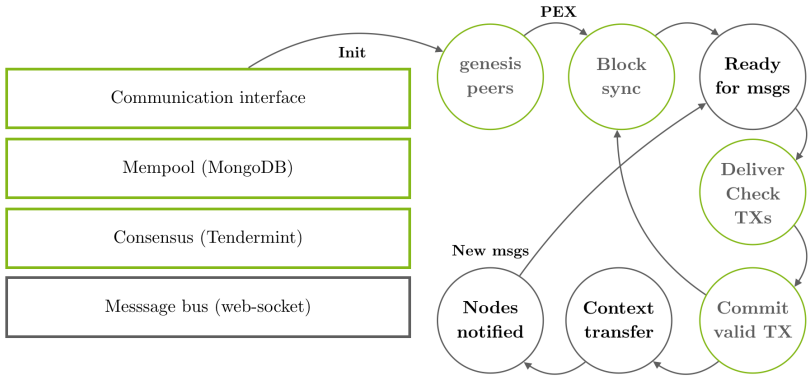

Fig. 2. State machine of the DezCom stack with messaging steps

The decentralized context broker implemented in this paper has three main layers as shown on the left of figure 2: *(i)* communication interface for forming peer-to-peer network, *(ii)* a consensus protocol and *(iii)* a message bus, which is a streaming protocol for interfacing the application. Along with the layers of the stack, the state machine of the stack's blockchain from the genesis block is illustrated in figure 2 with the *decentralized consensus protocol* Tendermint. The stack has a *communication interface*, which is initially implemented with a TCP/IP stack with IP networking. This communication interface can be changed, but the basic services such as reachable, addressable nodes with functions for discovery need to be provided. Any low power wireless sensor network with a 6LoWPAN networking stack can also be used to run this context broker if required computational resources are available. A memory pool or *mempool* is used to store all the incoming transactions (UTXO) arriving through the communication interface. In this reference implementation, Tendermint is used for the byzantine fault tolerant nature of consensus and the mining-less blockchain protocol.[6] There is a socket interface where the events of the consensus protocol are triggered as the *message bus* which interfaces with the application to perform CREATE and UPDATE transactions as well as to query time series data. Since the consensus is ledger based, there is no DELETE operation; instead, an update operation should be made on the data to deem it invalid. In a context broker case, the process state needs to be changed to completed.

In figure 2, the consensus protocol of Tendermint is shown in simplified steps as used in the dezCom framework. Nodes connected to a particular node are called *peer nodes* in the Peer-2-Peer (P2P) network, where P2P network discovery happens using the Peer Exchange protocol (PEX) as used in bit-torrent networks. In the case of industrial mobile robots, all the field robots are connected to the Fleet Management System (FMS) as peers, whereas the Warehouse Management System (WMS) connects to low-level hardware such as the PhyNodes using an edge router as a client which is Raspberry PI-based.[7] Any system meeting the necessary hardware requirements can host a node for consensus or connect to a node using the API interface. The decentralized consensus happens in multiple rounds where the process of deciding the next block (at some height H) is composed of one or many rounds.[6] NewHeight, Propose, Prevote,

ISSN 2379-5980 (online)
DOI 10.5195/LEDGER.2019.173

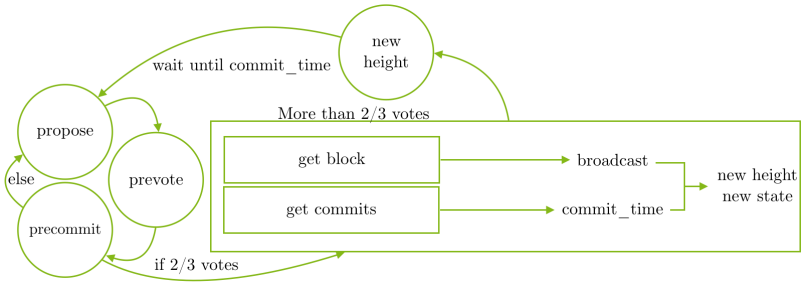

Fig. 3. Consensus protocol rounds for committing new blocks into the blockchain

Precommit, and Commit are the states / steps (S) in a state machine of every round (R).[6] At each height of the blockchain, a round-based protocol is run to determine the next block, which is depicted in figure 3.

Each round is composed of three steps—Propose, Prevote, and Precommit—along with two special steps: Commit and NewHeight. Valid transactions become available in the message bus after the Commit step is completed. In the optimal scenario, the order of steps is NewHeight to (Propose —>Prevote —>Precommit) to Commit to a new height and so it goes on.[6]

Logistics applications are interfaced using the message bus as an ***application interface*** where the Application Blockchain Interface (ABCI) of Tendermint is implemented. Events and REST API are the two ways to communicate with the Tendermint protocol. There are a set of 15 events available from the web socket of which the valid transactions are most important for the context broker. The DeliverTX message is where each transaction in the blockchain is delivered with the message to be verified to all the peers that are connected as validators.[6] A validated transaction then needs to update the application state—by binding a value into a key value store, or by updating the UTXO database. The CheckTx message is similar to DeliverTx, but is only for validating transactions.[6] The Commit message is used to compute a cryptographic commitment to the current application state, to be placed into the next block header. This simplifies the development of secure lightweight applications, as Merkle-hash proofs can be verified by checking against the block hash, and the block hash is signed by a quorum.[6] From a context broker perspective, assets and the data payload which the assets carry are more important, and integrity checks can be made by checking against the block hash. Here, an assets-based communication model is preferred over a topic-based communication.

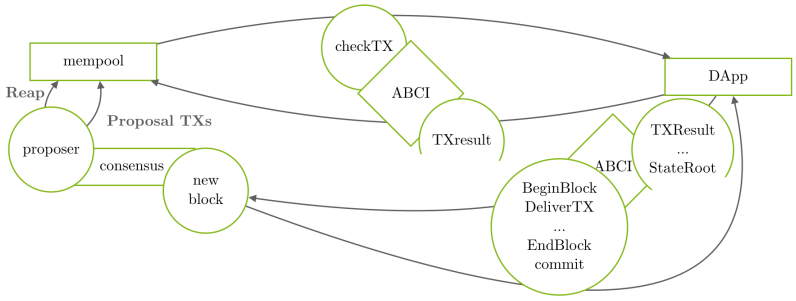

Fig. 4. Tendermint consensus protocol as used in dezCom

ISSN 2379-5980 (online)
DOI 10.5195/LEDGER.2019.173

4. Implementation of DezCom

In this section, a six node testnet dezCom stack is deployed and the results are analyzed for the framework and a use case scenario of heterogenous systems is also discussed in the following sections.

4.1. Use case—Being the goal of the socially-networked industry,[1] tracking the life cycle of customized product manufacturing through the whole supply chain, with support for various business services, is a suitable use case for dezCom. The use case takes the challenge of connecting flexible production stations in a decentralized approach as defined in Smart Micro Factory for Eletric Vechicles with Lean Production Planning (SMARTFACE).[8] The approach of messaging is addressed using assets, *i.e.*, the products in production as opposed to production stations directly or topics as in a publish-subscribe model. For example, consider a customized pre-order system of electric vehicles using a web interface. When an order is placed, an asset is created in the dezCom network. This created asset is then transferred between workstations with respective messages so that every production station can carry out their work on the product being manufactured. By the end of the production cycle, every message that pertained to the production of the asset is recorded in the blockchain.

4.2. Stack implementation—The dezCom testnet network was deployed with cloud servers in 4 major cities and two local instances in the logistics scenario. The cloud servers acted as validators, the local nodes ran an instance of Tendermint and connected to the cloud servers using the PEX protocol. Nodes that needed to communicate using dezCom hosted a Tendermint instance and communicated to the local instance. Nodes which did not meet the minimum requirements for running a Tendermint instance (*e.g.*, Raspberry PI) used the API interface to connect to one of the local instances. The messaging was relayed to other servers using Tendermint after the transaction was committed into the blockchain.

An asset is used as a topic for messaging in dezCom where it is initialized using a *CREATE* transaction in the blockchain, which has a unique hash ID for every transaction. Assets can be queried using this hash ID from the blockchain from any node locally and *UPDATE* transactions can be performed on these asset to change states. The data model for communication is not within the scope of this paper; here we try to evaluate dezCom as a reliable context broker where messages with a constant payload size are sent around for testing purposes. Topics play an important role in providing context to the messages, whereas in assets-based context broking, due to the possibility of collaborative machines, the asset dictates the decision of movement within the production or handling facility, rather than a process manager service. Because of this, if any node in the network were to be lost, the consensus would not grind to a halt. Lost nodes are not able to perform the necessary transactions, which triggers another station.

4.3. Results—Performance measurements were made between MQTT and dezCom where 100, 1000, and 10,000 messages were sent with 4 cloud servers acting as validators for dezCom each, with a 2x vCPU and 2GB of RAM hosted in 4 data centers (in New York City (NYC), London (LON), Amsterdam (AMS), and Bangalore (BLR) for location transparency). The local instances were run on an actual robot that otherwise also ran the MQTT client during operational tests for industrial scenario validations. The results, as plotted in Figure 5, show that the context broker has a constant delay when comparing the runs between 100 and 1000 messages. This delay was because of the decentralized consensus process before the messages were delivered.

ISSN 2379-5980 (online)
DOI 10.5195/LEDGER.2019.173

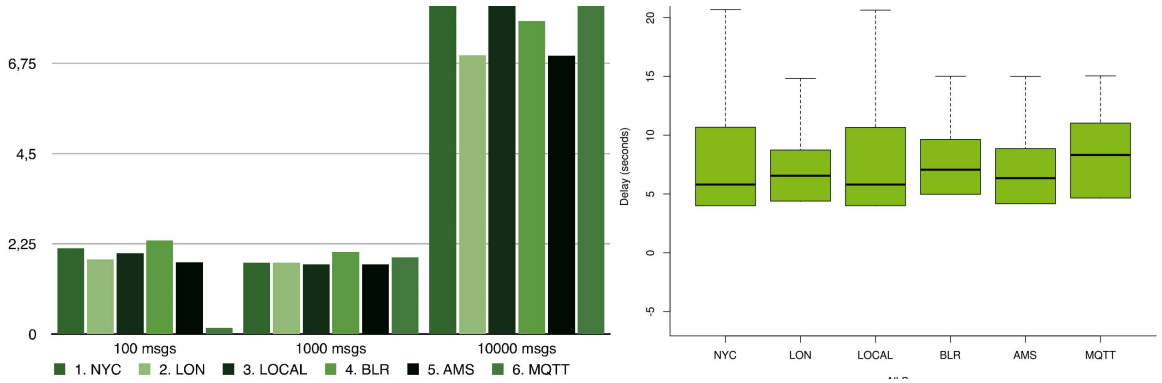

Fig. 5. Comparing message reception at various locations with MQTT, box plot with Maximum delay over all trails.

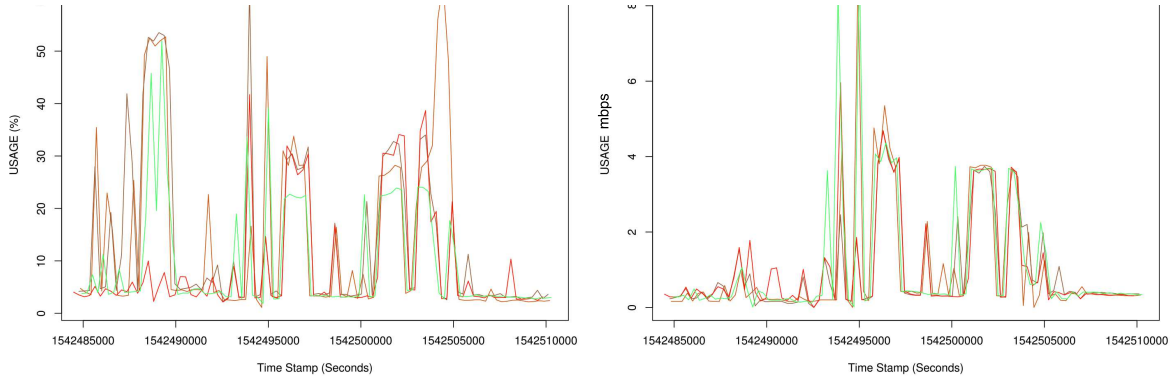

Fig. 6. Resource usage for cloud servers with Tendermint consensus

MQTT outperformed at a run where 100 messages were sent. During the trial run with 1000 messages, where MQTT performance decreased, dezCom had the same performance as 100, which was due to the availability of computation resource and message size that could be committed into a single block. There was another trial run with 10,000 messages, where dezCom performed as well as MQTT. MQTT stalled when a run with 100,000 messages was executed. In comparison, the dezCom local instance crashed due to inconsistencies in the socket interface driver, but the validators were not overloaded during the 100,000 request run—a fact which can be inferred from the data provided by the cloud service providers, as can be seen in Figure 6. The average time for preparing and sending the message was 18 ms. This was measured at every test and averaged throughout the tests. This number depends on the hardware where the messages were signed with the cryptographic keys, in this case, robots running ROS on a 4 core, 8 GB RAM processors were used.

4.4. Notes on production deployment—A load-balancer can be used to proxy the connections between the cloud servers which reduces any direct load to a specific server. The cloud servers for the tests were deployed with an SSL-terminated nginx web-server. The PEX protocol will gossip the other server locations within Tendermint, therefore one common address with the genesis file should be copied to all the node instances and allowed enough time to synchronize with the blockchain. In case of an industrial implementation throughout the supply chain, every location

ISSN 2379-5980 (online)
DOI 10.5195/LEDGER.2019.173

should host a validator node to improve the stability in messaging and reduce the geographic dependency of the context broker.

5. Conclusion

In the case of a socially-networked industry, where the field systems and the business logic have to organize themselves in a network and function collaboratively, truly decentralized systems are required. To build truly decentralized industrial applications, dezCom, a decentralized context broker, was conceptualized as a communication framework. The implemented communication stack was deployed in a robotics application for a goods-to-person picking scenario where the FMS, WMS, and the robots communicated using dezCom via transactions on a blockchain. This proof-of-concept implementation for partners to organize in a network and communicate in a socially-networked industry, with no single point of failure and almost no bottlenecks in scaling, was demonstrated with requirements such as location transparency, N-way scaling, and fault tolerance, which are not present in the centralized brokers that are currently used. The validator was set as a field on the transaction, which is suggested for future releases to create and validate transactions which are also physically responsible stations in the production or warehousing facility. Additionally, the use of consortium-based servers that can provide secondary support in validating the transactions is suggested for production deployment. Moreover, the ABCI currently used for the proof-of-concept was a Tendermint testnet. In future dezCom releases, transactions should be possible between two or more Tendermint blockchains, which will provide a true trust-less integration between networks *i.e.*, industries in the supply chain. Finally, in order to use this decentralized messaging system, a semantics for the transaction must be developed to generalize the process stages between participating entities using techniques from process interpretation, where the model is validated before propagating transactions into the chain.

Acknowledgements

Part of the work on this paper has been supported by Deutsche Forschungsgemeinschaft (DFG) within the Collaborative Research Center SFB 876 "Providing Information by Resource-Constrained Analysis," project A4.

Notes and References

[1] Dregger, J., Niehaus, J., Ittermann, P., Hirsch-Kreinsen, H., ten Hompel, M. "The Digitization of Manufacturing and Its Societal Challenges: A Framework for the Future of Industrial Labor." In *2016 IEEE International Symposium on Ethics in Engineering, Science and Technology (ETHICS)* IEEE 1–3 (2016) https://doi.org/10.1109/ETHICS.2016.7560045.

[2] Strobel, V., Castelló Ferrer, E., Dorigo, M. "Managing Byzantine Robots Via Blockchain Technology in a Swarm Robotics Collective Decision Making Scenario." In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems* International Foundation for Autonomous Agents and Multiagent Systems 541–549 (2018) https://dl.acm.org/citation.cfm?id=3237464.

[3] Cameron, A., Payne, M., Prela, B. "Research and Implementation of Multiple Blockchain Byzantine Secure Consensus Protocols for Robot Swarms." (2018) unpublished research (accessed 9 March 2019) https://courses.csail.mit.edu/6.857/2018/project/Cameron-Payne-Prela-ByzRobSwarm.pdf.

[4] FIWARE Foundation. "FIWARE: The Open Source Platform for Our Smart Digital Future." (2018) (accessed 9 March 2019) https://www.fiware.org/.

ISSN 2379-5980 (online)
DOI 10.5195/LEDGER.2019.173

[5] "MQTT Version 3.1.1 Plus Errata 01." *OASIS* (2015) (accessed 9 March 2019) Latest version: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html.

[6] Buchman, E. "Tendermint: Byzantine Fault Tolerance in the Age of Blockchains." PhD Thesis, University of Guelph (2016) http://hdl.handle.net/10214/9769.

[7] Venkatapathy, A. K. R., Riesner, A., Roidl, M., Emmerich, J., ten Hompel, M. "PhyNode: An Intelligent, Cyber-Physical System with Energy Neutral Operation for PhyNetLab." In *Smart SysTech 2015; Proceedings of the European Conference on Smart Objects, Systems and Technologies* VDE 1–8 (2015) https://ieeexplore.ieee.org/document/7323355.

[8] Blesing, C., Luensch, D., Stenzel, J., Korth, B. "Concept of a Multi-agent Based Decentralized Production System for the Automotive Industry." In *International Conference on Practical Applications of Agents and Multi-Agent Systems* Springer 19–30 (2017) https://doi.org/10.1007/978-3-319-59930-4_2.

SSN 2379-5980 (online)
DOI 10.5195/LEDGER.2019.173