Decentralised, Trustless Marketplace for Brokered IoT Data Trading

Shaimaa M. Bajoudah

School of Computing
Newcastle University

This dissertation is submitted for the degree of
Doctor of Philosophy

School of Computing January 2022

I would like to dedicate this thesis to my loving parents, brothers, sisters, husband and kids.

. . .

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in Newcastle University, or any other institution. Throughout this thesis plurals are used; this does not indicate multiple authorship unless stated explicitly.

Shaimaa M. Bajoudah
January 2022

Publications

1. *"Mind My Value: a Decentralized Infrastructure for Fair and Trusted IoT Data Trading". Missier, P.; Bajoudah, S.; Capossele, A.; Gaglione, A.; and Nati, M. In Procs. 7th International Conference on the Internet of Things, Linz,Austria, 2017.*

2. *"Toward a Decentralized, Trust-Less Marketplace for Brokered IoT Data Trading Using Blockchain". S. Bajoudah; C. Dong and P. Missier. 2019 IEEE International Conference on Blockchain (Blockchain), Atlanta, USA.*

3. *"Latency of Trading Transactions in Brokered IoT Data Marketplace in Ethereum". S. Bajoudah and P. Missier. 2021 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Internet of People and Smart City Innovation, Atlanta, USA.*

Acknowledgements

First and foremost, I would like to express my deep sense of gratitude and profound respect to my country the Kingdom of Saudi Arabia for giving me the opportunity to undertake a PhD at Newcastle University.

My extreme appreciation and thanks go to my supervisor Professor Paolo Missier for his patient guidance, motivating encouragements and unreserved advice throughout my whole PhD study. My appreciation also extends to Dr. Changyu Dong as a second supervisor; he directly contributed to the discussion of my thesis work with generous guidance.

Special thanks go to my sincere friends and my countrymen, especially: Abrar, Sara, Awatif, Hanadi, Aisha Belfaqeeh, Aisha Alarfaj, Ebtisam, Hanin, Muna, and Sabria, who were my family in this country and supported me morally and emotionally from the beginning to the end of finishing this work.

I would also like to offer my gratitude towards many colleagues from the School of Computing at Newcastle University who provided me with a great deal of additional insight, help and support during my four years of study, especially: Bakri Al-awaji, Amjad Aldweesh, Rawaa Qasha and Faris Llwaah.

Words cannot even come close to expressing my gratitude to my family, who deserve the credit for whatever positive achievements I have in my life – my parents, sisters and brothers. They have always supported me in all of my endeavours and have patiently waited for the completion of my doctorate.

Last but not the least, sincerest thanks go to my husband and children for their relentless support, encouragement and tolerance while I have been trying to finish my thesis.

Abstract

Trading data as valuable assets has become a trend. The use of real-time data generated from IoT devices provides a new insight into how to conduct a profitable business. As data marketplaces are becoming ubiquitous, it is also becoming clear that IoT data hold value for potential third-party consumers. This work introduces a marketplace for IoT data streams that can unlock such potential value in a scalable way, by enabling any pairs of data providers and consumers to engage in data exchange transactions without any prior assumption of mutual trust. It investigates the use of the power of blockchain technology in automating data trade agreements in a decentralised architecture. We present a marketplace protocol to support trading of streaming data, from the advertising of data assets and the stipulation of legally binding trading agreements, to their fulfilment and payment settlement, and managing trade participants' reputations. This work has two outcomes: a marketplace model and a reputation model. We present a decentralised, trustless marketplace for brokered IoT data trading, using Blockchain in Ethereum network that enables producers and consumers to start trading in the absence of trust; however, it is managed by a reputation model. Our marketplace is powered by a reputation system that is designed to address participants' trust and the reputation management of these traders in this marketplace. We mathematically define the reputation model by applying a reputation function to the marketplace participants – either producers or consumers – to quantify their trustworthiness in trading, based on various criteria. We evaluate the marketplace functionalities and its reputation model by designing a marketplace simulator. It is designed to simulate participant trading in the marketplace and how reputations are quantified based on rules and criteria defined in the system protocol. It is configured to replicate the behaviour of multiple pairs of producers and consumers in different trading scenarios and show how reputations are measured in these different scenarios. We experimentally show the trade-off between a trade overhead cost and the level of participant trust. On Blockhain Ethereum Mainnet, our system evaluates the latency of transactions an Ethereum takes to process and confirm our marketplace transactions.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

The Internet of Things (IoT) is a dynamic global network for objects that are connected together by the Internet, such as sensors, actuators and radio-frequency identification (RFIDs) to provide value added services. The term was first coined in 1998 and then defined as "it allows people and things to be connected Anytime, Anyplace, with Anything and Anyone, ideally using Any path/ network and Any service" [3].

Over the last decade, IoT data have become one of the traceable assets in our current ecosystem, while presenting opportunities for new insights into ways of doing profitable business for their owners and buyers. The monetisation of IoT data means that they have become one of the hottest businesses, not only for IoT device organisations, but also for individuals. With the current technical revolution, it is not surprising that any individual with a wearable watch can sell his/her information such as health records, racing records, fitness information or even medications in order to earn profits. The beneficiaries of these IoT data find much of the expected value in the streams of data generated by these devices. This value is realised by allowing IoT data trading to occur in a marketplace that rewards every single producer and consumer, at a very granular level. There is a need to find a platform for these data to be traded. As a consequence, data marketplaces have emerged. They represent a platform where data owners and buyers can negotiate and apply terms that both have agreed upon, in order to trade the desired data [4].

It is a platform that enables data vendors (entities that produce data) to offer their data for benefits, and buyers to purchase these data for different purposes such as analysis, aggregation or even reselling. The trading of these data is governed by different rules depending on the data type. For example, static data (data that remain unchanged for a long time, i.e., population numbers, geographical coordinates, i.e., population number, geographical coordinates, etc. [5]) are stored statically so that they are ready for consumers (buyers) to download, while dynamic data or real-time data (data that can change over time and lose their importance, i.e., current stock prices, weather data, etc.) have to be utilised by consumers immediately, otherwise they become useless and lose their meaning [6].

Most static data marketplaces are offering various datasets for sale, to individuals, government or industries, for the purpose of aggregation and analysis. They are also offered to buyers as whole datasets, cf. eg [7, 5, 8] for surveys on these.

Developing a marketplace for real-time data from the IoT is one of the most difficult challenges in the data marketplace due to their characteristics, which make them different from traditional datasets. First, IoT data devices generate heterogeneous data with different shapes and handling techniques due to the independence of IoT data owners [4].

A variety of IoT owners produce data from their own devices and sell them. Some of these generated data are measured in time and they need to be detected online and handled immediately, otherwise they lose their relevance [9]. Furthermore, the accumulation and rapid inflation of IoT data have attracted the attention of researchers, who aim to establish techniques and mechanisms for using these data values and then archiving them for historical purposes. As IoT data are time sensitive, a data buyer would have a trade agreement to buy data in advance, which is a challenging task. Buying data in advance means that the marketplace needs a mechanism to decide if the payment is made before or after delivering the data. If payment is made before the data are received, we require a system to fulfil the trade agreement between the data provider and the data buyer, enforce the contract and guarantee the delivery of the data to the buyers.

Application areas where interest in IoT data streams is growing range from health care [10] to personal fitness, smart cities [11], optimization of energy consumption at home, and many more. In each of these areas, much value expected from IoT data streams are delivered when they are aggregated and analyzed by consumers referred as Value Added Services (VAS).

Currently, there are many marketplaces that are tailored to deal with IoT data in different architectures and different areas of interest. Some of them leverage their marketplace with the use of *Blockchain*, which could be a vital component to enable decentralisation.

Misura in [4] introduced a centralised marketplace as a cloud application based on a queuing system. Each data provider registers their sensors into the application, along with all the details and description of the data. On the other hand, data consumers query the system for their needs and the system advises them on which data provider is the most appropriate. In a decentralised architecture, one example of the trading and monetising of data is represented in the Datum [12] marketplace. Datum is a blockchain data storage and monetization that allows a data provider (individuals or organisations) to store structured data securely in a decentralised way.

A data producer can monetise his data, created either from social networking or from wearable devices. He grants access to data under their precise terms, either for free or for a fee, for entities wishing to access his data.

Datapace [13] is a distributed and decentralised system based on the blockchain marketplace for trading IoT sensor data. It can also be used to sell or buy any type of data, independently of its type or provenance. It differs from Datum [12] in that it stores the hashes of data to be accessed by consumers.

While data exchange architectures based on brokerage systems such as MQTT broker [14] show potential large-scale open marketplaces for data producers to resell their real-time data streams for multiple parties, this could be used with a line of blockchain to separate the data exchange from the blockchain.There is no need to store all the data on the chain as in Datum [12] or store its hash, as in Datapace [13], where real-time data loses its value once it is generated. Instead, trade and settlement negotiations can be delegated to the blockchain.

As long as the data exchange is completed off-chain, there is no record which guarantees that the published data have been subscribed to by consumers. In such marketplaces, in fraud scenarios between producers and consumers it is difficult to resolve disputes when there is no evidence condemning the fraudulent party; this makes the marketplace blind in its judgements. This causes some risk for both the producer and the consumer, who trade with a party not trusted where there is no pre-mutual trust. It is necessary to guarantee that both of them behave honestly in an off-chain data exchange and to assess all of their trades in order to help make decisions for future trades.

The trust issue in online commerce platforms such as data marketplaces cannot be denied, as sellers and buyers are not directly connected physically [15]. With the absence of trust in these marketplaces, there is a need for a trust source that helps marketplace participants make a decision about their future trading partner, based on their trustworthiness. In centralised marketplaces, the trust element is provided by a centralised authority. Reputation systems may provide a level of trust for marketplace participants and could be a source of trust for their future decisions.

Reputation is the reliability of an entity in the overall public view [16]. In marketplaces, it measures the extent to which a participant – as a producer or a consumer – is trusted in the marketplace.

While reputation scores provide a level of trust, they can be used to assess the risk of entering into an agreement with an entrusted participant. Every participant is keen to build his honesty in the marketplace with a high-level reputation in order to gain more trading opportunities.

1.1 Research Aims

As highest beneficiary value of the real-time data is in its generation time, this research aims to study the capability of using the real-time data in the time of its generation in non-trusted marketplace environment. With the emerging of the new distributed ledger technology (blockchain) in IoT area, we examine if the blockchain could help facilitate trading a real-time IoT data without the need for trusted authority. We introduce a decentralised marketplace for exchanging real-time data in non-trusted environment with the power of the blockhain that is used to control and manage the trading affairs. The data exchange is made in a brokerage system such as MQTT broker, with the assumption of trust in a producer delivering his data stream (assuming no messages are lost and ignoring duplicate transmissions, as in MQTT QoS level 3). The use of the second generation of the blockchain Ethereum – smart contract – has automated the negotiation and fulfilment of a trade agreement between a data producer and data consumer in the marketplace. It should be designed with a reputation manager that quantifies their trust in the marketplace.

The leverage of the model is introducing a marketplace with the lowest monetary and storage cost for participants. Blockchain reduces the overhead costs of communications with low-priced transactions compared to traditional transactions. The model connects high trust with low monetary cost, with minimal fees for blockchain transactions. It does not use extra decentralised storage to store the stream of IoT data, as it becomes valueless over time.

1.2 Research Objectives

The objectives of this work are the following:

1. Investigate the capability of the blockchain technology to provide a marketplace for trading IoT data - as its main assets - in a trustless environment.
2. Study the ability of the second generation of the blockchain Ethereum – smart contract – in automating and monitoring the trade affairs of streaming IoT real-time data without storing the IoT data in the blockchain.
3. Evaluate the ability to leverage the marketplace by designing a reputation system that provides a level of trust for pairs of trade participants with an absence of mutual trust.
4. Evaluate the marketplace functionalities in a real blockchain network such as Ethereum with a test real stream dataset.

1.3 Research Contributions

With the assumption of having a decentralised marketplace for IoT data leveraged by blockchain technology, we guarantee the trustworthiness and transparency of a trade agreement fulfilment. The immutability of transactions recorded in the blockchain is one of its strength and the key factor to providing a digital trustless environment for businesses. We envision a decentralised marketplace for real-time IoT data on a blockchain that Ethereum is capable of supporting, with no trust between its participants. At the same time, it quantifies the level of trust of each participant with our proposed reputation system. The reputation system is tailored to address the trust and reputation management of participants in our marketplace.

This work provides knowledge of how to: (1) design the protocol of a fully decentralised brokered IoT marketplace, free of mutual trust between participants; (2) support the model with the use of blockchain, which ensures limited scope for fraud on either side; (3) design a model that configures participants' reputation scores, which define their trust level in the marketplace; and (4) validate the model by experimentally deploying the marketplace that is integrated with the reputation system on the main blockchain network.

1.4 Thesis Structure

Chapter 2 presents the background and work related to the available IoT data marketplaces and reputation systems. It shows the state of the art of blockchain inventions and their applications and integration into IoT data marketplaces.

Chapter 3 presents our initial model that uses blockchain to build a decentralised, trusted, transparent and open architecture for IoT traffic metering and contract compliance in a brokerage platform.

Chapter 4 shows the detailed elements of our decentralised trustless brokered IoT data with the use of blockchain.

Chapter 5 presents the protocol of the reputation system that is tailored to fit with our marketplace model.

Chapter 6 shows the technical architecture of the marketplace. It also presents the implementation and the evaluation of the deployed marketplace on the blockchain Mainnet.

Chapter 7 discusses the system limitations and summarises the full marketplace model with its reputation system.

1.5 Code Availability

The code of the marketplace smart contract, written with Solidity, is available on the Github repository https://github.com/ShaimaaBaj/Marketplace.

Chapter 2

Background and Related Work

2.1 Introduction

This chapter presents the state of the art of IoT marketplaces, blockchain and reputation systems that are applied to marketplace participants. It introduces the blockchain innovation and how it is integrated with the IoT field for various applications. It shows the current market for trading IoT data streams that are generated from IoT devices. Moreover, it discusses different marketplaces that are leveraged by blockchain and some approaches that quantify their participants' levels of trust.

2.2 Blockchain

The digital world changed when Satoshi Nakamoto introduced *Bitcoin*, which relies on blockchain [17]. The main concern of Nakamoto was providing trust in online payments between parties without going through a financial institution and simultaneously preventing the double spending problem. This led him to introduce *Bitcoin* as an electronic payment system based on cryptographic proof instead of trust using a peer-to-peer network.

The proposed solution by Nakamoto centred around the idea that any currency transfer transaction in the network must be timestamped and then hashed into an ongoing chain of hash-based proof of work, thereby forming immutable records that cannot be altered without redoing the proof of work. This is simply *The Blockchain* – the dawn of a new revolution in the digital world.

Although a blockchain was introduced in 2008 underpinning the cryptocurrency Bitcoin, its initial concept was depicted in 1991 by Stuart Haber and Scott Stornetta, who proposed a method that prevented tampering with the cryptographic hash linked to timestamped digital documents [18].

The blockchain, as defined in [19], is the public ledger of all Bitcoin transactions that have ever been executed. It is a fully-distributed, peer-to-peer network platform that can record immutable data and host applications and even transfer the ownership of digital assets that have monetary value in reality [20]. Blockchain technology is the foundation stone of decentralised trustless transactions where parties' transactions are completed without inter-mediation, through a decentralised structure on a global base. It is a distributed ledger that provides: (i) secure transaction settlement; (ii) completion of transactions; and (iii) low cost of asset transfer [21].

Blockchain technology is not a new concept in the computer community, it is just the result of the combination of three old concepts: peer-to-peer networks, public key cryptography and distributed consensus based on the resolution of random maths challenges (Cryptographic hashing) [20, 22].

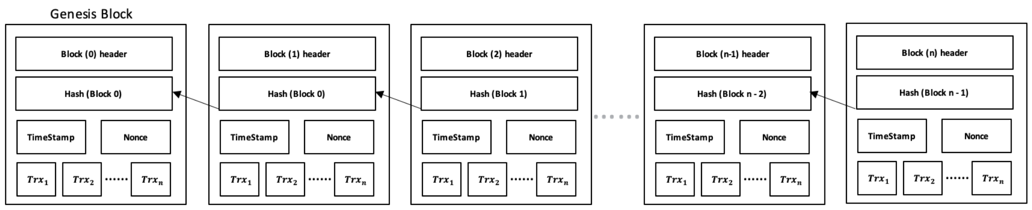
A blockchain is similar to a traditional database in some of its functionality. As long as the databases are considered as a data structure, a blockchain is a way of structuring the transactions into a chain of blocks. While they both provide a structure for storing data, traditional databases apply ***CRUD*** operations (Create, Retrieve, Update, Delete) on their data, while blockchain provides ***'Writes operation ONCE'*** and it can only retrieve the transaction information.

Money transfer problem has been addressed by the use of blockchain where it explains the blockchain concepts and principles. Suppose we have a network connecting a number of people and they each want to transfer money to each other. This is one of the crucial concepts and the basis for building a blockchain; it is a *P2P network* where the central third party is removed and instead, every node is able to communicate with any node in the network directly. In P2P networks, a mutual service is provided for every node such that it provides services to other nodes and simultaneously can ask for services [23].

Any person in this network is able to send as much money as he wants to another node in the network in the form of a new transaction. This depicts blockchain transparency, which is known as *an open ledger* where any transaction between two nodes is added to the ledger. The chain of the transaction has new blocks added to it, therefore it is known as the blockchain [24]. This chain is open and public to everyone present on the network. More precisely, any node can track the money transfer movement in the network and how much money each node has in its wallet.

The distributed ledger is one of the most popular blockchain principles. One of the main goals of blockchain technology is granting the same privileges to every node in the system to

Fig. 2.1 Example of Blockchain [1]



remove the centralisation infrastructure such that every node has the same copy of the open ledger. Likewise, every node has the privilege to check the transaction validation individually. The distribution architecture gives a node the ability to have its own synchronised copy of the all-open ledger without the need for a centralised place to retrieve information from. The trust between nodes in this trustless distributed architecture is built through mathematical methods [25].

Moreover, one of the most cited blockchain features is the ability to build economic systems that make the people existence unnecessary, which could be considered "trust-free" [26]. In non-academic setting, it is referred to as "the trust machine" [27]. It substitutes the trust dimension and shows that the trust during the transactions is superfluous [28–30].

The author in [31] considered the blockchain to be one part of the fourth industrial revolution, since the invention of the steam engine, electricity, and information technology, due to its ability to change the current state of the internet from "The Internet of Information Sharing" to "The Internet of Value Exchange".

Blockchain facilitates the exchange of a value with the absence of an intermediary, in a decentralised and a secure manner. As long as sharing economy of the world is growing rapidly, blockchain provides a public medium for anyone, either individuals or organizations, to share their data with presenting different sharing economy business models. Blockchain is expected to contribute in the growth of the global economy with various business models in different areas [32–34].

2.3 How Blockchain Works

Logically, a blockchain is a chain of blocks linked together in a secure way, grouped together in a P2P network. An illustrated example of blockchain technology is given in Figure 2.1. It consists of data grouped in packages (*blocks*) where a block is the fundamental storage space of information. It consists of many transactions and every block is linked into the chain to extend the open ledger.

As shown in Figure 2.1, every block in the chain consists of: (i) transactions, (ii) a timestamp, (iii) the hash value of the previous block and a (iv) nonce, which is a random number for verifying the hash. The first block in the chain is called the “Genesis block”. To guarantee the blocks’ integrity, each block stores the hash of the previous block (“parent”). The first block stores its own hash, while the second block is linked to the first block (genesis block) by storing its hash value, and so on.

Before adding a new transaction to a block, a transaction must be issued, verified, given its own hash and then stored in a block. Any transaction issuer must be a node in the P2P blockchain network where a cryptographic proof of identity is used. A transaction issuer must have a pair of cryptographic public and private keys to sign the transaction, and also to verify his identity in the network. The transaction is then broadcasted into the transaction pool of the blockchain network waiting to be verified.

A new block is created from a group of transactions that occurred at this time. It is generated due to the consent of a number of approved nodes [25]. Every node competes to verify the validity of each transaction and the validity of the block itself. If the majority of network nodes agree by a consensus mechanism, this block will be added to the chain [35]. This stage is achieved by following consensus algorithms, which is called mining [25]. Mining means reaching a consensus on the current state of the blockchain ledger [36].

The Consensus mechanism is defined in [19] as “the process in which a majority (or in some cases all) of network validators come to agreement on the state of a ledger. It is a set of rules and procedures that allows maintaining coherent set of facts between multiple participating nodes”. It is a process achieved by nodes in the blockchain network that are called *miners*. Miners are unique nodes that verify the transactions and then add them to the chain [37]. They verify all the transactions in the block and solve the mathematical problem of the digital signature using a hash function. The miners compete with each other, and when the first one solves the problem, its copy of the solution will be distributed to all the nodes in the blockchain network, and the winning miner receives additional bitcoins as rewards. Other miners accept the proof of work, and the new block will be added to the blockchain network [38].

It has been said that blockchain/P2P might do for transactions what the Internet/Web did for Information [23].

2.4 Blockchain and IoT Data Streams

Swan in [19] suggested that the development of blockchain applications could be divided into three categories: Blockchain 1.0, 2.0 and 3.0. *Blockchain 1.0* is cryptocurrencies deployed as a P2P cash payment system. *Blockchain 2.0* is the use of blockchain in the entire slate of economic, market and financial applications that are more extensive than simple cash transactions such as stocks, bonds, loans, smart property and smart contacts.

Blockchain 3.0 is the use of blockchain in different areas beyond currency, finance and markets by developing blockchain applications in areas such as government, health, science, IoT, literacy, culture and art. A number of blockchain-based applications are presented in [39–42].

The primary goal of the blockchain technology as introduced by Nakamoto is to free people from any form of trust that we are now forced to give to intermediaries who regulate and "manage" a large part of citizens' life. Although there are many distributed systems based on consensus algorithms, the blockchain is the only one that enjoys the following properties: (i) trustless – no need for a third party to identify and check nodes' identities; and (ii) censorship resistant – as blockchain is a network without a third party, where entities trust only the quality of the cryptographic algorithms that govern the operation, anyone can get involved as a node in the P2P blockchain network [37].

Due to the immutability, transparency and trustworthiness of all transactions executed in a blockchain network, this innovative technology has many potential applications [43]. Blockchain as an immutable open ledger or, more precisely, a distributed database of records, has been used in many fields, including some beyond currency and finance. It is a reliable infrastructure for creating platforms that provide various solutions in areas such as healthcare, transportation, IoT solutions, supply chains, etc. [44].

In addition to the first announcement of the digital currency *Bitcoin*, other currencies have gained prominence such as Ripple [45] and Stellar [46]. However, Ethereum [47] has gained more attention and its native cryptocurrency token *ETH*. follows Bitcoin as the second largest cryptocurrency.

Ethereum is a decentralised open-source, blockchain-based platform that uses its own currency, *ETH*. Ethereum is the most actively used blockchain; it provides Smart Contract functionality to be run without any downtime, interference from a third party, control or fraud. Ethereum blockchains have been leveraged to manage Smart Contracts to build Distributed Applications (ĐApps). A smart contract, as defined in [48], is "a computerized transaction protocol that executes the terms of contract". It permits trusted transactions and agreements to be carried out among parties with no need for mutual trust between them, or a central authority, legal system or external enforcement mechanism. It is a way of enforcing agreement obligations between parties automatically by writing a piece of code using Solidity [49]. Solidity is an object-oriented language initially proposed in August 2014. It is designed for developing smart contracts that run on the Ethereum Virtual Machine, also known as

EVM. It is a *Turing-complete bytecode language* that uses arrays, variables and constructors to write a code that runs on EVM. Ethereum's smart contract considers an account as a normal node when it is deployed in the blockchain to enforce the *if this then that* rules that perform certain jobs.

The emergence of blockchain technology could be a vital component for the decentralisation application in the digital world. Decentralisation architecture emphasises the lack of reliance on a third party and trusting an individual node in various areas. It has been shown in various marketplaces in various fields that no failure points are leveraged by blockchain. Cryptaur [50] is a decentralised Ethereum-based ecosystem that enables peer-to-peer interactions between suppliers and consumers to offer goods or services that can be bought in the marketplace. In a different area, Fysical [51] is a decentralised location data marketplace using an Ethereum platform. For individuals, DataWallet [52] is an ecosystem that enables the trade of users' personal data in return for profits.

Blockchain and IoT are a part of decentralised applications and a way of reshaping our future. Objects around us are connected to each other and they generate a massive amount of data that IoT marketplaces can trade in and monetise. New forms of dedicated data marketplaces are emerging to help unlock such value [4], but these are comparatively less mature than more traditional data marketplaces for static data, cf. [7, 5, 53, 8] for surveys on these. IoT marketplaces leveraged by blockchain technology do not store all the data on the chain as blockchain is not designed to record that much data. Datum [12] is an IoT marketplace application that is designed to trade stored IoT data based on the Ethereum Blockchain. A data provider stores his structural data securely in BigchainDB [54] and the Inter Planetary File System IPFS [55]. The same occurs in the marketplace RepuX [56], which gives small and medium enterprises (SMEs) the opportunity to monetise their massive store of generated data. All of these generated data are stored on decentralised storage technologies such as IPFS and Sia [57], and then the smart contract is used to grant access to buyers.

Although the new trend in the IoT data marketplace is profiting from the short-lived nature of IoT streams, real-time IoT data streams tend to lose their value if they are not consumed in near-real time. Trading real-time data streams generated from the IoT is the core tradeable element in such a marketplace as they are increasingly viewed as tradeable assets of value to all parties in the trade, i.e., to third-party buyers. In these marketplaces, Misura in [4] introduced a centralised marketplace as a cloud application based on a queuing system.

By moving to the decentralisation architecture that removes the middle trust and the centralised point of failure, if the marketplace is leveraged by the blockchain, we could address the IoT stream marketplace requirements without a trusted party to control the trade contract obligations and payment settlement. This adds transactionality to each of their interactions, before, during and after data exchange. As mentioned earlier, IoT data streams will not be stored on the blockchain as this is not designed to store a massive amount of data. Instead, blockchain may be used in the fulfilment of trade obligations, data exchange monitoring on the brokerage platform, or payment settlements.

The vision of our model is to introduce a decentralised marketplace for real-time IoT data with the use of the distributed ledger (Blockchain), which can be used by both enterprises and individuals. It should be designed for participants who trust each other based on their reputation, which is quantified by a reputation system. It should benefit from the innovation of the smart contract to automate the negotiation and fulfilment of trade agreements between data producers and consumers and the data exchange should be done off-chain through a brokerage platform. It should be powered by a reputation system that quantifies the trust of the participants and there will no longer be a need for mutual trust to establish trading between producers and consumers.

Suliman et al. [58] propose a marketplace to monetise IoT data using a smart contract in the blockchain. Similar to our model, their approach involves sending IoT data through a Message Queuing Telemetry Transport *MQTT* broker and using smart contracts to manage and settle payments. The main difference with our approach is that a deposit is required before subscription to a topic may take place. This conflicts with our no-trust assumption, as leaving a deposit ahead of receiving goods is likely to be viewed as risky by the buyer. Huang et al.'s decentralised platform for IoT data exchange [59] comes close to addressing the issues of mistrust amongst participants, and, similar to our approach, the data are exchanged off-chain and made available to buyers once the contract is in place. However, the data to be purchased is stored, making this solution unsuitable for streaming. Furthermore, no guarantees are offered to ensure that the data are genuine, so advance payment, i.e., to gain access to data downloads, is risky.

Another effort has emerged in the IoT marketplace in the area of data source verification. Datapace [13] is a distributed and decentralised system based on blockchain with technical and policy-based data verification. It is a marketplace for IoT sensor data where the IoT sensors are connected the IoT platform "Mainflux", which is integrated into a Datapace system part called the "Datapace IoT platform". The difference between this model and our model is that this model provides data source verification by its own sensing equipment, while our model assumes the honesty of data source producers and no special verification hardware is required.

Similarly, AnyLedger [60] is an embedded wallet for IoT devices that connect the physical world to the blockchain. Each IoT device will be able to execute transactions to the blockchain. It is the first IoT-blockchain application enablement platform, starting with hardware devices and the embedded software, and finally ending with remote device management and blockchain nodes. Based on their white paper [60], the AnyLedger blockchain solutions allow seamless deployment of tamperproof sensors, which are remotely controlled. It uses an IPFS file system [55] as a decentralised storage end point for secure storage and data monetisation.

Finally, a recently proposed alternative blockchain provider, IOTA [61], announced their support for decentralised marketplaces at the end of 2017, with the goal of "enabling a truly decentralised data marketplace to open up the data silos that currently keep data limited to the control of a few entities". One distinguishing feature of this solution is that, unlike the others cited above, here the IoT data are actually stored in the blockchain (or IOTA's version of it, called the Tangle) [62]. To the best of our knowledge, this solution has not yet been released.

As opposed to IOTA, Streamer [63] felt that there was no need to develop a completely new blockchain and instead, they saved resources by using the existing Ethereum blockchain. It is a real-time data stream exchange platform. It creates an ecosystem for data producers to sell their data to consumers. As explained in their white paper, a data producer creates data streams for their data and pushes them to brokers' nodes, which are responsible for delivering them to their data consumers, who purchase the desired data by interacting with the Ethereum smart contract for management, data permission and payment.

2.5 Blockchain and Reputation Systems

Blockchain technology can leverage fair and trusting marketplaces for IoT data to be exchanged, but it is not adequate for managing and controlling the data exchange outside of the blockchain network. As the long as the data exchange is done off-chain, there is no record that tracks whether the generated data have been published and the subscribed data delivered. It is difficult in such fraud cases between producers and consumers to solve disputes when there is no evidence condemning the fraudulent party. Designing a marketplace with the leverage of blockchain with no data exchange on the chain makes the marketplace blind to judges in disputes.

When there is no mutual trust between data producers and data consumers before becoming involved in a trade, there are risks for both of them. There needs to be a guarantee that both of them will behave honestly in an off-chain data exchange and an assessment of all the trades they are involved in, to help others make a decision about future trades with them.

Sustainability is a vital factor in any live marketplace. It is based on the behaviour of stakeholders in the marketplace who are involved in a trade and complete it successfully. While participants may use reputation scores to assess the risk of entering into an agreement with an entrusted participant, it is obvious that each participant seeks a high reputation score in order to build their honesty in the marketplace and therefore have more trade opportunities.

The trust issue in online commerce platforms cannot be denied; sellers and buyers are not directly connected in a physical location, and the same applies to data marketplaces [15]. In centralised marketplaces, with the absence of mutual trust between participants, the trust element is provided by a centralised authority. In trustless marketplaces, there is a need for a source of trust that can help trade parties to make the best decision about trading with another party.

Trust and reputation have many different definitions [64]. Trust is defined in [65] as a conviction that an entity is straightforward and will not harm other entities. On the other hand, reputation is a global perception of an entity's behavior based on the trust that other entities have established. It is a more objective belief than trust. Also, a reputation is defined as the reliability of an entity in the overall public view [16]. In marketplaces, it measures how much a participant – as a producer or consumer – is trusted in the marketplace.

Reputation reflects the estimation of an expected behaviour that is quantified from mathematical observations for all recorded behaviours in the marketplace. It can be calculated from all the participant's past behaviours or it can be based purely on experience with an exact party [66, 67].

Many blockchain-based reputation systems have been introduced in various domains and they each follow a different approach. A variety of research angles has been utilised to design a reputation system that fits these marketplaces and quantifies the participants' trustworthiness. One of the first attempts to introduce a generalised blockchain-based reputation system was by Dennis et al. [68]. The authors in this paper introduced reputation scores as either 0 or 1, such that a negative reputation is denoted by 0 and a positive reputation is denoted by 1. They rely on a numerical reputation dimension to quantify their users' trustworthiness. The reputation system lets its users identify the parameters to be used in the score calculation.

A textual reputation dimension was introduced in [69]. The authors in [69] developed a novel review system based on the Ethereum smart contract. It provides the opportunity to submit a written review about the data sold and it is immutably stored on the blockchain. The system showed that it guarantees the submitted reviews' integrity as the reviewer must enter a unique value that reflects the exact data: "the hash value of the IoT data transaction or metadata should be entered during the review registering process". Although a fake review can no longer be submitted due to the unique value required, a string review does not necessarily reflect the trustworthiness of the data seller or the data quality. Instead, it costs the reviewer more gas to submit the review on the blockchain. The long string review (the transaction payload), the more gas should be paid. A data consumer could write a very short review in order to pay less gas, and there are no incentives for him to reflect the truth through the review. The authors plan to improve their system to incentivise the reviewers in their future work plan.

The same reviewing system was introduced in [70] but they used IPFS to store the reviews off-chain. They showed that their system allows users to submit textual reviews on the services they have received through a smart contract and to use the IPFS file system [55] to store reviews immutably. It guarantees linking the consumer (who had the experience with SP) to the review by creating a unique token that links the user's address to the review. It also rewards the consumer with some Ether (set by SP) for his reviews. The system creates the token using Keeca256 hash to link the user with the reward. Moreover, huge storage is required to enable unlimited textual reviews to be stored on the IPFS file system [55]. It is a reviewing system only, and it does not show any numerical rating or calculate any overall reputation for the reviewee.

Another approach relying on reputer-rewards was introduced in [71]. Carboni presented an incentive-based reputation system. The system introduced a "voucher" with vote fees (3% of the service payment) as an incentive for a consumer to rate the service. The voucher should be signed digitally by both the buyer and the seller. The seller's reputation is calculated based on the sum of these fees. It provides an incentive to leave a rating for the seller to obtain the fees but it does not necessarily reflect the service quality or the seller's honesty.

The same approach of incentive-based reputation systems was introduced in [72], for a blockchain based emission trading system. The authors presented a novel reputation-based trading system that is relying on two motivations: direct financial incentive and improved public perception. It is based on two criteria to figure out the reputation such that it is a function of past emission rates and participant's strategy to achieve the emissions reduction. Participants with high reputation are able to choose a better trade offer and to conclude the trade faster.

Different approach was introduced in [73] relying on a reputer's information from online sources. It collects all valuable information from a number of online sources, and use that information in the calculation of the reputer's rating.

Sharples and Domingue in [74] introduced a blockchain solution for educational records related to reputational rewards. They showed that every educational institute or individual has an "educational reputation currency", which is called "Kudos", as an initial award. They are actually trading their educational reputation as a currency. Every user in this educational network has been rated by reputational transactions and therefore they earn more Kudos. More Kudos means a higher reputation. The paper does not show how they compute the reputation or what criteria should be considered before submitting the reputational transaction to the reputee.

2.6 Blockchain-Based Reputation System for real-time IoT data Marketplace

Deriving from the primary goal of the blockchain technology as introduced by Nakamoto is existing applications with the absence of trust. It is a good choice to build decentralised marketplaces for real-time IoT data trading that use the blockchain technology as a vital component, and remove the centralised trust. On the top, the need of a trust source is raised to act as a decision maker guider. With the trust absence between data producers and data consumers before becoming involved in a trade, the probability of a trade risk is high. Based on the literature that shows blockchain-based reputation systems, the needs and the criteria that are measured to calculate the reputation score for every participant are vary, follow different approaches, and applied in different domains. Our approach involves designing a system that quantifies reputations viewed by a user, as explained by [75]. The reputation is given to a user based on the average of all of his scores so as to prevent a collusion attack.

As long as the lower in cost is undeniable financial factor for data consumer, we envision that participant reputation in our marketplace will be as inexpensive as possible, with the lowest monetary and storage costs. It should not be not review based, as written reviews incur more gas and therefore more fees. In addition, it should be designed to be simple with no extra storage for written reviews, either on the blockchain or distributed storage platforms such as Swarm [76] or IPFS [55]. Instead, we propose a model that quantifies participants' trustworthiness based on their activities and the trades they have been involved in. The participants' scores are stored on the blockchain in a numerical representation.

2.7 Conclusion

This chapter showed the state of the art of current IoT data marketplaces, and how these are leveraged by the blockchain with the absence of mutual trust between participants in these marketplaces. It also presented different reputation systems that are employed to qualify the trustworthiness of the participants. It showed various examples of blockchain-based marketplaces that benefit from its smart contract for the fulfilment of their trade obligations. Most of the mentioned marketplaces were critically compared with our model, and their similarities and differences were elucidated.

Chapter 3

Brokered IoT Data Marketplace

Most of this chapter materials have been published in the paper:

”Mind My Value: a Decentralized Infrastructure for Fair and Trusted IoT Data Trading”. Missier, P.; Bajoudah, S.; Capossele, A.; Gaglione, A.; and Nati, M. In Procs. 7th International Conference on the Internet of Things, Linz,Austria, 2017.

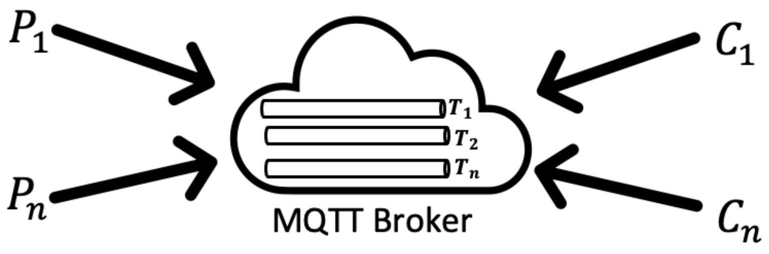
3.1 Introduction

IoT data are increasingly viewed as a new form of massively distributed and large-scale digital assets, which are continuously generated by millions of connected devices. The search for greater profits and the continuous desire to achieve greater economic gains have led to the use of IoT data as tradeable assets. Considering data from IoT sensors as tradeable assets is closely related to that of Sensing as a Service (SaaS) models, or even Sensing and Actuation as a service (SAaaS) [77]. Both are derivatives of the more general “Everything as a Service” (XAAS) cloud-based model for data exchange [78].

Furthermore, as data marketplaces are becoming ubiquitous, the real value of such IoT data assets can only be realised by allowing data streams generated from IoT devices to be traded. They have become more of a commodity and their tradability is a crucial factor for most businesses [7]. They offer opportunities for profitable benefits and rewards for every single producer and consumer, at a very granular level.

It is not surprising that data streams generated from IoT devices represent much of the expected value of IoT data [7]. They are offered in markets in various application areas, from health care [10] and personal fitness, to smart cities [11], the optimisation of energy consumption at home, and many more. In each of these areas, the value of the IoT is only delivered when the continuous data streams produced at the edge of the network are

Fig. 3.1 The Standard IoT data streaming with MQTT broker



aggregated and analysed by data consumer processes, hereafter referred as Value Added Services (VAS).

An example of one of the emerging applications is public transport networks like the London underground. Although the transportation authority uses data on the density of personal travel card swipes over time at individual metro stations, this is also useful to taxi companies. They can detect any anomalous passenger traffic patterns, and consequently re-route their fleet to the right stations at the right time.

In this chapter we propose an initial technical infrastructure for IoT data that are exchanged on a brokerage platform. We show the many requirements that should be applied to the model to achieve the maximum possible value from trading these IoT data on this brokerage platform. We present a conceptual model for tracking brokered IoT data flows from gateways to VAS in the cloud, which embodies a methodology to achieve granular metering of IoT data trading. For contract settlements, we explore the use of blockchain technology and smart contracts to remove the need for a centralised trust. In the last section, we carry out an experimental evaluation identifying the viable boundaries for the prices of digital assets, which make the trading infrastructure economically sustainable. We also assess the capability of Ethereum smart contracts to handle a stream of contract settlements at varying arrival rates, and conclude that they are indeed a viable option for the validation of contract compliance.

3.1.1 Publish/Subscribe IoT Data Architecture

The standard common IoT data streaming infrastructure, as shown in Figure 3.1, shows the exchange of streaming data between any pair of participants, i.e., a data Provider P and a Consumer C, mediated by some transaction-agnostic broker infrastructure.

The flow of IoT data is generated from data producers, for example sensors, and it is pushed to an intermediator broker through pre-defined topics. All these messages are transported using an MQTT protocol [14] to be subscribed to by data consumers. The MQTT broker is a server; it runs a lightweight, publish-subscribe MQTT protocol that receives all the messages from the data producers, and then routes them to subscribed consumers. We adapt the popular open-source Mosquitto MQTT broker [79] to add traffic metering capabilities to count the messages between each pair of P, and C.

3.2 Brokered IoT Data Marketplace Requirements

The data marketplace is a platform that enables data vendors (entities produce data) to offer their data for benefits, to buyers who would buy those data for different purposes such as analysis, aggregation or even reselling. We believe that such a marketplace should not be owned by anybody, but should instead fairly and transparently self-enforce a well-defined set of governance rules. We propose an initial technical infrastructure for a new kind of data marketplace that, in the long run, meet the needs of trading a stream of real-time IoT data in such decentralized platform that leveraged by blockchain technology. Also, it should be provided with fair governance rules that guarantee fairness transactions in such environment with a mutual trust absence. This infrastructure is designed to meet six main requirements that we believe it gains the most benefits of a marketplace for their primary parties in such decentralized marketplaces. Trading a stream of real-time IoT data in a decentralized environment with an absence of mutual trust should be designed to guarantee; (1) flexibility, (2) availability, (3) decentralisation, (4) support for data stream exchange, (5) dispute resolution and trade settlement, and (6) assessment of participants' trustworthiness.

First, the marketplace should be a fertile environment for unanticipated kinds of business relationships. It should be dynamic and flexible, and it should be possible to quickly establish and then fulfil contracts between one and possibly many producers and the VAS, with guarantees of compliance and fairness.

Second, the marketplace should be available for everyone for trading practice. It should not be for organisations only; individuals can also be a part of this marketplace and gain value from their data. For example, today it is possible to quantify an athlete's effort during a competition using a number of wearable devices, from a bio-harness to accelerometers and video feeds. One can imagine that individuals may decide to let VAS access their data feeds, in return for some benefit (monetary or otherwise). In the near future, athletes may be able to sell these feeds to their followers who are interested in tracking their competitions online. There are examples in the UK today where individuals receive heavy discounts on smart watches from health insurance companies, provided they let the company access their fitness data.

Third, the key marketplace characteristic is allowing IoT data streams to be traded. This is not usual: a 2012 survey of data vendors [5], for example, included 46 data suppliers, but the definition of data marketplace used in the paper is generic (“a platform on which anybody can upload and maintain data sets, with license-regulated access to and use of the data”) and it is geared towards static data, like Microsoft’s Azure Data Market. In contrast, our requirement entails the typical “Big Data” challenges of high volume, high velocity, and high variety in the streams.

Fourth, the marketplace should be completely decentralised. It should be run based on pre-defined rules that are stated in a contract and it should stipulate sanctions when the rules are violated.

Fifth, the marketplace should resolve participants’ disputes and completely settle trade payments with guarantees of compliance and fairness.

Finally, the marketplace should show a participant’s trustworthiness or dishonesty in their historical trading in order to build a reputation image for every participant in the marketplace. This is to help classify participants into different levels of trustworthiness and therefore establish unanticipated business with different participants.

3.3 Brokered IoT Data Marketplace Models

Data exchange architectures based on message brokers systems such as MQTT broker allow a single data stream to be delivered to multiple parties. This could lead to potentially large-scale open marketplaces where data producers can resell their streams in real time, multiple times.

While the IoT network and message-passing infrastructure can support a scalable marketplace, this inevitably leads to issues with mutual trust between participants, especially when these have no prior reputation within the marketplace. Also, the short-lived nature of streams requires efficient, automated mechanisms to create legally binding trade agreements, including payment arrangements, and to enforce such agreements throughout data transmission.

In our base assumption, we use trust as a parameter for initial requirements, and then we propose a variety of marketplace models with different network architectures: (i) a centralised trusted broker; (ii) decentralised trusted parties; and (iii) a trustless decentralised marketplace with a trust blockchain network (see Chapter 4).

3.3.1 Centralised Trusted Broker

Following the common brokered IoT infrastructure for streamed IoT data exchange in section 3.1.1, it is initially assumed that the broker is trusted.

Let $P = p_1 ... p_n$ and $C = c_1 ... c_m$ denote the set of producers (IoT devices) and consumers (VAS) that participate in the trading, respectively. In the standard publish/subscribe model for data brokering, the p_i act as publishers and the c_j are subscribers.

These participants agree on a set $T = t_1 ... t_r$ of topics. In IoT data brokering, messages are generated by gateways, which are responsible for segmenting raw data streams from edge devices into discrete messages. The topic associated with each message describes the type of data stream, for example "heart rate", "GPS track", "glucose reading", "energy reading", etc.

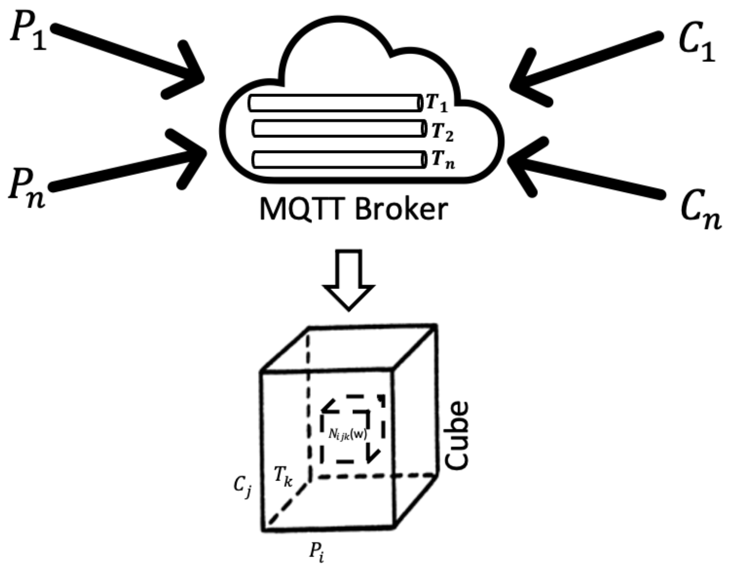
Suppose p_i publishes data on a set of topics $T_i \subset T$. A consumer c_j enters into a contractual agreement with a producer p_i by subscribing to a subset $T_{ij} \subset T_i$ of the topics available from p_i, possibly only for the duration of a time window $W = [w_s, w_e]$. Such an agreement is interpreted as "p_i agrees to let c_j receive a copy of all its messages tagged with any $t \in T_{ij}$ during W, and cj agrees to pay a corresponding data exchange fee". The broker manages all the subscriptions and is responsible for reliably delivering to c_j a copy of each message that has a topic that c_j subscribes to. Note that in the standard pub/sub model, publishers and subscribers are unaware of one another, and their interaction is entirely mediated by the broker. However, it is easy to extend the model by assuming that the broker will only deliver messages from p_i to c_j if c_j has an active agreement (i.e., relative to W) with p_i.

Contract enforcement and settlement involve calculating the total price associated with the messages that have been routed from each p_i to each c_j within each W. A variety of pricing models have recently been proposed for digital assets in the emerging data marketplace scenario [80], [81], [82], [83]. We are going to assume a simple model where each individual message has a constant unit value $val(t_k)$ that is determined solely by the message's topic t_k. We have assumed that the prices of all messages routed from p_i to c_j are determined only by the number of messages and the unit cost for each topic. To do this, we must keep a count of the number of messages about topic t_k that originated from p_i and reached c_j during W, grouped by p_i, c_j, and t_k . We denote each of these counts as $N_{ijk}(W)$.

Generating these counts requires the broker to be capable of metering all traffic; that is, of logging all messages as shown in Figure 3.2. The log consists of a set of tuples: $\{ \langle p_i, c_j, t_k \rangle \}$

At the end of each W, the log is aggregated over each $p_i \in P, c_j \in C, t_k \in T$,resulting in a set of tuples that we call a traffic cube:

Fig. 3.2 Centralised trusted broker with metering traffic capability



$$cube(W) = \{\langle p_i, c_j, t_k, N_{ijk}(W)\rangle\} p_i \in P, c_j \in C, t_k \in T$$

We borrow the use of *cube* terminology from a standard database where it is a table with N attributes, in which the first $N-1$ attributes are dimensions in a database schema (in our case, these are the Producers, Consumers (the VAS), and Topics) and the last is an aggregation of the values in the database for each combination of the dimensions, which is Message count in our case.

We use a matrix-indexing notation to refer to specific cells in the cube, i.e:

$$cube(W)[p_i, c_j, t_k] = N_{ijk}(W)$$

These cubes contain summaries of all data flows observed by a broker. Notice that they only contain metadata, i.e., the counts, but not the content of the messages. Note that the values in the cube may be sparse, i.e., $N_{ijk}(W) = 0$ whenever c_j does not subscribe to t_k.

Settlement is the process of calculating the total fee owed by each c_j to each pi at the end of each W. This is computed by suitably aggregating the counts in the cube, namely:

$$fee(c_j, p_i, W) = \sum_{t_k \in T} N_{ijk}(W) \cdot val(t_k)$$

and the total profit for p_i during W is:

$$profit(p_i, W) = \sum_{c_j} fee(c_j, p_i, W)$$

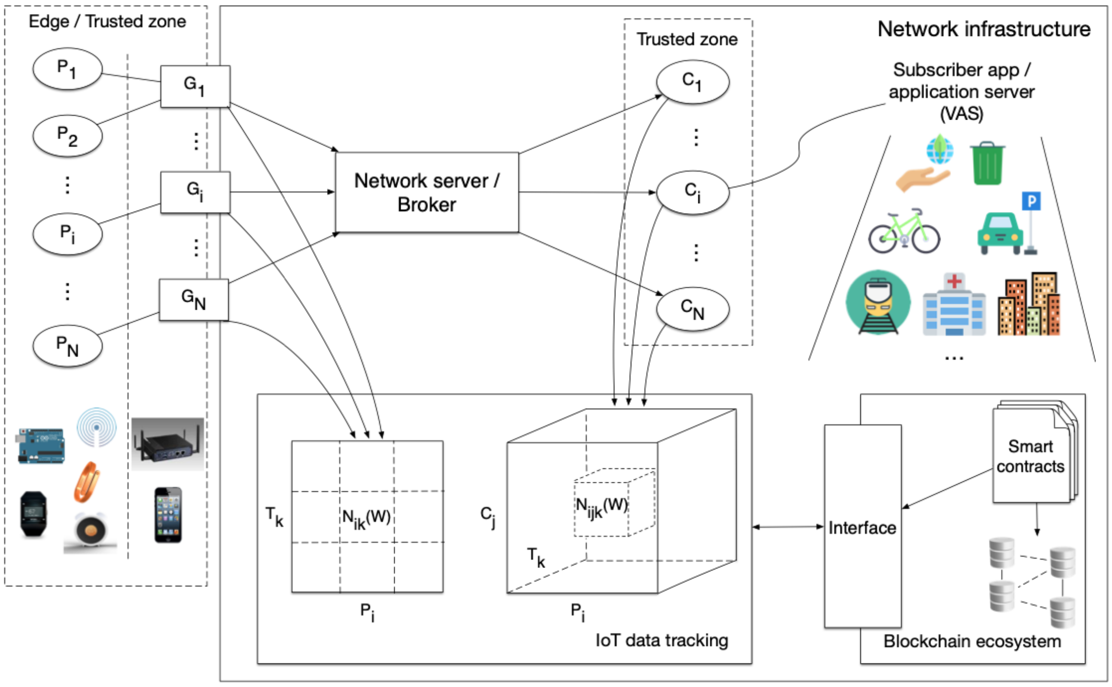
In this centralised scenario with the assumption of broker trust, we have considered so far:

1. The intermediary brokers are entrusted with making the final decision and the approved count for sent and delivered logs.
2. Settlement is straightforward, as the broker is entrusted with generating accurate logging and thus complete and correct cubes. Note that, under the same trust assumptions, settlement extends easily to a more realistic scenario where multiple brokers are deployed, each enhanced with the same logging capabilities and local traffic reporting service.
3. No reputation assessments of the marketplace participants, producers or consumers, where the broker is the central authority that manages the marketplace trades.
4. However, settlement and participants' reputation become challenging in an extended model where there is no assumption of trust in the broker.

3.3.2 Decentralised Trusted Parties

Extending the above model in a decentralised architecture means that there is no assumption of central trust in the broker, and rather it may be distributed into parties and could be transparently self-enforced. Trading in such a marketplace based on message counts is vulnerable to malicious behaviour. Specifically, producers have an incentive to claim to have produced more messages than they have in reality, while conversely, consumers (the VAS) have an incentive to under-report the number of messages they receive. When we remove the assumption that the brokers are trusted, we must also accept that the brokers may collude with any of the participants, and thus deliver traffic cubes that may not be correct or complete. Discovering such collusions may not be possible when the broker is the only source of traffic counts available to the settlement service. At the same time, resolving any disputes amongst pairs of participants requires a public and irrefutable record of the reported traffic. To address these problems, we rely on two overarching principles: (1) the personal responsibility of each participant in the trading, who must report their own counts of messages sent (publishers) or received (subscribers) using trusted zones (see Figure 1 and description below); and (2) transparency, whereby these reports are posted as part of immutable and verifiable blockchain transactions. These principles translate into a two-step approach.

Fig. 3.3 Decentralised Trusted Parties Model with trusted zones [2]



Firstly, we remove the assumption that traffic cubes are generated by the broker alone, and instead enable network elements close to the publishers and subscribers, i.e., gateways and VAS respectively, to generate the cubes. This is shown in Figure 3.3. Secondly, we adopt emerging consensus-based distributed transaction ledgers, specifically blockchain and smart contract technologies, to realise the settlement service. Smart contracts extend the standard blockchain transaction model by adding the capability to execute arbitrary code, which operates on data structures contained in the transaction itself. In this case, a blockchain transaction that is initiated at the end of each window W may operate on the collection of traffic cubes that participants make available at the end of W. This approach provides transparency and accountability, because the content of the blockchain is public and can be inspected, and at the same time a way to address disputes, because for each W, multiple (partial) views of each cube are made available to the settlement service.

Traffic cubes that are generated by the broker summarise the entire traffic during W. In contrast, traffic summaries generated by trading participants reflect the local views of each participant in the data exchanges. These are therefore necessarily partial and incomplete, as each participant, unlike the broker, has no visibility of the end-to-end data flows. We denote these as unilateral traffic cubes, defined as follows. Let us assume that a producer does not know which VASs subscribe to its stream, while subscribers know the source of the messages they receive.

Let $sub(t_K) \subseteq C$ denote the set of subscribers to t_k. A publisher's cube $cube^p$ is a slice of a complete traffic cube, for a specific producer p_i and without the consumer dimension:

$$cube^p(W, p_i) = \left\{\left\langle t_k, N^s_{ik}(W)\right\rangle\right\}_{t_k} \in T$$

where $N^s_{ik}(W)$ is the count of messages with topic t_k sent by p_i during W. Note that p_i can compute $N^s_{ik}(W)$ from its own data flow log, but not $N_{ijk}(W)$.

As subscribers know the source of the messages they receive, we may assume that a subscriber will produce summary reports that include the publisher dimension, but which only contain the tuples that pertain to a single c_j . Thus, a subscriber's cube is defined as:

$$cube^c(W, c_j) = \left\{\left\langle p_i, t_k, N_{ijk}(W) | c_j \in sub(t_k)\right\}_{p_i \in P, t_k \in T}$$

Figure 3.3 concretely illustrates this setting. To remove the need for centralised trust, we push it towards the borders of the data flow network by defining two *trusted zones*. The first trusted zone includes all the elements at the edge of the network infrastructure, such as the IoT devices P and the gateways G_i, whereas the second one includes C. IoT data are still routed towards the VASs through brokers – using a publish-subscribe pattern – or network servers. However, we now assume that a new, independent IoT data-tracking component receives the unilateral cubes from gateways and VASs. Finally, a smart contract, decentralised trusted service deployed on a blockchain, periodically accesses the traffic cubes to realise settlement services and resolve possible conflicts.

Suppose that, at the end of W, every p_i and c_j produces unilateral cubes relative to W. These form the set:

$$\left\{cube^p(W, p_i)\right\}_{p_i \in P} \cup \left\{cube^c(W, c_j)\right\}_{c_j \in C} \tag{3.1}$$

As each of these cubes provides a partial view of the same complete cube cube(W) that would have been generated centrally by a broker, we expect that the values found in these cubes be somehow consistent with cube(W). The pub/sub model implies that the number of messages sent by p_i with topic t_k during W must be equal (assuming no messages are lost and ignoring duplicate transmissions, as in MQTT QoS level 3) to the number of messages each c_j that subscribes to t_k receives from p_i.

We can capture this constraint formally using our cube notation, as follows. For each $p_i \in P, t_k \in T, c_j \in sub(t_k)$:

$$\begin{aligned}&cube^p(W,p_i)[t_k] = N^s_{ik}(W) = \\ &cube(W)[p_i,t_k,c_j] = N_{ijk}(W) = \\ &cube^s(W,c_j)[p_i,t_k]\end{aligned} \tag{3.2}$$

We say that the set (3.1) of all unilateral cubes is consistent at W, if and only their contents satisfy constraint (3.2). We use this definition as a basis for the settlement of message exchanges within each W , in the general case that the broker cannot be trusted to provide a single global cube that is complete and correct. Specifically, in our architecture we now assume that a new, independent component receives all cubes in (3.1) at the end of each W, and checks their consistency using (3.2). In the next section we discuss a practical implementation of this idea, where this new component is realised as an Ethereum Smart Contract and unilateral cubes are posted publicly as part of blockchain transactions. In this decentralised scenario, such a settlement service must be able to deal with two interdependent issues, namely (a) completeness and (b) consistency of the set (3.1) of all cubes. The case when set (3.1) is both complete and consistent is straightforward and results in successful settlement, as all information for settlement is available, and there are no disagreements.

When the set of cubes is incomplete, we may try to use (3.2) to propagate the missing values from the more complete to the less complete cubes. More precisely, suppose $cube^p(W,p_i)$ is missing for a p_i. If $N_{ijk}(W) = cubes(W,c_j)\left[p_i,t_k\right]$ is available for some t_k and some $c_j \in sub(t_k)$, then we set $cube^p(W,p_i)[t_k] = N_{ijk}(W)$. In practice, this can be viewed as "taking c_js word for p_is missing report".

Symmetrically, the settlement service may use the available $cube^p(W,p_i)$,in combination with subscription information $\{sub(t_k)|t_k \in T\}$, to fill in missing values in $cube^s(W,c_j)$, i.e., by setting $cube^s(W,c_j)[p_i,t_k] = cube^p(W,p_i)[t_k]$ for each t_k and each $c_j \in sub(t_k)$.

Of course, there is no guarantee that all missing values can be propagated. In this case, settlement for the (p_i,c_j) pairs corresponding to the missing cube entries is simply not possible.

The final, and perhaps most important case occurs when constraint 3.2 is violated for some combination of $\langle\ p_i,c_j,t_k\ \rangle$. This may be due to the malicious cases of over-reporting producers, or under-reporting subscribers. Either of these scenarios manifests itself as inequalities in (3.2), of the form:

$$cube^p(W,p_i)\left[t_k\right] > cube^s(W,c_j)\left[p_i,t_k\right]$$

In this situation, we are able to detect the inconsistency, but we may not have enough information to determine whether p_i, c_j , or both are guilty of fraud. This initial work presents initial ideas on promoting a self-regulating exchange infrastructure in the presence of such unresolvable inconsistencies. We present an initial implementation and evaluation of this work in the next section where the settlement service simply reports the detected inequalities.

3.4 Implementation and Evaluation

3.4.1 Implementation

For the purpose of experimentation and evaluation, we have adapted the open-source Mosquitto MQTT broker to support message logging and cube generation in a Cassandra NoSQL database. We refer to it as the *TrackerDB*. We connected to the MQTT broker real producers using channels provided by the ThingSpeak platform [84]. Using the TrackerDB, we are able to simulate the generation of unilateral cubes that can be either complete and correct, or reflect malicious behaviour, for evaluation purposes. The TrackerDB can be queried by any third party client through a REST service interface. Smart Contracts interact with the service through an Ethereum-specific mechanism, described below. In reality, unilateral cubes would be generated by gateways on the producer's side as well as by VASs within their trusted zones. This does not affect the properties of the cubes' compliance and settlement, because liability is pushed at the edge.

We now focus on the use of Smart Contracts in this setting. We developed them using Solidity, the Ethereum's scripting language. To implement the contracts, we assigned an Ethereum account to each producer and VAS. We connected these accounts to our private Ethereum test network, deployed on a single node with 6-core Intel Xeon E5-2640 and 16GB of RAM. We wrote, deployed and evaluated Smart Contracts in the network by using the Ethereum web-browser-based IDE Remix, connected to our private chain through a Remote Procedure Call (RPC) protocol. In our implementation, accounts prepare and send the transactions to the blockchain to instances of Geth [85] through RPC. To measure gas consumption, we used the debug tool provided by Remix and we observed the difference in the account balance before and after invoking a settlement contract.

A limitation of Ethereum smart contracts is that they cannot directly access off-chain data about real-world state and events. In our case this represents a challenge in acquiring unilateral cube values. More precisely, Smart Contracts are independently executed by any node in the chain, thus, each execution needs to retrieve such information from an off-chain source independently, without any assurance of the information integrity. To overcome this limit, the concept of *oracle* has been introduced. Simply speaking, an oracle is a special contract that serves data requests from traditional contracts, by sourcing them from designated data feeds. Two options are possible for implementing oracles. The first one is relying on existing proxy services. Oraclize [86] provides a *programmable* oracle that can interact with any data source selected among a pre-defined set of standard channels. In addition Oraclize provides proof of authenticity by means of a TLSNotary proof that guarantees the authenticity and integrity of the retrieved data. These functionalities come at a cost. For each off-chain query, Oraclize requires a fee that includes a commission, ranging from 0.01$ to 0.04$, and a refund of the gas used to perform the transaction. The other option is when each party of the contract, producer and VAS, independently updates their view of unilateral cubes by pulling their values from cube generators located within their trusted zones and then creating a transaction that embeds the cubes in the blockchain. This way, any node executing the smart contract will have the same copy of that cube. As a result, any costs associated with the use of an external oracle proxy, such as Oraclize, can be saved. Since in our model the responsibility and liability of producing faulty cubes is placed with producers and VASs, this option well suffices.

Pseudocode 1 shows the pseudocode of our settlement contract. For the sake of simplicity, this code snippet only accounts for the single producer and the single VAS scenario, although generalisation is straightforward. The contract first requests that the involved parties to provide their unilateral cubes; then it uses this information to perform the actual settlement, by combining the two unilateral cubes. If the processed combined cube is consistent then a payment to the producer is performed; otherwise, a dispute resolution mechanism should be invoked[1]. When a dispute resolution is invoked, payments are retained from being performed due to the impossibility of clearly identifying the correct unilateral cube. A reputation mechanism can be implemented in order to penalise both parties involved in a given settlement transaction and to promote them when a honest behaviour is identified. As it is not expected that the reputation computation will require off-chain interactions [87, 71], we are confident that not considering its implementation at this phase will not significantly affect the overall contract execution cost.

Table 3.1 shows the execution cost of cube settlement operations expressed in gas without and with Oraclize, respectively. The most expensive operation to be performed is the *contract deployment*, consuming from $175,000$ gas without *Oraclize* to $2,061,490$ gas with Oraclize. The difference between these values is due to the higher number of functionalities implemented within Oraclize's API, which the contract has to deploy[2]. Both the *update* and

[1]At this time, our implementation simply reports and logs the detected inequalities.

[2]It is worth noticing that most of such functionalities are not required in a distributed liability model such as the one promoted in our architecture

Algorithm 1: Cube settlement contract

```
if sender $\neq$ authorizedAddress then
    throw
    if queryId = producerQuery then
        producer $\leftarrow$ unilateralCube
        vasQuery $\leftarrow$ update()
    else if queryId = vasQuery then
        vas $\leftarrow$ unilateralCube
        if producer = vas then
            transfer(producerAccount, dataPrice, cube)
        else
            disputeResolution()
        end if
    end if
end if
```

Table 3.1 Execution cost of cube settlement contract operations.

	Gas used	
Operation	w/o Oraclize	w Oraclize
Contract deployment	175000	2061490
Update	41000	120000
Callback	23000	70000
Transfer	21000	21000

callback operations have a higher cost due to the Oraclize fee, whereas the *transfer* operation has the same cost.

3.4.2 Evaluation

The aim of this section is to quantify the cost of the smart contract described above and the associated cube settlement operations. By considering the scenario in which one VAS consumes the data of one producer, we evaluate how the cost of performing such a contract affects the data price when the amount of data exchanged and the settlement transactions need changes. The quantity of exchanged data varies according to the different purposes of the exchange (event-based data rather than real-time series acquisition). Nevertheless, the reason for considering a variation in the number of required settlements needs some clarification. The most natural strategy will be to perform the settlement at the end of each contractually agreed data exchange, however, in the early stage of a hypothetical marketplace where new producers and VAS join without necessarily trusting each other or having an already established reputation, two situations might occur:

- Producers and VAS have a low reputation, hence, their trust levels are low and the risk of claiming wrong unilateral cubes is high. Performing more than one cube settlement, in an initial rump-up phase of a given data exchange, allows them to mutually increase their reputation and trust.
- Producers and VAS have a high reputation, hence, they are expected to act honestly. Cube settlements may occur at a lower rate, only at the end of a data exchange phase, because the risk of producing faulty cubes is mitigated.

By evaluating the cost of performing the settlement operations, we are able to define the minimum price that VASs should pay for each unit of consumed data in order to sustain the settlement infrastructure and eventually generate profit for the producers. We define the minimum data price as the amount of Ether needed to at least cover the cost of the contract deployment and transactions needed to perform cube settlement operations. This means that if a producer sells data at the minimum price, its profit will be zero. At the time of writing, one Ether costs 220 $, however, its price is still very volatile.(http://etherscan.io/chart/etherprice).

As a result, transaction costs may frequently vary, thus leading to uncertainty about the economic feasibility of a specific application. We analysed the capability of Ethereum to support a stable transaction cost by tuning the gas price. The main drawback when setting a low gas price is the increase in the time required before a transaction is validated. Assuming a range of gas prices between 0.9 Gwei and 20 Gwei (9e-10 and 2e-8 Ether respectively), as the minimum and average reported by the Ethereum network in 2017, the time required for a transaction to be validated in the chain varies from 2 minutes to 14 seconds (etherscan.io/chart). As explained before, even in the case of multiple settlements, we do not expect that meaningful data exchange will last less than 2 minutes, thus we consider it a viable choice to select the current minimum gas price.

Figure 3.4a shows a general overview of the minimum data price by varying the frequency of cube clearance operations and the amount of transferred data. The price is directly proportional to the number of cube settlements performed while being inversely proportional the amount of data exchanged. Clearly, the more the data a VAS purchases, the lower the impact of the cost of performing cube settlement. Depending on the type of data exchanged and the trustworthiness of the involved parties, this figure clearly shows how an optimal settlement strategy can always be found to dynamically adapt to the number of settlement operations.

Figures 3.4b and 3.4c show the total cost of performing 1 or 5 cube settlement operations for a fixed amount of transferred data, when considering a gas price ranging from 0.9 Gwei to 20 Gwei. More specifically, Figure 3.4b shows the case when each party of the contract use its oracle implementation, while Figure 3.4c shows the case when Oraclize functionalities are used. It is worth noticing the large costs increase (on average 4 times more), due to the commission and refund of the gas used to perform the transaction to be paid to Oraclize. Without Oraclize, the cost of a single cube settlement transaction ranges from 9.9e-5 ($ 2.18e-2) Ether to 2.2e-3 Ether ($ 4.84e-1) when the gas price selected is 0.9 Gwei and 20 Gwei, respectively. The more data are transferred, the less impact the transaction cost has per single unit of data. In fact, when performing a cube settlement operation spread over 2000 units of data, its cost ranges from 1.26e-7 Ether ($ 2.77e-5) to 2.8e-6 Ether ($ 6.16e-4). Alternatively, when performing 5 cube settlements over 2000 unit of data, their cost ranges from 3.15e-7 Ether ($ 6.93e-5) to 7e-6 Ether ($ 1.54e-3). When using *Oraclize*, the cost of a single cube settlement transaction ranges from 3.61e-4 Ether ($ 7.94e-2) to 8.02e-3 Ether ($ 1.76) when the Gas price selected is 0.9 Gwei and 20 Gwei, respectively. When performing a cube settlement operation spread over 2000 uint of data, its cost ranges from 1.11e-6 Ether ($ 2.44e-4) to 2.46e-5 Ether ($ 5.42e-3). Alternatively, when performing 5 cube settlement over the same amount of data, their cost ranges from 1.83e-6 Ether ($ 4.03e-4) to 4.07e-5 Ether ($ 8.95e-3).

Table 3.2 Estimated data price for different use cases.

	Data price			
	w/o Oraclize		w Oraclize	
Data rate	ETH	USD	ETH	USD
high	5.73e-8	1.26e-5	2.09e-7	4.59e-5
medium	3.44e-6	7.56e-4	1.25e-5	2.76e-3
low	2.06e-4	4.54e-2	7.52e-4	1.65e-1

In order to derive a profitable data price, we can assume that the cost of performing settlement operations has to be equal to 2% of the price for that data amount and that only 1 cube settlement is performed per day. We consider two examples: (1) an air quality monitoring application, with a low data rate, running on a low-power wide-area network (LPWAN), such as LoRaWAN, that samples and transmits data every hour, resulting in 24 measurements per day; (2) a heart rate monitoring application (e.g., Fitbit), with sampling frequency of 1 second and 1 minute corresponding to a high and medium data rate, respectively. Table 3.2 shows that data price ranges from $5.73e-8$ Ether ($\$1.26e-5$) to $7.52e-4$ Ether ($0.165) depending on the data transfer rate and the type of data feed selected.

3.5 Discussion and Conclusion

The analysis above helped us to identify the feasibility of building a decentralised open and transparent accounting infrastructure, useful for creating a fair data marketplace, where the data price can evolve depending on data quality, demand and offer. To minimise the shared costs of running such an infrastructure, we observed how the gas price can be tweaked at the cost of a lower transaction rate, leveraging the lack of real-time requirements for the settlement operations. Moreover, we demonstrated how the cube architecture allows for scalability by reducing the settlement transaction frequency. Nevertheless, we recognise that the estimated infrastructure costs are related to the current inflation in the Ether value, due to the large number of currently deployed general purpose smart contracts (raising the Ether price by over 20 times in just one year). While we plan to perform similar analyses using different blockchain implementations like hyperledger [88], we anticipate that a decentralised trading infrastructure will require a new dedicated Ethereum network, dedicated to contract settlement, with lower incentive fees for the miners. While keeping it open, we are confident that, due to the large amount of IoT data exchanges, such a market will provide a viable business opportunity for miners even at lower transaction and incentive fees.

This chapter showed the many requirements that should be applied in our model of a brokered IoT data marketplace. We proposed an initial technical infrastructure for a new kind of data marketplace that, in the long run, is designed to meet these requirements. We developed initial models for marketplaces in different network architectures, applying these requirements and using trust as a parameter.

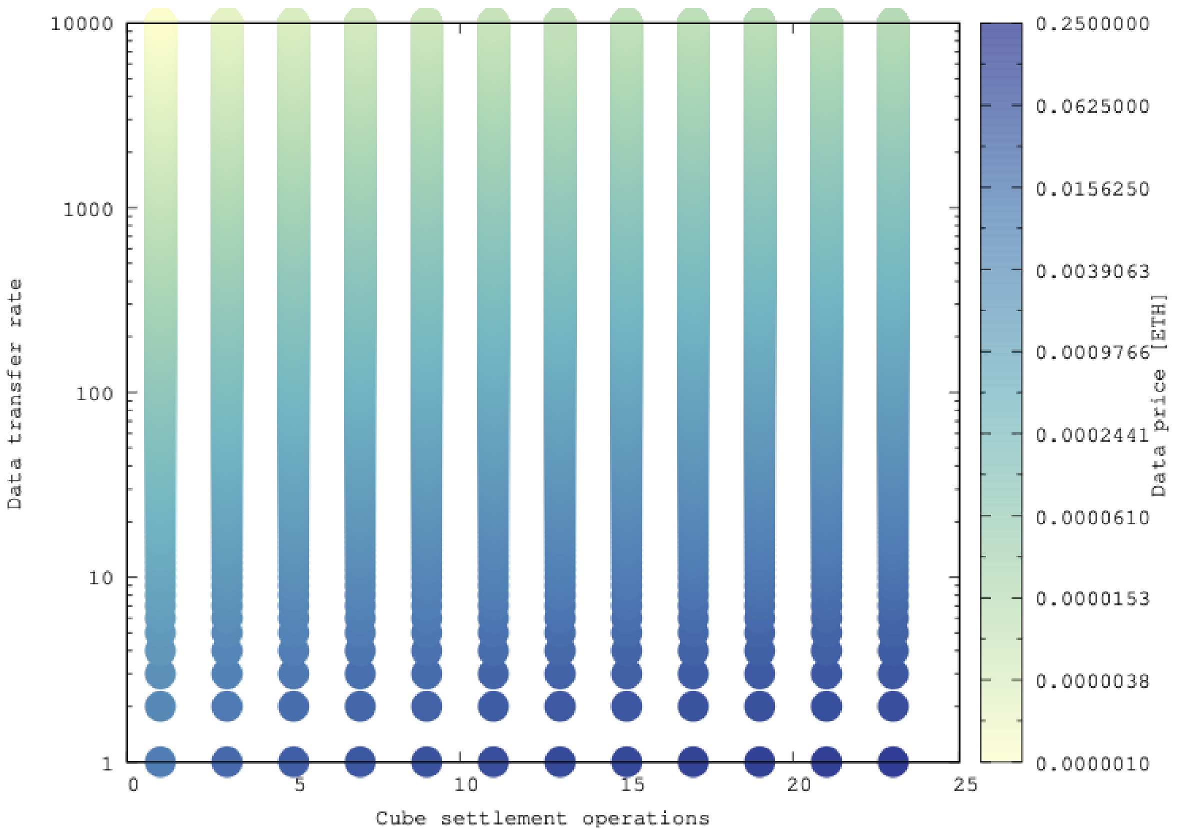
In a centralised model, we adapt the MQTT broker by adding traffic metering capabilities to generate cubes of data exchange between parties, and we use the Ethereum smart contracts technology for enforcing contract definition and triggering dispute resolution. In this model, settlement is straightforward as the broker is trusted with generating accurate cubes for data exchange. However, settlement and participant reputation become challenging in an extended model with the absence of trust in the broker.

In a decentralised architecture, the trust parameter has to be distributed among the marketplace participants with personal responsibilities and self-enforced transparency to report local data cubes. This model is vulnerable to malicious behaviour with under-reporting from the consumer side or over-reporting from the producer side. We evaluated the settlement service that simply reports the detected inconsistency in participants' local cubes.

This model encourages us to further develop the idea to remove trust between the parties and instead leverage the model with blockchain. We recognise that the cube settlement component is very important but it is still only one building block of such an infrastructure. In the next chapters, we present our model of a trustless decentralised marketplace that trusts the blockchain and removes the trust from the marketplace participants. Moreover, it shows the missing elements that contribute that quantifying participants' trust and handling dispute cases that appear as a result of independent reports.

Fig. 3.4 Cost of performing cube settlement operations for different data transfer rates.

(a) Minimum data price



(b) Cost without Oraclize

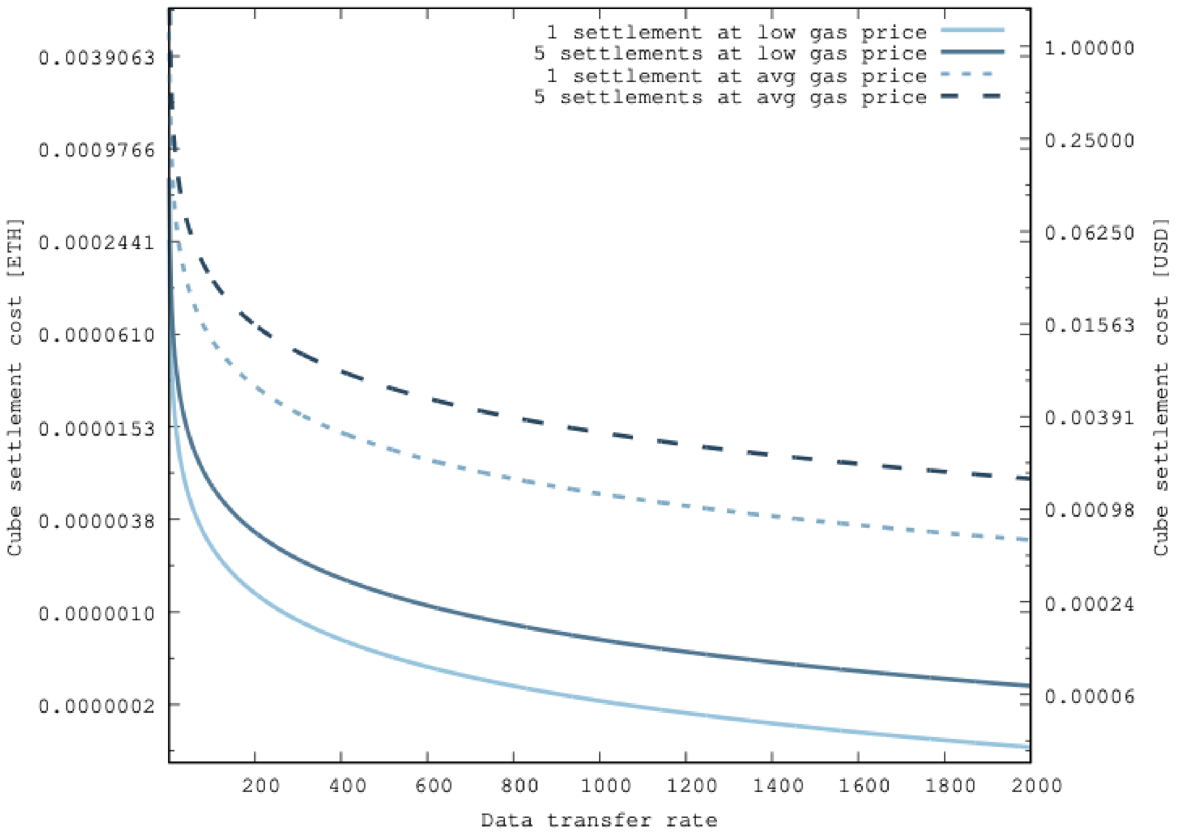
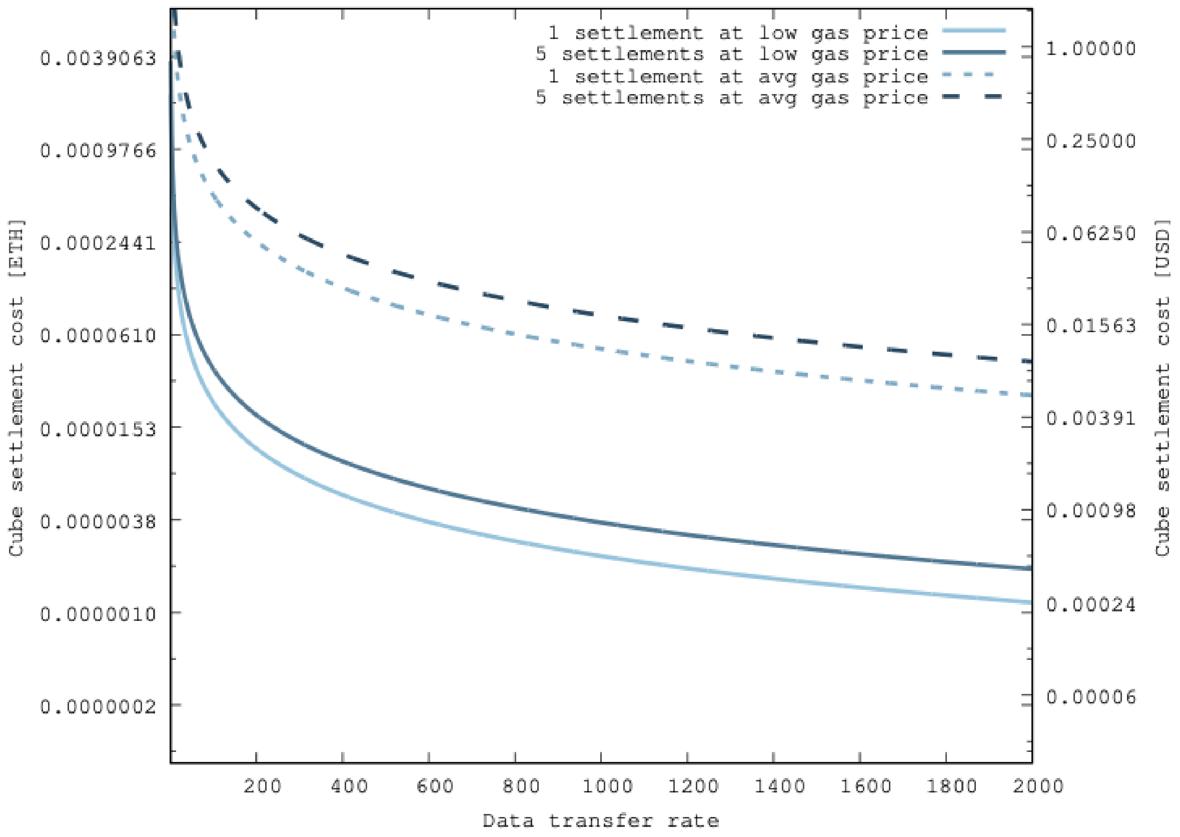


Fig. 3.4 Cost of performing cube settlement operations for different data transfer rates.

(c) Cost with Oraclize



Chapter 4

Decentralised Marketplace

Most of this chapter's materials have been published in the paper:

"Toward a Decentralized, Trustless Marketplace for Brokered IoT Data Trading Using Blockchain". S. Bajoudah; C. Dong and P. Missier. 2019 IEEE International Conference on Blockchain (Blockchain), Atlanta, USA.

4.1 Introduction

The emergence of blockchain technology has facilitated the new prospect of developing decentralised IoT data marketplaces on the blockchain. It represents a new business model where data in various forms can be traded in a data marketplace as traditional goods [8].

Unlike static data, IoT data streams tend to lose their value if they are not consumed in near-real time, and data transmission and delivery may be unreliable.

On the other hand, brokered IoT data architecture marketplaces based on message brokers systems such as MQTT [14] allow a stream owner to resell their streams in real-time multiple times to multiple parties.

While the IoT network and message-passing infrastructure can support a scalable marketplace, this inevitably leads to issues over the mutual trust between participants, especially if this marketplace does not provide a prior reputation measurement for parties in the marketplace. In addition, the short-lived nature of streams requires efficient, automated mechanisms to create a legal trade agreement, enforce its obligations and contracts and result in payment settlement.

In Chapter 3 we showed an initial technical infrastructure for IoT data that are exchanged in Mosquitto MQTT broker [79] with adaptation of traffic metering capabilities. We extended the model into a decentralised architecture and distributed the trust among parties. We showed that trading in such a marketplace is based on message counts and that this is vulnerable to malicious behaviour.

In this chapter, we investigate the capability of the blockchain technology to provide a trustless marketplace for trading IoT data as its main assets. We study the ability of the second generation of the blockchain Ethereum – smart contract – in automating and monitoring the trade affairs of streaming IoT real-time data without storing the IoT data in the blockchain. we envision a trustless decentralised marketplace for real-time IoT data with blockchain trust, and a reputation system that is capable of quantifying the participants' reputation in the marketplace.

Our approach involves using Ethereum smart contracts to support each phase of the interaction between a data provider and a consumer. It separates the data exchange interaction, which occurs on the IoT network and core cloud network, from transaction-based interactions aimed at enforcing the non-repudiability of participants' actions and resolving their disputes on the blockchain network.

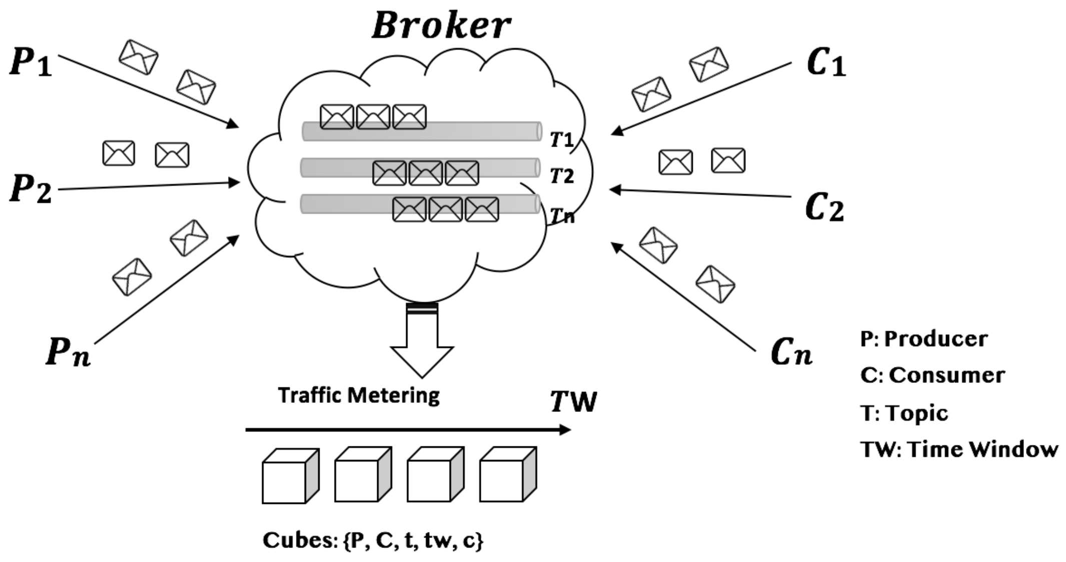
This approach follows on from our earlier proposal for an IoT data marketplace in Chapter 3, where we suggested that Ethereum is capable of supporting a fully decentralised marketplace without any assumption of mutual trust.

Both trade parties in the proposed approach in Chapter 3 are required to report to a smart contract how much data have been sent and received and then the smart contract settles any disputes.

Compared with the current proposal, we are relying in this approach on the notion of periodic checkpoints during data exchange, supported by blockchain transactions to ensure limited scope for fraud on either side.

In our decentralised model, the settlement and the trade monitoring are done on the blockchain layer, which means that we need a level of trust to accomplish these trades successfully. The blockchain provides the opportunity to remove the third party that guarantees the trust in the centralised model, and instead to distribute the trust and place it elsewhere, namely in public key cryptography and a "consensus mechanism" that allows us to determine the truth.

Fig. 4.1 Centralized Brokered IoT Data Marketplace Architecture



4.2 Decentralised Marketplace Model

4.2.1 Brokered IoT Data Exchange

We assume, following standard IoT data streaming practices, that the exchange of streaming data between data producer P and data consumer C is mediated by some transaction-agnostic broker infrastructure, such as the one shown in Figure 4.1.

In this data transfer model, the stream is broken down into discrete message batches. Providers tag their messages with *topics* that uniquely identify that provider's stream. A consumer is allowed to subscribe to a topic subject to the conditions set in a Trade Agreement, as described below.

In our previous work in Chapter 3 we initially assumed a network architecture where the broker is a trusted component that can be relied upon to generate truthful data exchange reports, which in turn can be used to settle disputes between producers and consumers (the "cubes" in Figure 4.1). In such a scenario, the smart contract is simply in charge of settlement, according to the reports. In the decentralised trusted parties model shown in the same chapter, the generating of reports is left to each participant. In this case, the smart contract has a difficult task because the reports themselves cannot be trusted, and disputes cannot be settled by ascribing certain responsibilities to either participant.

4.2.2 Model Elements

In this chapter we work around these difficulties, as we do not require the broker or the participants to generate any report at all. Instead, the broker is simply a network element. The goal of the marketplace is twofold.

The first goal is to enable the trading of streaming data through the broker while offering guarantees, i.e., regarding the maximum loss incurred by either the producer or the consumer in case of a failed trade due to a fraudulent behaviour caused by the disputes about the amount of data exchanged. The second target is to resolve this dispute of the data exchanged.

To achieve this, we augment the data exchange with the exchange of *data receipts* between a consumer and a producer, which occurs at regular intervals and throughout the duration of the data stream. Such receipts are exchanged as part of transactions that are mediated by a smart contract, denoted *SC*, on the blockchain. The length of the exchange interval, denoted as Batch Size or *BS*, is set at the time of the trading agreement negotiation. As we will see, this parameter enables a producer to control the level of risk they are prepared to tolerate given limited trust in a consumer.

The model consists of the following elements:

1. The description of data offered by a producer;

2. A trade agreement, which includes details of the data to be exchanged and the exchange protocol, the corresponding market value, and additional parameters such as *BS* mentioned above;

3. A protocol for the exchange of data receipts, which includes both parties in addition to a neutral smart contract;

4. A reputation model, which allows a reputation score to be assigned to every pair producer and consumer of participants at the end of each transaction they are involved in. Participants may use reputation scores to assess the risk of entering into an agreement with an untrusted participant.

The smart contract is responsible for each transaction associated with 1-3, and specifically for recording (i) the specification of the data offering, (ii) the trade agreement, and (iii) each data receipt. The last model element represents the reputation system that scores participants, producer and consumer (see Chapter 5).

4.2.3 Data Offering

The first function of the smart contract is to let data producers publish their data offerings on the blockchain, where they can be then be discovered by prospective consumers. As mentioned, a data stream consists of a sequence of messages uniquely identified by a provider's topic, and a data offering describes the type of stream and specifies how to subscribe to the stream. Specifically, a data offering $DO = \langle T, TI, MR, UP \rangle$ includes, in addition to the topic T, a specification of (i) the time interval TI during which the offer is valid, (ii) the expected streaming message rate MR, e.g., in messages/time, and (iii) the unit cost UP of each message in the stream.

4.2.4 Trade Agreement

The trade agreement is a legally binding contract (we use the term "agreement" to avoid confusion with smart contracts) between producer and consumer, which defines the terms of the data exchange. An agreement comes into force when (i) it is signed by both parties using their blockchain account keys (Ethereum in our implementation), and (ii) a smart contract transaction containing the agreement is committed to the blockchain, at which point it can no longer be amended. The agreement contains (i) a specific data offering DO and (ii) a time interval $TATI$, contained within the time interval TI, during which the agreement is in force. For instance, a consumer may want to subscribe to a portion of an event that is offered over a long period of time. We denote the total price as $TP = UP \cdot TATI$ and the *estimated* total number of messages in the agreement as $ETM = MR \cdot TATI$. The latter is an estimate, rather than a set value, because the total number of messages that can be sent within interval $TATI$ is affected by the time required to carry out the Data Receipt protocol, as explained next.

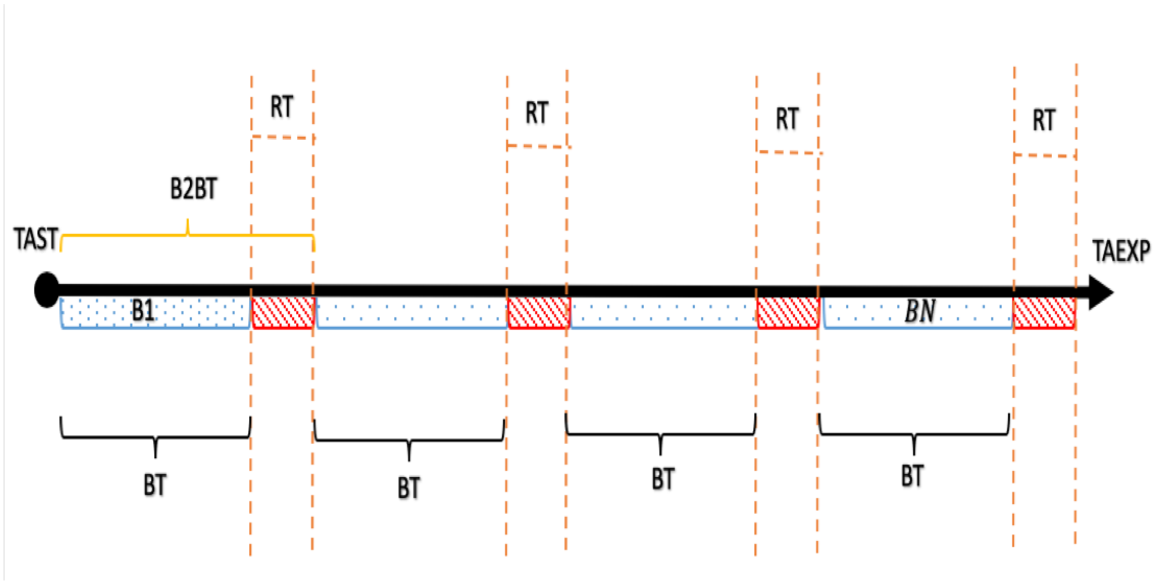
4.2.5 Data Receipt Protocol

Once the trade agreement is in force, a consumer is allowed to subscribe to a producer's stream. Under normal circumstances and when both parties comply with the agreement, and data transfer takes place as expected, at the end of the $TATI$ interval a consumer informs SC that the agreement has been fulfilled, and SC proceeds to settle the payment as per the agreement. Suppose, however, that a consumer fails to inform SC. This may happen because the consumer actually failed to receive some of the data in the stream, or because it fraudulently *claims* not to have received the data. In our model we assume that SC is unable to distinguish between these two events, because there is no requirement for the data broker to keep a (verifiably truthful) log of its message delivery. In this situation, the only possible course of action for *SC* is to believe consumer's claim, and to withhold producer's payment as a consequence. Thus, assuming minimal accountability on the part of the broker and no trust amongst the participants, producer may become the victim of consumer's fraud.

Our approach to mitigate this circumstance is to introduce *checkpoints* throughout the duration of data delivery. The number of messages between two checkpoints is the batch size *BS*, which a producer can configure as part of the agreement negotiation with a consumer. At each checkpoint, a consumer is expected to send a *data receipt* to *SC* as part of a blockchain transaction, which acknowledges receipt of one batch of data from a producer. When the transaction is confirmed, *SC* records the receipt and then informs the producer.

Meanwhile, at the end of each batch the producer will have suspended its streaming to the consumer until it receives the acknowledgement from *SC*. If the producer does not receive a message within a certain time limit called the Receipt Processing Time *RT*, it times out and terminates the trade agreement in order to cut its losses (in practice, the consumer's subscription to the stream is cancelled). Thus, the data exchange protocol and data receipt protocols are interconnected as shown in Figure 4.2.

Fig. 4.2 Data receipt protocol Interactions between the producer, the consumer and *SC*



RT is determined based on the unit gas price *GUP* that a consumer is willing to pay in the transaction of the receipt. Because *GUP* largely determines the duration of the transactions in the blockchain to be confirmed by miners, the consumer has the option to increase the *GUP* in order to process their transaction faster and therefore he will have more time out of the total *TATI*, to receive more batches. This is because, in Ethereum (used for our implementation) cost and POW model, higher *GUP* means that this transaction has higher priority for being validated by miners.

The strategy followed is usually picking the transactions with higher *GUP* to be included in the next block. Thus, a higher offered gas price *GUP* leads to quicker processing of the transaction and can therefore handle more messages ATM.

The minimum and the maximum *GUP* in the network can be found using the *ETH Gas Station* [89]. This is a tool to understand the conditions of the current gas market and current policies of network miners. It shows the maximum and the median time taken by miners to confirm the transaction for each *GUP*. While the transaction confirmation in most cases cannot exceed the maximum time estimated for each *GUP*, we have set RT_{max} to equal the maximum time of the confirmation. We expect the consumer to trigger the receipt transaction when it is verified and published in the network within this time. However, longer *RT* intervals translate into fewer effective messages delivered to the consumer, as in our model the latency *RT* counts as part of the total agreement interval, *TATI*, as explained next.

RT_{max} is a function of *GUP* where it is configurable parameter based on the gas market and current policies of network miners. It reflects the maximum expected time required for a receipt transaction to be confirmed on the blockchain.

$$RT_{max} = f(GUP)$$

4.2.6 Cost/risk/time trade-offs

In the model just illustrated, the producer and the consumer agree on a total duration for the data streaming, *TATI*, and a streaming rate, *MR*. We have also assumed that the producer has a way to assess the risk of *data loss* when the consumer is not trustworthy, i.e., by accessing the consumer's reputation score. In this setting, the term data loss refers to the number of messages that the producer will have sent to the consumer, that the consumer will not acknowledge and therefore will not pay for.

In order to minimise its risk, the producer is motivated to choose frequent checkpoints, that is, by setting *BS* to a small value. This, however, has a cost impact, because checkpoints are smart contract transactions and as such, in blockchain models like Ethereum, each of them incurs a fee. There is therefore a trade-off between the risk of losing data and the cost of engaging in a long-running trade with many checkpoints along the way.

Now suppose that (i) the transaction fees for data receipts are charged to the consumer, and (ii) the latency *RT* due to each data receipt transaction is detracted from the total contract time, $TATI$.

These conditions potentially create a tension between the producer and the consumer, as the producer is interested in low risk, while the consumer is interested in low transaction costs and a minimal reduction in effective contract time. Such tension is embodied by the consumer's reputation score, $Rep^C \in [0,100]$. The ideal win-win scenario occurs when the consumer is fully trusted, that is, $Rep^C = 100$, as in this case there is no need for checkpoints, i.e., $BS = TATI \cdot MR$. When $Rep^C < 100$, the producer will set $BS < TATI \cdot MR$, and the consumer will experience higher cost and fewer total messages delivered within *TATI*.

Thus, it makes sense to assume that *BS* is a function of $Rep^C : BS = f(Rep^C)$, where $f()$ is a parameter in the model and can be chosen to either amplify or reduce the effect of reputation. In our experiments we have used a logarithmic function: $f(Rep^C) = \ln(Rep^C/100+1)/J$.

Choosing a suitable value for the constant *J* in order to help control the number of receipts required. For this, we have defined a family of functions $BS = f(Rep^C) = \ln(Rep^C/100+1)/J$ where parameter *J* is set by constraining the minimum number of receipts when *TATI* is set to one day (24 hours) and $MR = 100$ msgs/s.

The three columns in Table 4.1 show the effect of setting $J = 1.38$, 2.10, and 2.77, the corresponding minimum number of receipts 2,3, and 4, across the range of reputation scores.

Table 4.1 Minimum number of receipts with three different constant values in $f(Rep^C)$

Rep^C	**Number of Receipt**		
	$\ln(Rep^C/100+1)/1.38$	$\ln(Rep^C/100+1)/2.10$	$\ln(Rep^C/100+1)/2.77$
10	15	21	29
20	8	11	16
30	6	7	11
40	5	6	9
50	4	5	7
60	3	4	6
70	3	3	6
80	3	3	5
90	3	3	5
100	**2**	**3**	**4**

Setting $J = 1.38$ produces the minimum number of receipts, 2, for $Rep^C = 100$ and increases to 3 for $50 \leq Rep^C \leq 90$

In contrast, $J = 2.10$ produces the minimum number of receipts within the reputation range $Rep^C \geq 70$, and $J = 2.77$, the minimum occurs for $Rep^C = 100$, while the number of receipts are fairly evenly distributed in the range $60 \leq Rep^C \leq 70$ with 6 receipts and for $80 \leq Rep^C \leq 90$ with 5 receipts.

We settled for $J = 2.77$, as this provides as good segregation of cost relative to reputation while limiting producer loss. An increase in the number of batches received means more receipts and therefore more gas incurred by the consumer. If we assume that the producer has no incentive not to send the data as agreed in the agreement, a trade fails when the consumer fails to send a receipt or was not honest when reporting the exact number of messages received. However, the producer loses less when $J = 2.77$. The data receipt protocol is designed to make the marketplace sustainable by providing incentives to parties to increase their reputation.

It is straightforward to see that, in this model, the producer's maximum data loss is simply one batch of messages. As mentioned, RT_{max} denotes the producer's estimate of RT, which will be used as timeout.

While this setting has no direct effect on data loss, it does affect the consumer's cost and the actual number of messages received. Because of our assumption (ii), a large value for RT_{max} results in fewer messages sent to the consumer. To make this observation precise, consider *TATI*, *MR* and *BS* as constants from the agreement. The time required to send a batch of data is

$$BT = \frac{BS}{MR}$$

and the number of batches sent during *TATI* is

$$BN = \frac{TATI}{BT + RT_{max}}$$

because of assumption (ii) above. Equivalently,

$$BN = \frac{TATI \cdot MR}{BS + RT_{max} \cdot MR}$$

Assuming the Ethereum cost model with gas unit price *GUP* and gas consumption per transaction *GT*, the total cost *RC* due to the receipt transactions is:

$$RC = BN \cdot GT \cdot GUP$$

As expected, the cost is inversely proportional to *BS*.

The actual number of messages *ATM* delivered at the end of *TATI* is

$$ATM = (TATI - BN \cdot RT_{max}) \cdot MR$$

which decreases as *BS* and RT_{max} increase, as expected.

As assumed above (i), cost *RC* is charged to the consumer. The total cost associated with a trade agreement also includes a one-off transaction fees, which is split between the producer and the consumer as follows. Firstly, in order to participate in the marketplace, each participant, in either a consumer or producer role, must register itself with the network. This incurs a one-off *registration cost* to execute the smart contract user registration function. Secondly, the deployment of a trade offer to the network is also implemented as a smart contract function, which again incurs a fee. This is a provider-only cost. Thirdly, a smart contract fee is paid when a new trade agreement is recorded on the blockchain. This cost is split between the producer and the consumer.

4.2.7 End of trade

Marketplace practice suggests that the consumer should pay a deposit at the start of the trade, as a guarantee that sufficient funds are available to settle the agreement at the end of it. The funds are held by the *SC* and used against the final payment, or they are returned to the consumer if the trade is terminated early, i.e., if the producer times out on a data receipt. Importantly, however, this deposit cannot be used as leverage to ensure the consumer's honesty, because we have assumed that *SC* cannot distinguish between fraud and genuine data loss, i.e., in the brokered network. Thus, deposit details do not add to the specific model we are proposing, and we are not going to elaborate on them further.

Finally, it is important to note that the reputation manager (introduced in Chapter 5) should be able to update the participants' reputations at the end of each trade, i.e., based on the outcome of the trade. This is not straightforward, because a trade terminating early does not automatically apportion blame to either the consumer or the producer.

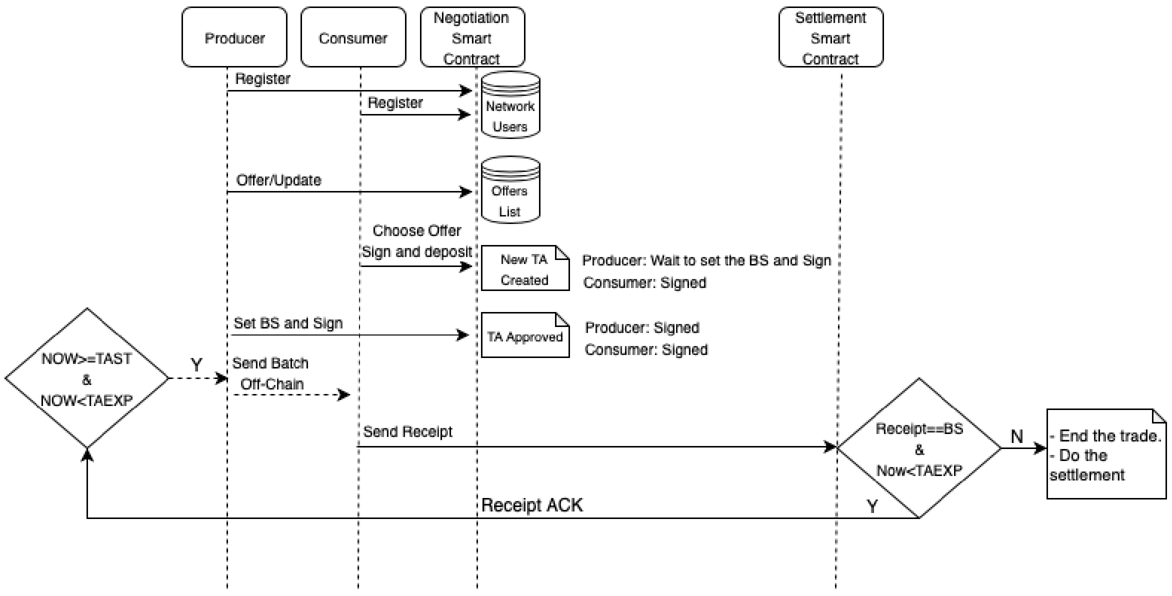
4.3 System View and Marketplace Interactions

The system consists of a data transfer layer, where IoT data transfer is mediated by brokers, and a blockchain layer, where all trade-related transactions occur.

In the data transfer layer, the actual data is transferred from producers to consumers off-chain (in the broker level) in different batch sizes, as stated in the trade agreement in the blockchain layer.

The blockchain layer consists of a collection of smart contracts *SC* written in Solidity [49], Ethereum's smart contract language, and executed on the Ethereum Virtual Machine (EVM).

Fig. 4.3 System Sequence Diagram



As shown in Figure 4.3, initially a new participant must register itself in the blockchain, by calling the register function of SC. Data providers P publish their data offers , or post updates to current offers, again using SC so that the offers are stored in the blockchain and are publicly visible. At the same time, consumers C can inspect offers, and then make a request to the SC including the reference to and the required time interval. This causes a new Trade Agreement Request to be created on the blockchain that is signed by $C's$ Ethereum private key.

The offer's provider is then involved in the definition of the agreement, which includes setting parameters BS based on the consumer's current reputation score.

Once the producer signs the agreement requested by the consumer, this is posted on the blockchain through a SC call as a new confirmed Trade Agreement TA.

Once the TA is in-force, the producer starts sending a data batch as agreed in the TA and then waits for a receipt. the producer is waiting RT_{max} as a maximum timeout. Once a receipt is sent within this time, a SC checks how much data was delivered compared to the BS. In the meantime, the next data batch is suspended. A data stream is resumed if the reported messages are equal to the BS, otherwise, the TA will be automatically terminated, and payment settlement and reputation management are applied.

4.4 Conclusion

In this chapter we have proposed a decentralised marketplace for trading brokered IoT data under assumptions of limited trust amongst the participants. Smart contracts on the Ethereum public network are used to mediate all interactions between data producers and consumers, in order to achieve non-repudiability and transparency.

The model separates the exchange of streaming data, which is supported by message brokers off-blockchain, from transactions that occur at regular checkpoints during data transfer. Our receipt-based model is expected to receive a receipt at a checkpoint after sending every data batch. Once a receipt is approved, the next data batch is sent.

This work is complemented by the reputation system in the next chapter, which aims to deliver a customised reputation model where reputation changes dynamically as a function of the history of past trades in the marketplace.

Chapter 5

Reputation Model

5.1 introduction

A reputation system is a trustworthiness measurement in the sense of reliability [15]. It is implemented in both centralised and decentralised environments and architectures. Furthermore, it is used in various domains like online communities such as StackOverflow, which applies a reputation system for its users in order to certify their trustworthiness. In a decentralised architecture, file sharing and other peer-to-peer networks use a reputation system to incentivise the network users to avoid free-riding [90].

Reputation systems are more commonly found in the E-commerce domain. The two best-known examples of reputation systems are found on Amazon and eBay. The former is considered to be one of the first successful reputation systems as online in centralised marketplaces [91], while the latter is one of the best-known reputation systems that has been analysed and studied how it works [91]. eBay reputation system follows the strategy that at the end of every transaction, every participant - sellers and buyers - can leave comments about each other. Each comment that is added by transaction participants has been tied to the this particular transaction. Each comment consists of one line of text, plus a numeric rating of +1 (positive), 0 (neutral), or -1 (negative).

The main objective, particularly in economic transactions, is to help create trust among sellers and buyers, and measure the quality of offered assets or even participants' credibility. Williamson in [92] showed various dimensions of economic activity trust: social trust, political trust, regulatory trust, professional trust, network trust and trust in the corporates themselves. Our trust dimension is participant credibility. The main challenge is to quantify and express the trustworthiness of participants in fair economic transactions.

In decentralised IoT data marketplaces with the absence of a trusted authority, the normal scenario is that there is no third-party authority to manage reputations and trades settlements or resolve participants' disputes. Instead, participants are scored based on their behaviours in these marketplaces. A data producer is rated by a consumer and similarly, a consumer is rated by a producer.

We have shown in Chapter 4 a decentralised, trustless marketplace for brokered IoT data trading using Blockchain in Ethereum that enables producers and consumers to trade in the absence of trust; however, it is managed by a reputation model. In this chapter, we present the reputation model that is tailored to address participants trust and the reputation management of theses trades in this marketplace.

Our approach is to design a system that quantifies every participant's trustworthiness with the lowest monetary and storage cost. The advantage of blockchain is credited for lowering operational costs [93]. It can reduce overhead cost of communications with low-priced transactions compared to traditional transactions. In our model, we aim to provide minimal fees for the transactions in the reputation system, which could be affected by the transaction data payload.

We envision that in our marketplace, participant reputation is not reviews based, as reviews are sent to the blockchain as a transaction payload and therefore incur more fees. Moreover, storing written reviews on the blockchain is unnecessarily exhausts a storage for the blockchain itself or for distributed storage platforms such as Swarm [76] or IPFS [55]. Instead, we have proposed a model that quantifies participants' trustworthiness based on their activities and trades. It only allows the participants' scores to be stored on the blockchain in a numerical representation.

The model demonstrates under which circumstance the marketplace scores a participant. In terms of consumer reputation, it encourages honest behaviour and eradicates fraudulent behaviour by applying penalties. Likewise for producers - it assigns them high reputation scores if they act responsibly, are quick to respond to consumers' demands and sign agreements. Following the trust assumption in producer delivery as shown in Chapter 4, this reputation system is built on the hypothesise that perfect delivery on the producer's side is based on a QoS protocol.

5.1.1 Decentralised Brokered IoT Data Marketplace

Our marketplace model presented in Chapter 4 is designed under assumptions of limited trust amongst the participants. It supports the trading of data streams, offers a data assets specification, stipulates legally binding trading agreements and ends with payment settlement for trade participants. It proposes a protocol involving data producers, data consumers, and a smart contract. It is based on the notion of periodic checkpoints during data exchange, supported by blockchain transactions to ensure limited scope for fraud on either side. It is a receipt-based model such that after every data batch sent, a checkpoint is carried out to verify that a receipt has been received from the consumer, reporting that the data have been delivered. Smart Contracts on the Ethereum are used to mediate all the interactions between data producers and consumers, in order to achieve non-repudiability and transparency.

The model separates exchanges of the data stream (data are transferred in the brokerage platform), from transactions that occur at regular checkpoints during data transfer in the form of data receipts. Every trade is stated in a trade agreement (TA), which is signed by its parties and deployed on the blockchain. The TA ends with payment settlements and scoring of the participants' reputations by using the reputation model. This allows a reputation score to be assigned to every pair of participants at the end of each transaction they are involved in.

5.1.2 Contributions

This reputation system is complementary to our work, and it is designed to be integrated with the marketplace model in our earlier work [94] (Chapter 4). This reputation system is tailored to address the trust and reputation management of participants in our marketplace. Its contribution is the marketplace's primary model for determining the participants' reputation scores and the criteria and rules that are used to quantify every participant's trustworthiness.

5.2 System Approach

The system is designed to build up a relationship between participant honesty in the marketplace and the trade opportunities they are involved. While participants may use reputation scores to assess the risk of entering into an agreement with an untrusted participant, it is obvious that each participant seeks a high reputation score in order to build his honesty in the marketplace and therefore gain more trades opportunities.

The researcher in [95] stated the characteristics that every reputation system for economic transactions must possess: (1) it should provide an information for its users -sellers and buyers- in order to distinguish between trustworthy and non-trustworthy participants; (2) it should incentivise sellers to be reliable; and (3) it should discard and discourage untrustworthy participants. The aim of this current work is to accommodate these requirements in line with our marketplace's needs, and to design a reputation system that can integrate with our model. Our system has no need for a monetary incentive to take part in the scoring, as opposed to what was mentioned in [71], which introduced "voucher" as an incentive for rating. In addition, it does not support written reviews as implemented in [69], or use extra platforms such as IPFS file system as implemented in [70].

Our approach involves designing a system that measures every participant's trustworthiness with the lowest monetary and storage costs. It aims to make the least possible use of blockchain storage with a minimal transaction fee. It also takes into consideration all the possible cases for calculating scores and correlating them to visible reputation in the marketplace. It provides firm resolutions to fraudulent cases that could appear, and finally discourages untrustworthy participants. However, to keep the marketplace booming and attract a considerable number of participants, the marketplace tolerates the dishonest attitudes, but imposes penalties.

With the assumption that data producers are trusted in their data delivery as shown in Chapter 4, this reputation system is built on the hypothesise that perfect delivery from producer's side is based on a QoS protocol, so data delivery is not considered in the calculation of scores. A variety of criteria are taken into consideration when measuring a producer's reputation.

The prime participants whose level of trust this system is designed to quantify are the Producer P, and the Consumer C. The system is an integrated set of rules, incentives and penalties that define the reputation score and therefore achieve the goals of the two sides in the system: $P's$ reputation and $C's$ reputation.

It intends to achieve a number of goals for defining $C's$ reputation:

- Keep the marketplace public so that any new C can trade.
- Define $C's$ trustworthiness in the marketplace -as a reputation score Rep^C- on a scale from 0 to 100.
- Define clearly the criteria of a trusted C, and likewise fraudulent behaviours that affect the reputation score.
- Define clearly which Cs are permitted and banned from trading.
- Maintain trading by an honest C and assign a reward by building up Rep^C.
- Apply the systematic stimulation of honesty by moving Rep^C up *faster* in successful trades with honest behaviours.
- Move Rep^C down if fraudulent behaviours are detected and apply penalties.
- Define clearly the penalty types and how they are applied to $C's$ score and account.
- Applying the systematic stimulation of honesty by moving Rep^C down *faster* in fraudulent behaviours.

Correspondingly, $P's$ reputation goals are the following:

- Allow any new P to trade in the marketplace.
- Define $P's$ trustworthiness in the marketplace -as a reputation score Rep^P- on a scale from 0 to 100.
- Define clearly the criteria of a trusted P, and likewise behaviours that affect the reputation score.
- Define clearly which Ps are permitted and banned from trading.
- Define clearly the response process for every trade request and how it affects the reputation score.

5.3 System Design

The system achieves the above goals by defining *nine* rules that show how reputation scores are quantified. It introduces a detailed and clear infrastructure for a reputation system, in the form of rules, that guarantee the characteristics of every reputation system must possess, as stated in [95]. This system, in the long run, is able to distinguish between the trustworthy and non-trustworthy participants, incentivise trustworthy participants to be reliable, and end by penalize and discourage untrustworthy participants. These rules introduce an exhaustive approach of scoring participants in different stages; before get trade, within trade considering an expected behaviours, and after trading that done in the marketplace model introduced in chapter 4.

Rule 1: A Score Limit

- Every reputation score in the marketplace is a value between the minimum limit 0 and the maximum limit 100.
- Any participant with a negative score is banned from trading. However, a tolerance is applied in some cases.

A participant's score in the marketplace can range from the minimum value of 0 and the maximum value of 100. A participant with a negative score is excluded from trading. A negative score means that continuous fraudulent behaviours have been committed, which means that trust is below the accepted level to trade. The reputation model is designed to measure how much a participant is trusted and it restricts or bans any continuous fraudulent attitudes, in order to achieve a healthy marketplace. With the negative score, a participant can no longer trade in the marketplace as the score reflects the participant's trustworthiness. However, the model tolerates negative-score participants according to the terms and conditions explained next.

Rule 2: Initial Reputation Score

- A new C reputation score in the marketplace starts with the minimum value 0, denoted by $Rep^C = 0$.
- A new P reputation score in the marketplace starts with the maximum value 100, denoted by $Rep^P = 100$.

The system model assumes trust in a producer's data delivery, therefore he is trusted to start trading with an initial reputation $Rep^P = 100$. The main behaviour for a producer in the marketplace is trust in publishing data. On the other hand, the initial reputation score for a new C is $Rep^C = 0$. The consumer's initial intention at the beginning of the marketplace is unknown, therefore he starts from the minimum value and builds up his trust based on his behaviours in trades.

Rule 3: Right to Stop

- Any participant in the marketplace (P or C) has the option to withdraw or stop continuing in a trade, however, penalties are applied to a commitment breach.

Any stop request made by C will be fulfilled *if and only if* it is requested in the last checkpoint of a current trade, and a penalty is applied for this breach. Any requests before or after checkpoint time will be discarded. A $C's$ stop request should be attached in the last receipt that reports the last data batch delivered. He is not allowed to stop within the time of sending a data batch, and instead, he must wait for the next checkpoint or decide to stop before the current batch is sent. If the stop request, as a blockchain transaction, is occurred within the time of sending the batch, the transaction takes some time to be confirmed in the blockchain, and this time is based on the gas price GUP paid of this transaction.

To clarify the process of verifying the transaction in the blockchain, GUP of a transaction determines the duration of the transaction verification by miners. In Ethereum (used for our implementation), higher GUP means that this transaction has higher priority for being validated by miners.

In the meanwhile, C continues subscribing the data and he may report the total data delivered until the time of the stop request. This costs a producer' loos for the data in the elapsed time of the transaction verification.

With regard to a producer, any stop request is fulfilled. As long as the hypothesise is the producer's trustworthiness in data delivery, he is trusted in his stops time.

Rule 4: Request Confirmation Time *RCT*

- Every P in the marketplace must set a Request Confirmation Time RCT^P.
- RCT^p is the maximum time that P may take to respond to a trade request. It can be updated.

P must set an RCT in order to let C know the maximum time for their requests to be responded. RCT^P can be longer or shorter based on $P's$ readiness to receive new trade requests. Setting a sufficient RCT^P corresponding to the request queue list is one of the criteria that affect Rep^P, as it shows a trade-off between the producer's reputation and the validity and the freshness of the data offered. Setting a long RCT^P to the request queue gives P an opportunity to respond to all the requests and avoid disregarded trades, thereby earning a high reputation. However, the long response time may mean that the real-time data stream loses its value. A producer has to manage a balance between these elements in order to keep his reputation high and the data live.

Rule 5: Trade Requests

- Every trade request in the marketplace is responded to with either: (1) Accept, (2) Reject, or (3) Disregard.
- Every trade request must be responded to by its P within RCT^P.
- The first trade request of every C in the marketplace must be accepted.
- Any rejection for a trade request, the fist trade of its C, denoted by RNC^P, affects $P's$ reputation.
- Any disregarded (unreported) trade, denoted by UR^P, affects $P's$ reputation.

Once P has a trade request from C, within RCT^P, P must accept or reject.

If P accepts, a new trade agreement (TA) is established between P and C.

If P rejects, and the trade request is the first request for C in the marketplace, denoted by RCT^P, $P's$ reputation is affected. The more RNC^P, the low Rep^P.

If there is no response by P within RCT^P, this trade is denoted by UR^P, and therefore, $P's$ reputation is affected. Similarly to RNC^P, the more UR^P, the low Rep^P.

Table 5.1 Consumer bands with a score range in every band

Band (B)	**Name**	**Score Range**
3	Trust	> 75 - 100
2	Good	> 50 - 75
1	Barley Enough	> 25 - 50
0	Low Acceptance	0 - 25

Rule 6: Score Bands and Parameter *K*

- $C's$ reputation in the marketplace is classified into four equal bands.
- Every band represents a level of trust that groups together a range of scores.
- Every C in the marketplace belongs to one band that reflects his trustworthiness in the marketplace.

As long as the high attention in the marketplace is attracting more C, motivating C to boost his trust is one of the main reputation system challenges for applying the systematic stimulation of honesty. We believe that classifying the scores into bands shows the different levels of trust. A high-trust participant is eligible for high trade chances as he is preferable over other parties for trading with. This variation in trust levels encourages C further to ascend to a higher level of trust and thus gain more trades opportunities. Table 5.1 shows four bands of different level of C' trust, with minimum value $B_{min} = 0$, and maximum value $B_{max} = 3$. For example, $B(x) = 1$, if $25 < x \leq 50$.

In every new trade, C is encouraged to attain a trust level higher than his current level before this trade. Suppose that c_j has a trade R in time $(t+1)$, $Cur_{t+1}^{c_j}$ represents the reputation score of c_j in the current trade R, where $Rep_t^{c_j}$ represents the reputation score of c_j in the marketplace for all his trades before trade R.

The focus is boosting $Cur_{t+1}^{c_j}$ as an incentive for further honest behaviours. To do this, we compare the trust band of $Cur_{t+1}^{c_j}$, with the band of $Rep_t^{c_j}$ to calculate the disparity between the two bands. High disparity is determined by $Cur_{t+1}^{c_j}$, which shows a sharp change in $c_j's$ behaviours. $Cur_{t+1}^{c_j}$ in a higher trust band than $Rep_t^{c_j}$ is an indication of a $c_j's$ desire to increase his cumulative trust and build up his $Rep_t^{c_j}$. Likewise, c_j may tend to put his trust at risk if his score is low in $Cur_{t+1}^{c_j}$.

There is a direct relation between the variation in the trust bands of the two reputations, and the maximum boosting value of the current band's scores. The current band is the band of the $Cur_{t+1}^{c_j}$.

Table 5.2 The values of the parameter K

$B(Rep_t^{cj})$ \ $B(Cur_{t+1}^{cj})$	**0**	**1**	**2**	**3**
0	**K=1**	$K = 1.02$	$K = 1.0267$	$K = 1.03$
1	$K = 0.96$	**K=1**	$K = 1.0134$	$K = 1.02$
2	$K = 0.92$	$K = 0.98$	**K=1**	$K = 1.01$
3	$K = 0.88$	$K = 0.96$	$K = 0.98$	**K=1**

The variation value between $B(Cur_{t+1}^{cj})$ and $B(Rep_t^{cj})$, denoted by E, is represented as the maximum value to be added or deducted to/from all score range in the band $B(Cur_{t+1}^{cj})$.

$$E = f(B(Cur_{t+1}^{cj}), B(Rep_t^{cj})) = |B(Cur_{t+1}^{cj}) - B(Rep_t^{cj})|, 0 \leq E \leq B_{max} \tag{5.1}$$

To generalise and apply the same proportion to every score, we have defined a moving factor K.

$$K_{(t+1)} = \frac{Cur_{t+1}^{cj} \pm E}{Cur_{t+1}^{cj}}$$

Then, it is multiplied by Cur_{t+1}^{cj}

$$Cur_{t+1}^{cj} *= K_{t+1} \tag{5.2}$$

The K value as shown in Table 5.2 defines the score range of every band. As shown in Table 5.1, we divided the scores into four equal bands where the upper scores of band 1, band 2, and band 3 (25, 50, and 75, respectively), are boosted by E. Let us examine an example for further explanation.

Suppose $Cur_{t+1}^{cj} = 50$ and $Rep_t^{cj} = 20$. From Table 5.1, $B(Cur_{t+1}^{cj}) = 1$ and $B(Rep_t^{cj}) = 0$, $E = 1$. To find K from Table 5.2, it is a function of the bands as in the following:

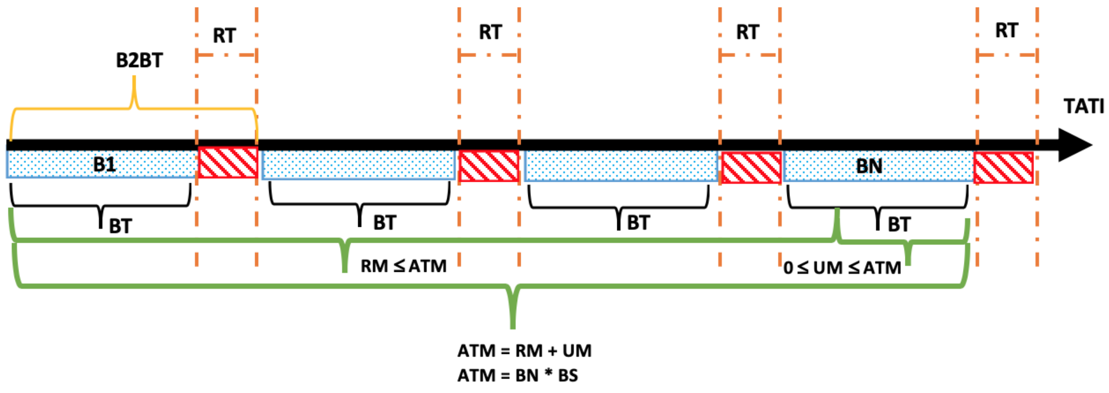
$$K_{t+1} = f(B(Cur_{t+1}^{cj}), B(Rep_t^{cj})) \tag{5.3}$$

$K_{t+1} = 1.02$, the new value of $Cur_{t+1}^{cj} = 51$ (from Eq. 5.2). This value of K shows that the upper value of band 1 which is 50 has been boosted by 1, the value of E.

Rule 7: Penalties of Consumer's Fraudulent Behaviours

- Any fraudulent behaviour carried out by C is penalised.

Fig. 5.1 *ATM* with comparing to *UM*, *RM* and *ETM*



- **A consumer penalty** Pnt^C is a numerical value ,in the range [0,100], that is defined for every failed trade caused by a fraudulent behaviour.
- Pnt^C is zero in two cases: (1) failed trades caused by a stop decision by one of the trade parties, and (2) successful trades.
- Pnt^C in a trade indicates the proportion of the data that has been unreported.
- Every new value for Pnt^C of a trade is deducted from Cur^C of this trade before it is accumulated in Rep^C.

As explained in the marketplace protocol in Chapter 4, once the *TA* is in force, in data transfer layer, *P*'s data stream is sent as batches. After every batch sent, *P* suspends its stream waiting for a confirmation receipt as acknowledgement from *C* as shown in Figure 5.1.

The number of messages between two checkpoints (RT) is a batch size *BS*, as agreed in the *TA*. $B1$ represents the first data batch sent within time BT, and *BN* is the last batch. At the end of the *TA*, the actual total number of messages (denoted by *ATM*) is the total number of messages in all batches sent from $B1$ to *BN*. *RM* is the total number of messages reported in $C's$ receipts. The unreported messages (denoted by *UM*) are the missed messages that have been disregarded (unreported) by *C*.

Fraudulent behaviour is the behaviour of reporting an incorrect number of messages not equal to *ATM*. A *C* who commits this behaviour reports $RM < ATM$, and disregards *UM*. In this case, a penalty is applied.

Let us use the same example as above. A trade R in time $(t+1)$ agreed between p_i and c_j. $ATM(R)$, $RM(R)$, and $UM(R)$ is represented as the following:

$$ATM_{(R)} = RM_{(R)} + UM_{(R)} \tag{5.4}$$

$Pnt^{c_j}_{t+1}$ indicates the proportion of the data that has been unreported. It is a percentage representation of how much data has been disregarded by c_j. The more $UM_{(R)}$, the more $Pnt^{c_j}_{t+1}$.

$$Pnt^{c_j}_{t+1} = \frac{UM_{(R)}}{ATM_{(R)}} \cdot 100 \tag{5.5}$$

Then, a penalty deduction is applied out of $Cur^{c_j}_{t+1}$.

$$Cur^{c_j}_{t+1} -= Pnt^{c_j}_{t+1} \tag{5.6}$$

Rule 8: Tolerance of Consumer's Fraudulent Behaviours

- A tolerance is applied to C whose score $Rep^C = Z < 0$.
- **One** opportunity is granted to every C in exchange for a cost that must be borne to resume trading in the market.
- The cost (denoted by Rz) is paid from $C's$ Ethereum account to the smart contract as an Ethereum tokens.
- Rz value is a proportion taken from the cost of every $C's$ local trade.
- C can resume trading with a new $Rep^C = 0$.

The model motivates C to behave honestly and grants him the opportunity to boost his trust to avoid being banned if his score is negative. However, $C's$ are allowed to resume trading in exchange for a cost. We believe that a payable opportunity may contribute to minimising the chances of continuous failed trades that leads to a negative score.

For this, the system tolerance is linked by the trades costs of C. It shows a trade-off between the trades costs of C and tolerance fee (Rz^C). A high trade's cost means either C has few costly trades or many uncostly trades. In the first case with few trades, C tends to behave honestly in his future trades to avoid paying the high tolerance fee Rz^C. In the second case, he may build his reputation from a number of trades. This is a motivation to behave honestly in future trades to avoid a decline in reputation and to be exempt from the tolerance fee. In the long run, this reduce the number of failed trades caused by fraudulent behaviours.

Rz^C is a monetary value that is paid as an Ether token from the Ethereum account of C to the smart contract account. It is a proportion taken from the local cost of every trade that who was involved in. It is quantified based on the value of local trades.

In time $(t+1)$ for a trade R between p_i and c_j, the local cost of trade R is the cost of *ATM* (Eq.5.4).

$$LC(R) = ATM(R) \cdot UP(R) \tag{5.7}$$

$UP(R)$ is the unit price as agreed in the TA of trade R.
$Rep^{c_j}_{t+1} = -Z$. $Rz^{c_j}_{t+1}$ is $Z\%$ of all local cost c_j.

$$Rz^{c_j}_{t+1} = \frac{|Z_{(t+1)}|}{100} \cdot LC^{c_j}_{t+1} \tag{5.8}$$

$$LC^{c_j}_{t+1} = \sum_{i=TrV^{c_j}_{t+1}} ATM^{c_j}_i \cdot UP^{c_j}_i \tag{5.9}$$

It should be noted that the more costly the local trades for c_j, the higher the value of Rz^{c_j} that should be paid. After paying Rz^{c_j}, c_j resumes trading in the marketplace, with the reputation score is boosted to $Rep^{c_j} = 0$.

Rule 9: Producers' Data Quality

- A data sold by P is quality rated by C, ,in the range [0,100].
- Every data batch may be rated by C in a receipt that reports this data batch.
- The data quality of P in a trade is the average of all rates of its data batches.
- The data quality of P in the marketplace is the average of all the rates of the trades he was.

One of the main challenges in measuring $P's$ reputation is quantifying his data quality. The term data quality has been widely used to describe the measure of the quality of data. It is generally measured against a set of criteria that assess the health of the data from many dimensions. Many industries create their own metrics for assessing data quality that fit their needs. A variety of professional data communities have set generic dimensions for measuring data quality; DAMA UK (dama-uk.org) is an example. It is one of the communities that manage data and information as key assets. It sets out six primary dimensions for data assessment: (1) Completeness, (2) Uniqueness, (3) Timeliness, (4) Validity, (5) Accuracy, and (6) Consistency.

It is not an obligation to use theses dimensions with equal weights. It may be prioritised certain criteria more strongly than others based different visions and goals. In a system like ours, we have not prioritised one dimension more than another, but rather we have combined more than one dimension and allowed C the opportunity to assess the data quality that has been received from P.

In our work, we separate the exchange of streaming data from transactions that occur at regular checkpoints during data transfer. It is difficult to assess the data that we do not have, as it is out of this work's scope. Rather, we assess the quality of the data by combining some dimensions to be measured by C where he is the direct receiver from P through the intermediate broker. Moreover, relying on the hypothesis of trust in $P's$ delivery for its real-time stream, data completeness and timeliness are guaranteed.

Q^P is the overall value ,in the range [0,100], that is determined by C to quantify the data quality of P. Following rule (9), Q^P is determined by following a hierarchical calculation.

Receipt Level: C rates the quality of $P's$ data in a batch with one single value (denoted by qr) attached in every receipt. Every data batch delivered might be rated by C with its receipt. At the end of every data batch, it is expected that C will send a receipt that reports the total messages delivered in the last batch, and in addition, he may rate the data quality of that batch by qr.

Suppose we have a trade R in time $(t+1)$ between p_i and c_j with N receipts. $qr^{p_i}(R,b)$ is calculated as the following:

$$qr^{p_i}(R,b) = x, 0 \leq x \leq 100, p_i \in P, c_j \in C, b \in N-1 \tag{5.10}$$

The rate of the last batch in the last receipt $qr^{p_i}(R,N)$ may not be counted. If a trade failed due to a fraudulent behaviour committed, the last receipt reports an incorrect number of messages delivered. An incorrect report in receipt N causes a monetary loss for p_i caused by disregarded messages. Consequently, a simple guarantee is provided for p_i by disregarding $qr^{p_i}(R,N)$ (if any), and instead, it is set to the highest value (100).

$$qr^{p_i}(R,N) = \begin{cases} 100 & \text{, if adversarial behaviour is detected} \\ x, 0 \leq x \leq 100 & \text{, Otherwise} \end{cases} \tag{5.11}$$

Trade Level: $Q^{p_i}(R)$ the data quality of p_i in a trade R is the average of all $qr^{p_i}(R)$.

$$Q^{p_i}(R) = \frac{\sum_{i=N} qr^{p_i}(R,i)}{N}, p_i \in P \tag{5.12}$$

Marketplace Level: the visible data quality for p_i in the marketplace is the average of the data quality rates for all trades p_i was involved in, as in the following:

$$Q^{p_i}_{t+1} = \frac{\sum_{i=TrV^{p_i}} Q^{p_i}(i)}{TrV^{p_i}}, p_i \in P \tag{5.13}$$

5.4 System Elements

Working with the rules stated earlier, the model is divided into two subsystems: consumer reputation and producer reputation subsystems. They define a tuple of parameters for every participant - P or C - and these are used to quantifying their reputations and achieve the above-mentioned goals.

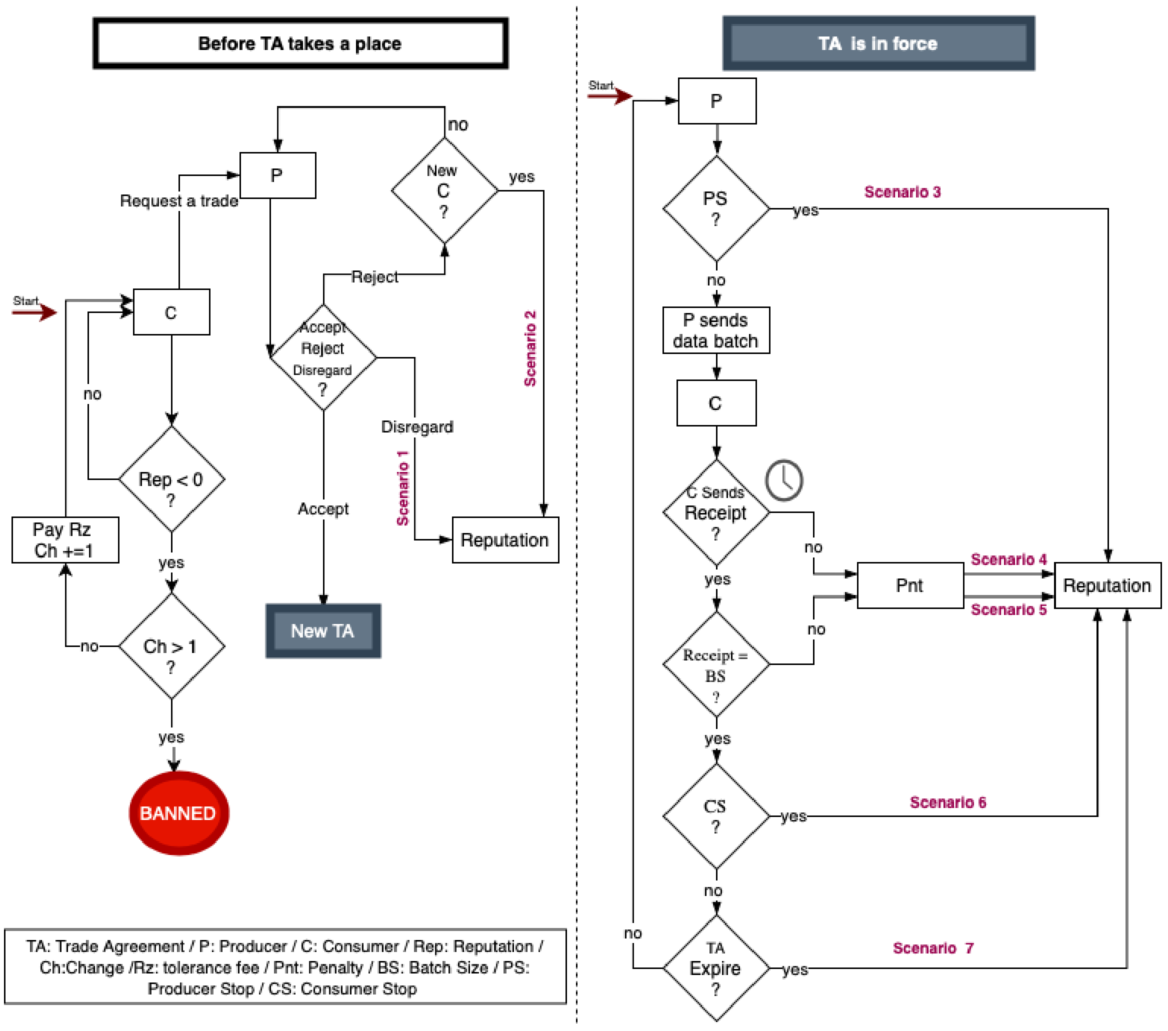
The model consists of the following elements:

1. **Consumer Reputation Subsystem**, which define c_j tuple with the following variables.:
   $c_j = \langle Cur^{c_j}, B(Cur^{c_j}), Raw^{c_j}, Rep^{c_j}, B(Rep^{c_j}), Pnt^{c_j}, Rz^{c_j}, TrV^{c_j} \rangle$ where: (i) Cur^{c_j} is the reputation score of the last trade he was involved in; (ii) $B(Cur^{c_j})$ is the trust band of Cur^{c_j}; (iii) Raw^{c_j} is a raw reputation as a mediator to accumulate the summation of all Cur^{c_j}; (iv) Rep^{c_j} represents the visible total and the final reputation in the marketplace; (v) $B(Rep^{c_j})$ is the trust band of Rep^{c_j}; (vi) Pnt^{c_j} is a penalty that is calculated at the end of each failed trade; and (vii) Rz^{c_j} is the amount that should be paid to the smart contract (in Ether) as tolerance fees if $Rep^{c_j} < 0$.

2. **Producer Reputation Subsystem**

   This is the subsystem that defines p_i tuple as the following:
   $p_i = \langle RCT^{p_i}, Res^{p_i}, Q^{p_i}, TrV^{p_i}, UR^{p_i}, RNC^{p_i}, Cur^{p_i}, Raw^{p_i}, Rep^{p_i} \rangle$ is a tuple of: (i) RCT^{p_i}, is request confirmation, (ii) Res^{p_i}, which is the percentage of the saved time of RCT^{p_i} that are not elapsed in responding to trade request, (iii) Q^{p_i} is data quality, (iv) TrV^{p_i} is the number of all trades p_i get involved, (v) UR^{p_i} is the disregarded trades requests that have been left without response within RCT^{p_i}, (vi) RNC^{p_i} is the rejected trades that represent the first trade requests for their consumers, (vii) Cur^{p_i} a reputation score for a current last trade involved, (viii) Raw^{p_i} is a raw reputation as a mediator to sum all Cur^{p_i} and then calculate the total reputation Rep^{p_i}, and (ix) Rep^{p_i} represents the total reputation in the marketplace which is visible to everyone.

Fig. 5.2 The Integration of the Marketplace Protocol with the Reputation Model



5.5 System Workflow Overview

The integration of the marketplace protocol in Chapter 4 with this reputation model is explained in detailed in Figure 5.2. It represents seven different scenarios and each ends by calculating the reputation for either P, C or both. The reputation scores are calculated *only* in these seven scenarios, which are classified according to the time of occurrence.

Let us examine a trade negotiation between p_i and c_j, only one of following scenarios can occur.

5.5.1 Before The TA approved

As we have shown in the protocol earlier, c_j can request trade from p_i from the available offers in the marketplace that belong to p_i.

At the beginning, following system rule (1), c_j is banned and can no longer trade in the marketplace if $Rep^{c_j} < 0$; however, one unique chance, denoted by ch^{c_j}, is granted to resume trading in exchange for Rz^{c_j} to be paid as an Ethereum tokens. Therefore, the system adjusts Rep^{c_j} to its new value, $Rep^{c_j} = 0$, and $ch^{c_j} + = 1$.

c_j is **BANNED** from trading in the marketplace with the same Ethereum address if there is a second drop in the score with $Rep^{c_j} = 0$, as shown in Figure 5.2.

Scenario 1:

p_i has a trade request in his queue, and within RCT^{p_i}, he did not respond by either accepting or rejecting. The request will be automatically disregarded and Rep^{p_i} will be recalculated and affected by this disregard.

Scenario 2:

p_i has a trade request in his queue from a c_j with $Rep^{c_j} = 0$. This request is the first request for c_j. p_i rejects this request. Rep^{p_i} will be recalculated and affected by this rejection.

It should be noted that if such a trade that is accepted by p_i, he has the right to set BS and it is not a function of Rep^{c_j} ($BS = 0$, if $Rep^{c_j} = 0$). He may set the smallest BS possible to limit his loss in the event of fraudulent behaviour being detected. Any requests from c_j (except first one) can be rejected within RCT^{p_i} without any effect on Rep^{p_i}.

5.5.2 When the TA is in force

This is the normal scenario if p_i and c_j have approved a new TA. p_i starts sending the first data batch as stated in the TA. The reputation scores for both p_i and c_j are recalculated and updated at the end of the TA. These differ according to the trade status and how the trade ends, as follows:

Scenario 3:

In such a trade, p_i is eligible to stop the trade anytime within $TATI$ by interacting the smart contract as a blockchain transaction. The trade will stop automatically and be described as *Failed*.

The reputation score for both p_i and c_j are as follows:

1. Rep^{p_i} will be recalculated and affected by this stop.
2. BS is resized to the time of the stop, so c_j is not bound to report the old value of BS.
3. Rep^{c_j} will be recalculated and it is not affected by this stop.

Scenario 4:

The first fraudulent behaviour occurs if no receipt is sent by c_j within RT_{max} as a maximum timeout. Such a trade is described as *Failed* and the reputation scores are as follows:

1. Rep^{p_i} will be recalculated and not it is affected by this misbehaviour.
2. c_j will be penalised and Pnt^{c_j} will be calculated.
3. Rep^{c_j} will be recalculated and updated with the effect of Rep^{c_j}.

Scenario 5:

The second fraudulent behaviour is sending a receipt with wrong messages report. This case is no different from scenario 4 and in both cases, Pnt^{c_j} is applied. The trade is described as *Failed*, and Rep^{c_j} is updated. Similarly, Rep^{p_i} will be recalculated and it is not affected by this misbehaviour.

Scenario 6:

As p_i is eligible to stop, c_j has the ability to stop a trade based on the system rule (3). It is fulfilled *if and only if*: (1) TA is not expired and c_j can stop within $TATI$, (2) c_j sends the receipt of the last batch and before start subscribing the next one, and (3) c_j sends the receipt of the last batch correctly. The receipt must report a number of messages the same as BS.

The trade is described as *Failed*, and it ends automatically after this stop. The reputation scores for both p_i and C_j are as follows:

1. Rep^{p_i} will be recalculated and it is not affected by this stop.
2. Rep^{c_j} will be recalculated and it is affected by this stop.

Scenario 7:

This is the last scenario and the only *Success* trade as it ends with the normal scenario of any successful trade. All the agreed data batches are sent, the correct receipts are submitted, and there is commitment to the TA with no stops. The reputation scores are recalculated with no penalties.

5.6 Reputation Updates

5.6.1 Consumer Reputation

A consumer reputation is calculated at the end of each trade, weather it succeeds or fails.

Any trade ending when the *TATI* has expired is considered *a successful trade*, while all other trades are *failed*. A failed trade is a trade that ends before *TATI* expiration due to: (i) *P* stops, denoted by *PS*, (ii) *C* stops, denoted by *CS*, or (iii) any fraudulent behaviour has been detected.

On occasion, *P* may decide to stop one or more of his current trades for a number of reasons, such as his desire to have a more profitable trade request in his queue list. Although this stop will affect Rep^P, the desire to sell more is the primary motivation for this *P*, especially if the trade is more valuable than the current one. However, in such a trade Rep^C will not be affected due to cancellation, as mentioned earlier.

C also has the opportunity to stop a current where he may found better data production source than the current one, or received the required data from the current one.

Using the same example, a trade *R* in time $(t+1)$ between p_i and c_j, the reputation scores calculation process for c_j runs as follows:

Calculate $Cur^{c_j}_{t+1}$

It is calculated as shown in Eq. 5.14. It is the average of:

- The reported messages received $RM(R)$ out of the total actual data $ATM(R)$ should be received within $TATI(R)$. It is represented by a in Eq. 5.15.
- The actual time $AT(R)$ is taken from $TATI(R)$. It is represented by b in Eq. 5.16. It represents the actual time that the trade was in force from its start time until it is ended by either: (1) $TATI(R)$ expiring, or (2) being terminated by PS^{p_i}. As mentioned earlier, c_j will not be affected by PS^{p_i}.

$$Cur^{c_j}_{t+1} = \frac{a(R)+b(R)+b'(R)\cdot PS^{p_i}(R)}{2}, p_i \in P, c_j \in C \tag{5.14}$$

$$a(R) = \frac{RM(R)\cdot 100}{ATM(R)} \tag{5.15}$$

$$b(R) = \frac{AT(R)\cdot 100}{TATI(R)} \tag{5.16}$$

$$b'(R) = 100 - b(R)$$

$$PS^{p_i}(R) = \begin{cases} 0, & \text{if } p_i \text{ does not stop the trade} \\ 1, & \text{if } p_i \text{ stop the trade} \end{cases}$$

Find K_{t+1}

From Eq. 5.3 and Table 5.2, we can find K_{t+1}. Then, it is multiplied by $Cur^{c_j}_{t+1}$, as shown in Eq. 5.2.

Calculate $Pnt^{c_i}_{t+1}$

(This step for trades of fraudulent behaviours only).

$Pnt^{c_j}_{t+1} = 0$ in: (1) successful trades, (2) failed trades caused by stops. $Pnt^{c_i}_{t+1}$ is a percentage representation of how much data has been disregarded. It is calculated by using Eq. 5.5, and it is deducted from $Cur^{c_j}_{t+1}$, as shown in Eq. 5.6.

Update $Raw^{c_j}_{t+1}$

$Raw^{c_j}_{t+1}$ is a mediator variable to sum all Cur^{c_j}, and it is then used to calculate the $Rep^{c_j}_{t+1}$.

$$Raw^{c_j}_{t+1} = Raw^{c_j}_{t} + Cur^{c_j}_{t+1} \tag{5.17}$$

If we combine Eq. 5.2, Eq. 5.6, and Eq. 5.17, we can have:

$$Raw^{c_j}_{t+1} = Raw^{c_j}_{t} + Cur^{c_j}_{t+1} \cdot K_{t+1} - Pnt^{c_j}_{t+1}$$

Calculate $Rep^{c_i}_{t+1}$

It is the total publicly visible reputation which represents the trustworthiness of c_j in the marketplace.

$$Rep^{c_j}_{t+1} = \frac{Raw^{c_j}_{t+1}}{TrV^{c_j}_{t+1}} \tag{5.18}$$

$Rep^{c_j}_{t+1}$ will be visible for everyone in the marketplace, and once c_j is involved in a new trade, a new iteration of the above steps will be followed, unless $Rep^{c_j}_{t+1} = Z < 0$.

Pay $Rz^{c_j}_{t+1}$

Following the rule (1) which states that any C must be excluded from trading if his reputation is negative, the system is tolerates such consumers. c_j has one unique payable opportunity that pays $Rz^{c_j}_{t+1}$ to the smart contract as an *Ether token*. Consequently, the smart contract set $Rep^{c_j}_{t+1} = 0$ to resume trading. This is calculated as shown in Eq. 5.8 and Eg. 5.9.

5.6.2 Producer Reputation

A trusted P is a preferable trading partner for C to establish a trade with. P is trusted if he has a high reputation score Rep^P. As long the system is built on the hypothesis that P is trusted in data delivery, the system has different criteria that help to quantify $P's$ trustworthiness in the marketplace.

In our system design, high Rep^P means:

- High rate for data quality Q^P.
- Quick response, denoted by Res^P, to all trades requests within RCT^P.
- Reduction or absence in the number of unreported or disregarded requests, denoted by UR^p.
- Reduction or absence in the number of rejections of first trade requests of a new C, denoted by RNC^P.
- Commitment to trade clauses with a reduction in or absence of trade stops, denoted by PS^P.

The reputation score calculation process for p_i in time $(t+1)$ in trade R with c_j, runs as the following:

Calculate $Cur^{p_i}_{t+1}$

$Res^{p_i}(R)$ is the percentage of the saved time of RCT^{p_i} that has not elapsed in responding to trade R. The higher the $Res^{p_i}(R)$, the quicker p_i is responding to R. The reputation of p_i is affected by the speed of his response to requests.

$$Res^{p_i}(R) = 100 - \left(\frac{m(R)}{RCT^{p_i}_{t+1}} \cdot 100\right) \tag{5.19}$$

Where *m(R)* is the actual recorded time of $p_i^{\prime}s$ response to trade R. $Cur_{t+1}^{p_i}$ is the average of: (i) the response speed $Res^{p_i}(R)$, Eq. 5.19, (ii) the data quality $Q^{p_i}(R)$, Eq. 5.12, and (iii) the actual time taken from $TATI(R)$, Eq. 5.16.

$$Cur_{t+1}^{p_i} = \frac{(Res^{p_i}(R) + Q^{p_i}(R) + b(R))}{3} \tag{5.20}$$

Update $Raw_{t+1}^{p_i}$

$Raw_{t+1}^{p_i}$ is a mediator variable to sum all Cur^{p_i}, and it is used to calculate the $Rep_{t+1}^{p_i}$.

$$Raw_{t+1}^{p_i} = Raw_{t}^{p_i} + Cur_{t+1}^{p_i} \tag{5.21}$$

Calculate $Rep_{t+1}^{p_i}$

$Rep_{t+1}^{p_i}$ is the total publicly visible reputation which represents the trustworthiness of p_i in all his trades in the marketplace.

$$Rep_{t+1}^{p_i} = \frac{Raw_{t+1}^{p_i}}{TrV_{t+1}^{p_i} + RNC_{t+1}^{p_i} + UR_{t+1}^{p_i}} \tag{5.22}$$

Rep^{p_i} is recalculated if one of the formula's parameters has changed.

5.7 Discussion and Conclusion

This chapter introduced a reputation system for a trustless decentralised brokered IoT data marketplace. It follows our previous work on marketplace architecture that introduced all the marketplace elements, with the absence of trust and reputation management for marketplace participants, P and C. It introduced a model of how to mathematically define the reputation scores of the participants, showing which criteria and rules have been taken into account to adjust the scores up or down. It defined the fraudulent behaviours and showed all the scenarios in which the reputation scores must be recalculated. It is complemented by the model in Chapter 4 as both represent our decentralised trustless marketplace for brokered IoT data.

As with all such systems, it has limitations and issues, and these are detailed below.

Sybil Attack: A P starts trading with the maximum reputation value in the marketplace due to the hypothesis of his delivery trust. This potentially causes adversarial behaviour by a consumer aiming to subvert this reputation. A C may create multiple identities and exploit the rule which states that P must accept the first trade request, and then terminate with failure.

Moreover, P would suffer monetary losses due to the unreported data from those trades. This would be a benefit for C in two ways: the trade would be free of charge data, and it would lower the reputation of P.

For identity validation, a unique Ethereum address is required to register in the marketplace. Creating multiple identities needs different Ethereum funded addresses to afford the transactions costs of sending the receipt at every checkpoint. As long as P has the ability to set the batch size when C is new with reputation 0, he could send the smallest batches in order to mitigate his monetary loss from unreported data. Subsequently, C has to pay a high cost for many transactions due to the multiple small batches receipt in one trade. The total cost includes the cost of every first trade C has from all the new multiple identifies.

Moreover, each Ethereum address would not be able to be used for another trade due to the penalty applied from the previous failed trade, which would lower the reputation score to less than 0, and therefore Rz^C would be applied. We believe that the monetary cost borne by an adversarial consumer in the long run would strongly deter him from his fraudulent behaviour.

Inaccurate Data Quality: While the data quality is directly proportional to the $P's$ reputation in the marketplace, C may under-rate the data batched delivered as an adversarial behaviour to subvert the $P's$ reputation. This may not accurately reflect the $P's$ data quality. P has the capability to stop the trade if he thinks that his reputation will be significantly affected by this rating. Although the stop will affect his reputation, stopping the trade could have less affect than a low data quality rated by C. Thus, the decline in reputation is mitigated but this is still an issue. A solution to further mitigate this problem is to choose Cs with a high reputation score to trade with, as fraud is not expected with these Cs.

Set a long time for RCT^P: As $P's$ reputation is directly proportional to their quick response to requests, and the response is a fraction of the RCT^P, P may set a long RCT^P as this gives him the opportunity to increase the value of Res^P.

The marketplace relies on the trading real-time data streams, which need a quick response for subscription, otherwise they lose their value and are no longer needed.

Rapidly growth in C reputation from many small trades due to parameter K:

A C may seek to build up his reputation from small trades in order to have larger data batches in his future trades. Although he may earn a high reputations in a short time, he is exposed to a quick drop due to parameter K if any further fraudulent behaviour is detected. In addition, the penalty is linked to the extent of this misbehaviour, which is affected by the sharp drop.

Chapter 6

Implementation and Evaluation

6.1 Introduction

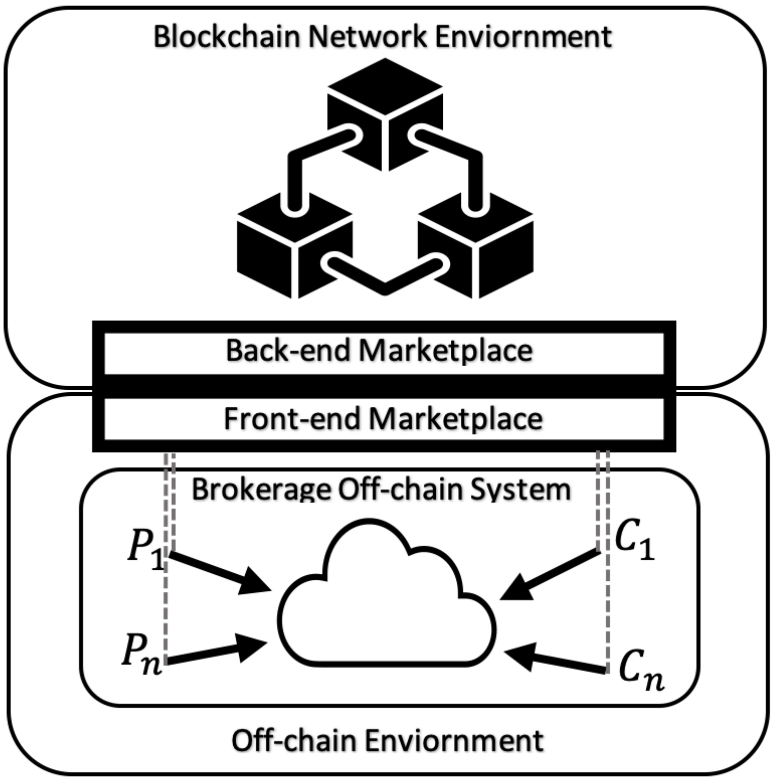
This chapter presents the technical implementation of the marketplace model and its reputation system. It shows how the model is designed and integrated with the reputation system, which is tailored to fit its needs. Moreover, it technically shows how it is structured and programmed in both its layers: the data transfer layer and the blockchain layer. A simulation approach has been taken to model participant communication in the marketplace, and to demonstrate how the reputation system measures their trust based on the rules and criteria defined in the system protocol. We experimentally evaluate the trade-off between the overhead cost of trading and the level of participant trust. Another angle of the system evaluation is introduced when we experimentally show the latency of transactions for an Ethereum network in processing and confirming our marketplace transactions.

6.2 Marketplace Implementation

The marketplace is an integration of many sections that are implemented separately and then linked together as a Decentralised Application (DAPP), where all the marketplace functionalities are present.

As explained in the model architecture in Chapter 4, the marketplace separates the exchange of streaming data, which is transferred off-chain using a brokerage platform, from transactions that occur between the producer and the consumer for trade agreement affairs. This classifies the marketplace implementation based on the layer's functionality into two layers: the data transfer layer and the blockchain layer.

Fig. 6.1 The marketplace architecture and its implementation in blockchain and off-chain environment



We also distinguish a on-blockchain and a off-chain component to the marketplace architecture. The next explanation follows this classification.

The marketplace implementation as shown in Figure 6.1 is classified based on its existence on the network in two layers:

Network Layer: This represents the use of Ethereum (ethereum.org/en/) and its smart contract, which run on Ethereum blockchain. It helps to automate and enforce legal obligations for trades in the marketplace.

Off-chain Layer: This is divided into two parts:

1. The marketplace front-end development, which is implemented by Python, and connected to the Ethereum smart contracts (the network environment).

2. A brokerage platform that represents the medium to transfer IoT data between the producer and the consumer.

6.3 Blockchain Layer

The core advantage of the second generation of the blockchain is the smart contract. It is a way of enforcing the obligations automatically by writing a piece of code using Solidity [49]. Our marketplace is leveraged by implementing a smart contract on Ethereum blockchain.

6.3.1 The Marketplace Back-End

The smart contract of our marketplace acts as the marketplace admin, managing and controlling all trade affairs. It helps with automating and enforcing legal obligations such as those stated in a trade agreement. It represents the marketplace back-end, which governs the obligations from many aspects: data offering, trade negotiations, monitoring the course of trades and their completion, payment settlements and ends by participants' reputation.

It achieves the marketplace obligations and functionalities through many methods written with Solidity where every method represents one of the system's functions. It shows the system workflow, starting from registration in the marketplace, and ending with participants' reputations. All methods descriptions are detailed in the appendix A.

6.4 Off-Chain Layer

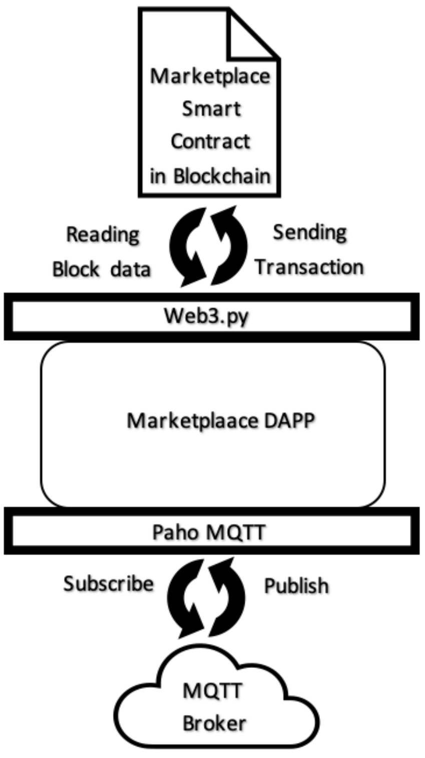
6.4.1 The Marketplace Front-End

The marketplace was developed as a decentralised application (DApp), which communicates with the blockchain to manage the state of all network actors .*DApp* is a computer application that runs on a distributed computing system such as Ethereum and Bitcoin. It has a back-end code that runs on the blockchain in the form of the smart contract. It can also have a front-end code, which could be implemented and written in any language. Whatever its front-end implementation (mobile apps or servers), the key data and operations are kept in smart contracts in a blockchain. In theory, a single smart contract can be seen as a DApp [96].

The off-chain marketplace section is represented as a piece of software that communicates with the blockchain through the front-end development, where it can make calls to its smart contract back-end. It was written and designed using Python3 [97] and the *Web3* library [98], to make calls and interact with the smart contracts in the form of sending transactions, reading block data, and making monetary settlement in ether currency.

The*Web3 library* is the Ethereum API in Python, which consists of many functions that help the application to connect to Ethereum-through Python- to send transactions and read block data, and for a variety of other use cases.

Fig. 6.2 The use of Ethereum API (web3.py) and Mqtt broker API (Paho-mqtt) with our marketplace DAPP



It is the main gate of the marketplace interface, with every participant's activity is accomplished through this interface. It consists of many functions that are eventually connected in the back end with the blockchain.

6.4.2 MQTT Broker Platform

The marketplace is modelled to send the data from the producer to the consumer through a standard broker for IoT data streaming, the *MQTT broker*. We have adapted the open-source Mosquitto MQTT broker [79] as a mediator to support transferring the streaming data between the data producer and its consumer. It has been set up in our front-end development by using Eclipse Paho [99], as shown in Figure 6.2. Paho is a MQTT Python client library that provides group of functions which enable applications to connect to the MQTT broker to publish messages, subscribe to topics and receive published messages. It allows our application to be a client that is able to publish or subscribe to data streams from the MQTT broker.

Table 6.1 Dataset specification

Sensor Type	Temperature
Theme	Weather
Brokers	aq_mesh_api

Table 6.2 Dataset structure of the dataset

Sensor Name	Variable	Unit	Timestamp	Value	Flagged as suspect reading

6.5 IoT Data Stream

This system uses a dataset of a real-time data from the Urban Observatory project [100] as a test IoT data stream. The Urban Observatory is the largest set of publicly available real time urban data in the UK. The real-time data is generated from publicly available sensors in the city of Newcastle, which gather data across the city with different parameters such as: Weather, Traffic, Noise, Vehicles, and Air Quality.

For the purpose of system evaluation, we randomly chose a raw sensor dataset for the last 28 days (the time since our evaluation started) with the following parameters as shown in Table 6.1.

The air temperature in the dataset is measured by a sensors and refreshed second by second with a *Celsius* unit. It is structured as shown in Table 6.2:

The first data batch is sent when the trade agreement *TA* is enforced and it is stored on a *.csv* file in a consumer client, while the second one is paused until receiving the receipt of the first batch. is delivered. If the receipt has been delivered, data streaming resumes and a consumer is subscribed. All data streams are appended on the same *.csv* file such that at the end of the *TA*, a consumer has a full *.csv* file that contains a full stream of the period stated in the *TA*.

6.6 Marketplace and Reputation Model Simulator

Simulation provides a way to validate how the marketplace would perform with multiple producers and consumers with varying degrees of trust. The primary goal of designing this marketplace is to have an environment built on the rationale of fair trading in a trustless climate. This simulator allowed us experimentally to meet the needs of a fair marketplace with a novel reputation manager for its participants to quantify their trust. It is configured to replicate the behaviour of multiple pairs of producers and consumers in different trading scenarios.

Every producer or consumer is able to trade based on the marketplace policies. All their all behaviours are recorded in order to reflect a long-run trading attitude in the marketplace. It is capable of showing how the trading and scoring mechanism will be in the actual trading.

Our simulator goal is to show the marketplace functionalities that affect on participants' reputations. It simulates the marketplace DAPP by presenting the fundamental obligations to which every producer or consumer is committed, and affecting their reputations, and eventually ending by updating their reputations. It simulates all possible behaviours that may occur: trade requests, trade responses, and enforcing a trade agreement. To facilitate this task, we assume that all producers and consumers are registered on the marketplace with their Ethereum addresses. Moreover, we assume trade offers are already published and available for consumers' requests.

Suppose that we have interacted with our marketplace interface, which is published on the real blockchain Mainnet at time t. To clarify, blockchain Mainnet is the live blockchain where a blockchain protocol is fully developed and deployed, and any transaction occurring will be recorded on its blocks.

The simulator shows what the producer and the consumer can do from time t as the following:

A consumer is able to:

- Send a trade request.
- Send correct receipt for data batch received for current in-force trade.
- Commit fraudulent behaviour by sending receipt that reports an incorrect number of messages delivered.
- Assess data quality delivered with high rate.
- Assess data quality delivered with low rate.
- Disregard the data quality rate.
- Finalise current trade successfully with no stop.
- Stop current trade.

A producer is able to:

- Accept trade request.
- Reject trade request.

- Disregard trade request where it is left unreported within Request Confirmation Time RCT^P.
- Accept trade request from new C_j with $Rep^{C_j} = 0$.
- Reject trade request from new C_j with $Rep^{C_j} = 0$.
- Finalise current trade successfully with no stop.
- Stop current trade.

6.6.1 Simulator Architecture

The advantage of our system is its totally reliance on a participant behaviour in trading practice, providing honesty and a lack of bias in calculating the reputation of everyone involved in a trade, once the trade is completed. This gives our marketplace the reliability in terms of our participants' scores due to the total judgement of reputation being based on how participants behave in their trading activity.

We adopted a probabilistic approach to show all possible cases and participant behaviours in trading. In addition, it shows how the model works in calculating the reputations with different trade parameters.

Intent Probability

A participant's initial intention upon entry into the marketplace is unknown, as he has not engaged in any trades yet, so cannot be judged as having high or low reputation.

We adopt a probabilistic approach to draw the expected conclusion regarding participant behaviour based on his intent probability. The intention probability gives our simulator the variety in participant' attitudes between honesty and fraudulently. A trusted participant is expected to end a trade successfully, while, on the other hand, a fraudulent participant is expected engage in fraudulent behaviours.

To facilitate this task, we have created two different profiles: one for expected honest behaviours (denoted as Group H), and the other for the fraudulent behaviours (denoted as Group C). We have applied intent probability to distribute every participant to the appropriate group according to his expected behaviour.

A participant is classified into either *Group H* or *Group C* based on their given intent probability. A participant in *Group H* tends to have high probability to be honest in his attitude, while participants in *Group C* are expected to engage in fraudulent behaviours as shown in Table 6.3.

Table 6.3 Behaviour probabilities for participant profiles

Profile	**Participant**	**Behaviour Probabilities**			
		Stop the Trade	Accept requests from new consumers	Accept requests within RCT	Send a correct receipt for data batch received
GROUP (H)	Producer	Low	High	High	-
	Consumer	Low	-	-	High
GROUP (C)	Producer	High	Low	Low	-
	Consumer	High	-	-	Low

A *Group H producer* is expected to: (i) terminate his trades successfully, (ii) accept trade requests from new consumers, (iii) stick with the pre-defined period of request confirmation time *RCT* and respond to requests within this period.

Conversely, a *Group C producer* is a participant who is strongly expected to have frequent withdrawals, few confirmed requests of new consumers, and many disregarded requests that must be responded to within *RCT*.

In terms of consumers, a *Group H consumer* is strongly expected to behave honestly and send all receipts correctly, with few cases of stopping trades, or the complete absence of this behaviour. A *Group C consumer* who continuously engages in fraud, with frequent stops occurring. It is highly probable that he will tend to withdraw from trades, or trades end in failure due to either incorrect message reporting or not sending a receipt.

Receipt Probability

As long as the receipts belong to consumers, *receipt probability* is applied as follows:

Suppose that we have a trade *i* with *x* receipts, *the receipt probabilities* that are applied in our simulator are used to answer these questions:

1. How likely is it that a consumer will send a $receipt_1$ ... $receipt_x$?
2. For every $receipt_1$ ... $receipt_x$, how likely is that a consumer will correctly report the number of messages delivered ?

6.6.2 Simulator Implementation

The simulator has been implemented in Python 2.7 [97] by using the SimPy simulation library [101]. Its approach is built on the concept of *real-world processes programming*, using the SimPy library. The SimPy library is a process-based discrete-event simulation framework based on standard Python. It comes with different characteristics and features, making it one of the simplest simulation packages.

Every process represents one of our marketplace components, such as producer, consumer, and smart contract. The components interact together to simulate the prime functionalities of each of them in the marketplace, starting by responding to trade requests, sending data batches, receiving data batches, sending receipts for every batch received, ending the trade, and finally, updating the reputation score based on a current trade status.

6.7 Marketplace Evaluation

Our marketplace is evaluated at different implementation levels, with each using different parameters. **First,** at the simulator level, we have evaluated the marketplace functionality and its integration with the reputation system to draw a conclusion of how participants are scored in different scenarios. This functionality has been evaluated in our simulator, as explained earlier in Sec. (6.6). **Second,** we have moved up to the test level of implementation and deployed the marketplace through a local blockchain connected through **Ganache** [102], formally known as *Ethereum TestRPC*. Ethereum TestRPC or ganache, is a local fast blockchain emulator designed for development and testing. It simulates the features of the Mainnet Ethereum network and provides funded accounts with test Ether for testing purposes. We have implemented our second evaluation that shows the trade-off between the overhead cost and consumer reputation. **The Final** evaluation is tested on the real Ethereum Mainnet where it evaluates the transaction latency in Ethereum. It shows the confirmation time of processing transactions *RT*, with different *GUP*, for one individual trade and for a high workload number of concurrent trades.

6.7.1 Simulator Evaluation

The model testbed is our simulator (explained in Sec. 6.6.1), which experimentally replicates the behaviour of multiple pairs of producers and consumers in different scenarios and shows how reputations are quantified.

For the purpose of evaluation, we have run the simulator with **ten** producers and **ten** consumers, distributing them into group profiles, and then tracing their expected behaviours through 2000 trades. In every trade, a pair of participants (producer and consumer) is picked randomly to trade, and this explains why the producers and consumers have been involved in different numbers of trades.

As the purpose of this experiment is tracing the expected behaviours with variety levels of trust, and showing how the reputation system quantifies the score, we picked the intent probability of consumers from the highest band (Honest) and the lowest band (Low Acceptance), as shown in Rule (6) Sec. 5.3.

To simplify the choice, we picked 80 as an initial intent probability for consumers in *Group H*, and 20 for consumers in *Group C*, where both values are in the highest and the lowest bands. We also applied the same intent probabilities for producer groups to unify the values in all groups where the purpose is similar.

Table 6.4 The initial participant probability for two group profiles

		Probability			
	Participant	Stop the trade	Accept requests from new consumers	Accept trade request within *RCT*	Send receipt for data batch received
Group H	Producer 1,2, 3,7,8	20	80	80	-
	Consumer 1,2 4,6,8	20	-	-	80
Group C	Producer 0,4 5,6,9	80	20	20	-
	Consumer 0,3 5,7,9	80	-	-	20

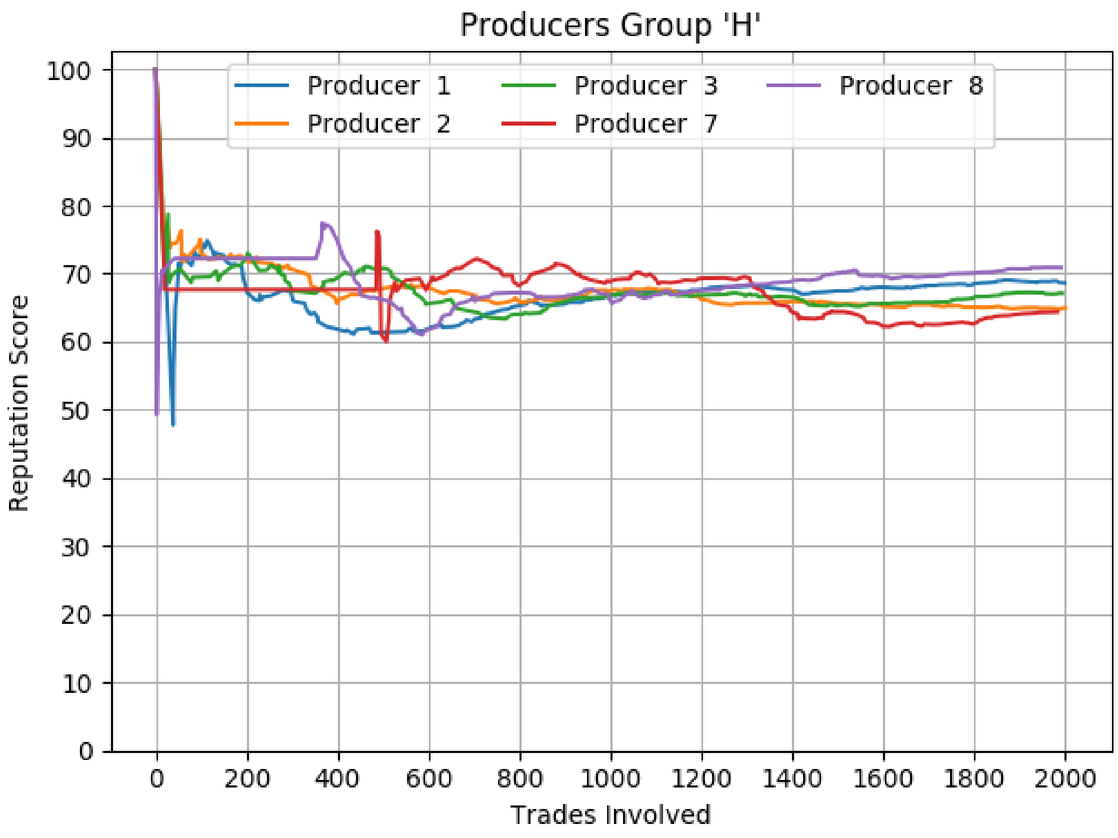
In Table 6.4, we have classified producers and consumers into two groups based on the intent probability they received from the simulator. Then, all of them will trade in the marketplace based on their initial given intention.

A trusted producer is willing to make trades with new consumers and strives hard to process all trade requests in his queue before RCT^P expires to avoid disregarded trades with 80% probability. On the other hand, an un-trusted producer tends to stop the trade with 80% probability and with 20% probability to be trusted. Similarly, a trusted consumer behaves with 80% probability of ending a trade successfully and has the commitment to send receipts for all data batches received. This is unlike the fraudulent behaviour, where a consumer tends to cheat in one of the receipts or stops the trade before the expiration of the *TATI* as agreed.

Table 6.5 shows all *ten* producer behaviours, and the logs for trades in which they were involved. Every producer has been picked randomly to be involved in trades for 2000 trade simulations, which caused each producer to be involved in a different number of trades. Every producer's *request confirmation time RCT* was set for 2*mins*; they were required to respond to trade requests within *RCT*, otherwise it was considered disregarded. All trade

Fig. 6.3 Reputation scores of all producers through their trades

(a) Profile for Group H Producer



(b) Profile for Group C Producer

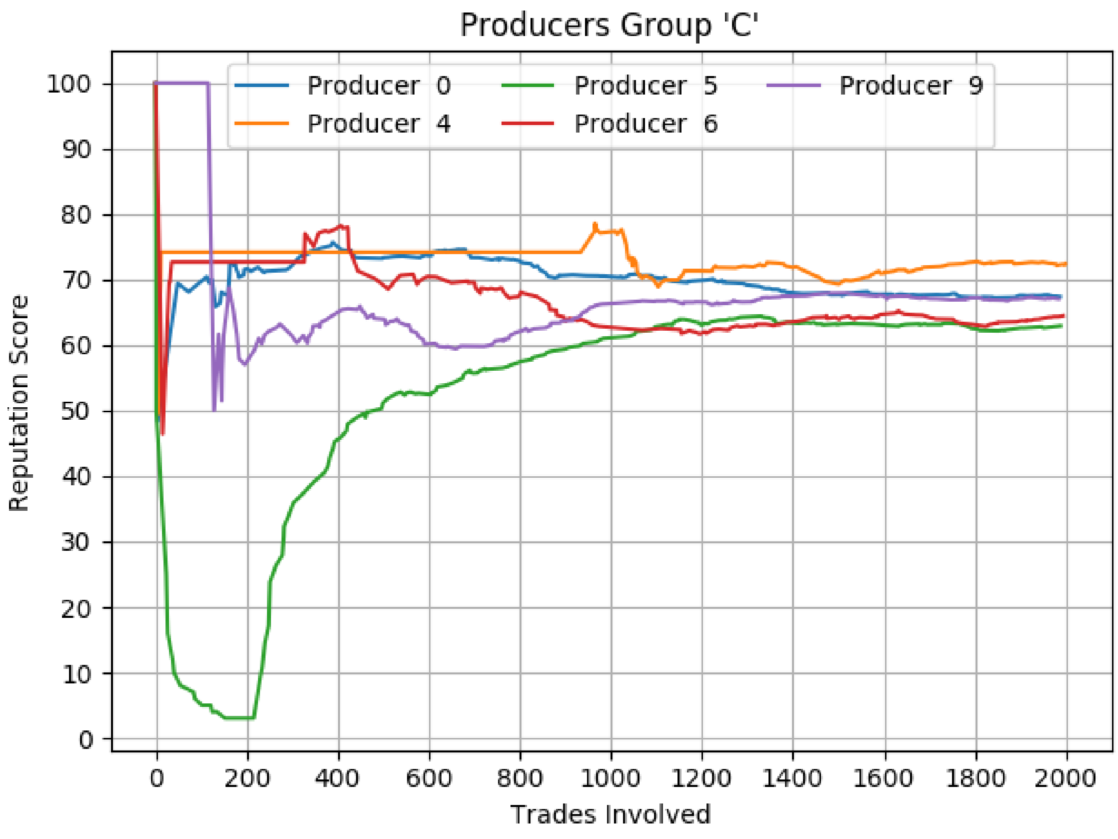


Table 6.5 Producers' behaviours in their trades

Id	*UR*	*RNC*	*TrV*				Avg *Res* (*RCT%*)	Q	End Rep	Max Rep	Min Rep
			Total	Success	Failed						
					PS	Other					
0	0	0	236	111	85	40	40.40	64.34	75.68	75.68	48.39
1	0	0	223	124	77	22	39.57	65.5	68.3	74.87	47.76
2	0	0	219	106	88	25	40.87	59.69	64.95	83.33	64.83
3	0	0	226	100	78	48	41.57	64.50	67.09	88.89	63.36
4	0	5	124	75	29	20	39.52	68.41	72.41	78.6	49.31
5	15	0	204	87	67	50	49.03	64.01	62.91	64.42	3.00
6	0	2	170	69	66	35	38.45	59.25	64.47	78.23	46.43
7	0	4	178	66	72	40	39.31	58.07	64.39	76.21	60.02
8	0	8	206	100	52	54	38.38	65.95	70.87	77.45	49.3
9	1	0	214	134	63	17	40.63	62.51	67.05	100	50

involvement is denoted by *TrV*, which includes *Success* and *Failed* trades. Furthermore, it shows the number of disregarded trades *UR*, the number of rejected trade requests of new consumers *RNC*, total trades stopped *PS*, the average of all trades *Res* (time taken out of request confirmation time *RCT*), the quality of his data *Q*, the minimum and maximum reputation score for every producer, and the current *Rep* in the marketplace denoted by *End Rep*.

The table shows that the two highest reputations are for P_0 and P_4 with scores of 75.68% and 72.41%, respectively. Although Q^{P_4} at 68.41% is rated higher than Q^{P_0} at 64.34%, and he acted with 39.52% of *RCT* elapsed, which means he responded to requests faster than P_0, P_4 rejected 5 requests of new consumers. This shows, as expected, that reputation and *RNC* are inversely proportional.

P_2, P_6, and P_7 has achieved the same reputation score of 64%. Although P_2 had the most successful trades, he also had the most stops with 88 trades, while P_6 achieved the fastest response, with an average of 38.45% represented by a response within 32 seconds. P_7 represents the highest number of rejected trades of new consumers, with 4 trade requests rejected.

This shows the inverse correlation between reputation and *RNC*, *Res*, and *PS*, as expected. Every producer must manage a balance between those elements to keep his reputation high.

Moreover, the table 6.5 shows the system's ability to build a reputation again, due to the frequent occurrence of one of the elements that affects reputation.

Leaving trade requests without responses within *RCT* causes a drop in producer reputation, as occurred for P_5. He disregarded 15 trades which dropped his reputation sharply to

3%, as shown in Fig. 6.3(b). However, with more successful trades, he built up his reputation gradually to ends by 62.91%.

Fig. 6.3 presents all producers' reputations through their trades, with fluctuation from high to low scores, as expected. It is not surprising that the score of an honest producer may move down due to different reasons, with their initial intention given not a guarantee for their honest behaviour in the long run. Likewise, for a producer with low initial intention, the intent probability is nothing but a prediction as to what his behaviour could be in the next trade. In addition, after roughly 200 trades for every producer, the change in score is slight, with this providing an incentive to create an honest reputation earlier.

Table 6.6 Consumer behaviours in their involved trades

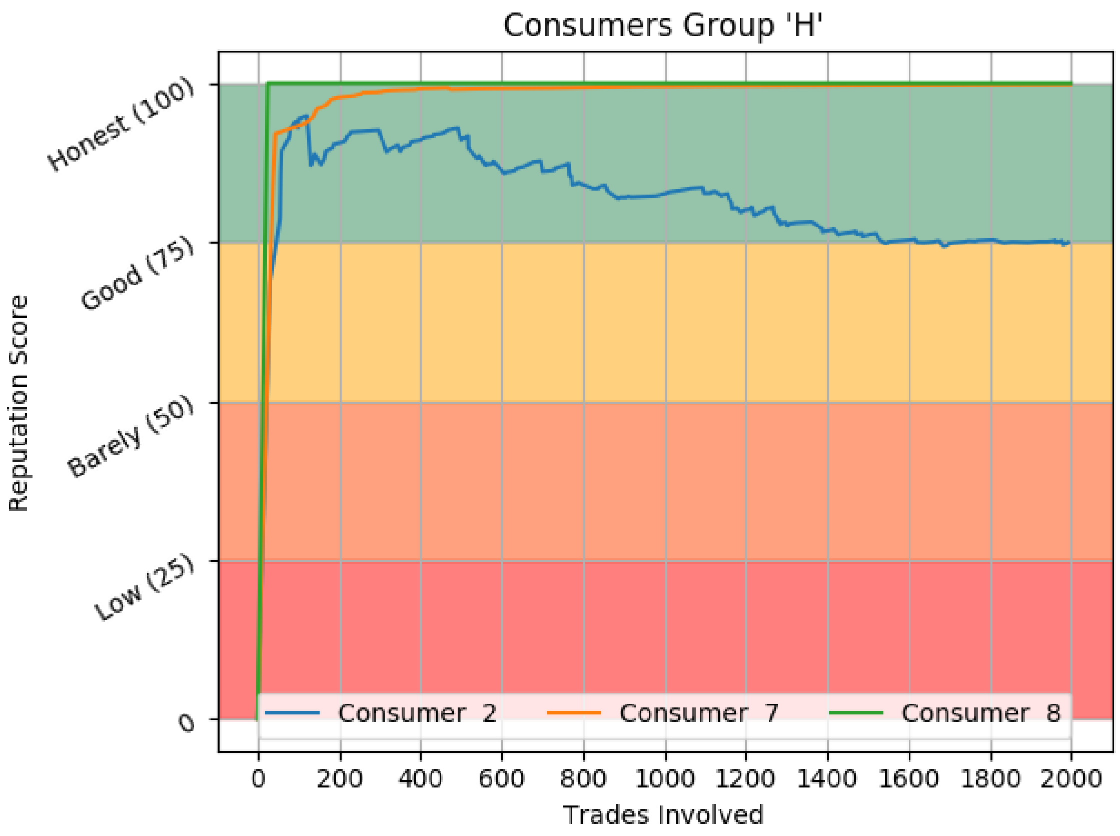
Id	*TrV*					AVG *Pnt*	AVG CS *TATI%*	End Rep	Max Rep	Min Rep
	Total	Success	Failed							
			PS	CS	Dishonest Behaviour					
0	174	84	86	4	0	0	93.55	100	100	99.68
1	231	44	90	44	53	27.53	46.07	75.72	91.73	36.55
2	195	81	39	40	35	29.98	44.55	74.92	94.87	68.67
3	229	137	24	32	36	28.55	66.61	83.71	84.63	41.39
4	182	100	56	9	17	15.96	43.92	97.87	99.53	64.78
5	194	50	98	29	17	44.31	50.42	83.12	91.22	53.41
6	185	144	41	0	0	0	0	100	100	100
7	196	88	98	10	0	0	48.69	99.76	99.76	68.19
8	205	71	134	0	0	0	0	100	100	100
9	209	173	11	15	10	24.09	57.32	97.27	99.63	70.76

Table 6.6 shows all *ten* consumers' behaviours and the logs of trades in which they were involved. Similar to producers, every consumer was picked randomly to be involved in trades for 2000 trade simulations, and that resulted in a different number of trades for each consumer.

TrV represents all trades in which a consumer was involved. All total trades either end successfully or in failure. As mentioned earlier, a trade fails if a producer stops *PS*, a consumer stops *CS*, or fraudulent behaviour is detected, represented by the column *dishonest behaviour* in Table 6.6. *AVG Pnt* presents the average value of *Pnt* that a consumer incurred in all his failed trades. The next column, *AVG CS*, represents the average trade completion before stopping trades. It shows how much time is taken from *TATI* before a consumer decides to stop. The last two columns show the minimum and maximum reputation scores for each consumer, while the current *Rep* in the marketplace is denoted by *End Rep*.

Fig. 6.4 Reputation Scores of all consumers through their trades

(a) Profile for Honest Consumer



(b) Profile for Dishonest Consumer

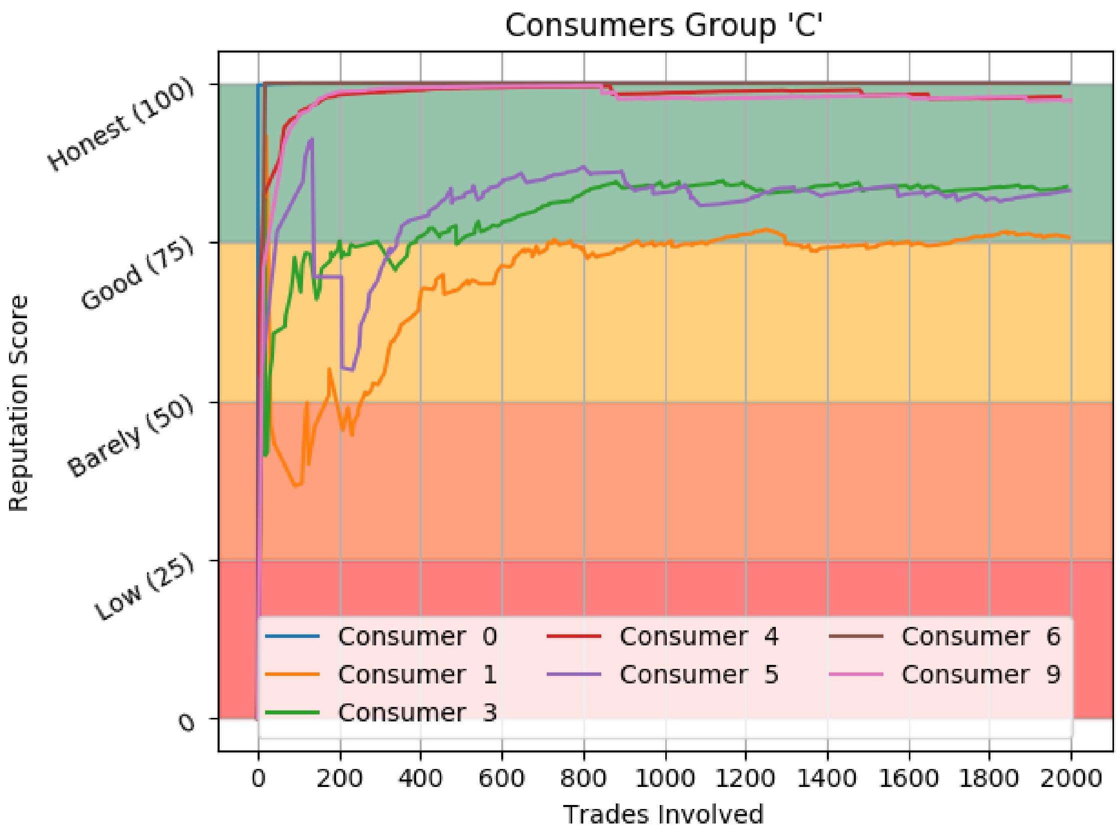


Table 6.6 shows the highest value of consumer reputation given to C_0, C_6 and C_8. C_6 and C_8 reflect the optimal consumer behaviours, with no stops, successful trades and receipts for all the data received. However, C_0 achieved the highest value, with 4 trades stopped. This case shows what is expected from building up the consumer reputation due to parameter K, which moved the score from 99.68 to 100 with more successful trades.

C_7 withdrew and stopped 10 trades, only completing 48.69% of theses trades, his reputation dropped to 68.19. However, it was built gradually through 88 successful trades that, moving the score up due to parameter K, as expected.

C_1 and C_2 are good examples of quick moves up or drop down in reputation due to parameter K, with a marked decrease in Rep^{C_2}, and a marked increase in Rep^{C_1}, as shown in Fig. 6.4. As shown in Table 6.6, they have quit similar scores, with 75.72 and 74.92 for C_1 and C_2, respectively, although they have different TrV. C_1 had the most fraudulent behaviour, and in addition, the highest number of stops CS. However, he built up his reputation again gradually with a number of successful trades, which took his reputation to the honest band. C_2, on the other hand, started with a number of successful trades, and then exhibited a dishonest attitude, with 35 failed trades, incurring a penalty of 29.98.

C_3 and C_5 ended with the same reputation of 83. They showed the trade-offs between the reputation and the elements: dishonest behaviours and trade stops.

C_3 had double the dishonest behaviours of C_5, but his Pnt was lower. Lower Pnt means a lower number of unreported messages. Similarly, C_3 had more stops than C_5, completing 66.61% of trades, compared with C_3 who withdrew and stopped after 50.41% of trades. The same case is shown between C_4 and C_9.

This evaluation showed the expected trade-offs between reputations and the elements that affect reputation. A consumer should control the balance between these elements in order attain a high reputation. This also shows optimal behaviour, with no stops, no fraudulent behaviours, and the absence of penalties.

6.7.2 Reputation and Trade Cost Evaluation

The testbed of this evaluation is the marketplace back-end development, which is represented by the smart contract SC. It is deployed on a private Ethereum test network, which is connected via ganache [102]. We used Ethereum's web-based IDE Remix [103] to write, deploy and connect to the private chain through Remote Producer Calls. We used fake accounts with balances provided by Remix as trade participants.

Here we experimentally determined the costs associated with each phase of the $P-C$ interaction through SC. To recall, SC and thus gas fees are involved in registering new participants, deploying new offers, and creating new trade agreements. Once the agreement is

Table 6.7 Transactions cost in each cost category (in gas)

Cost Category	**Operation**	**Producer Gas Consumption**	**Consumer Gas Consumption**
Registration Cost	- Register in the network	204739 gas	199093 gas
Offering Cost	- Deploy an offer	491862 gas	-
Setup Cost	- Make an order and create a new TA	-	620865 gas
	- Set Batch size and sign off the TA	82063 gas	-
Receipt Cost	- Send a receipt	-	144367 gas

in place, *C* provides a deposit *TP* based on *ETM*, as suggested earlier in the model (Chapter.4). Importantly, we assessed the cost *RC* in accordance with the data receipt protocol.

Table 6.7 shows the costs broken down per phase, incurred by *P*, *C*, or both. We have measured the gas consumption using the Remix debugger [103], which provides consumed gas for every transaction. This can also be obtained by monitoring the balances of participants and checking the differences before and after invoking the smart contract method.

The settlement is made by the settlement smart contract when the trade ends.

As the unit gas price *GUP* largely determines the duration of the transactions in the blockchain to be confirmed by miners, a consumer has the option to increase the *GUP* in order to process their transaction faster, and therefore he will have more time out of the total *TATI*, to receive more batches. To clarify, in the Ethereum network, higher *GUP* gives the *SC* transaction priority, as miners who validate transactions usually follow the strategy of picking transactions with higher *GUP* to be included in the next block. Thus, the increase in *GUP* contributes to decreasing *RT*, and therefore provides a larger *ATM* (actual total messages) within the *TATI* interval.

The minimum and maximum *GUP* in the network can be found using the *ETH Gas Station* [89]. This is a tool used to understand the conditions of the current gas market and current policies of network miners. Based on the current condition of the network at the moment of writing, the recommended gas prices from the gas station are shown in Table 6.8. The Table shows the maximum time taken by miners to confirm the transaction for each

GUP. In addition, the gas station provides the median time of transaction confirmation for each *GUP*.

Table 6.8 Gas prices in Gwei and their speeds for trade cost evaluation

Gas Price (Gwei)	**Speed**	**Median Speed**
1.6	SafeLow (<30 m)	4.3 m
4.2	Standard (<5 m)	2.5 m
7.2	Fast (<2 m)	0.8 m

For the purpose of evaluation, we have used the three different *GUP* for different consumer reputations to calculate the *RC*, as explained in Sec.4.2.6.

Table 6.9 The RC in USD and data percentage delivered for consumer reputations for $TATI = 1day$ with three GUP values, MR=100 msg/sec. (1 Eth $\approx$ 202.70 USD)

Consumer Reputation	**GUP= 1.6 Gwei**		**GUP= 4.2 Gwei**		**GUP= 7.2 Gwei**	
	Cost in USD	**Data Percentage**	**Cost in USD**	**Data Percentage**	**Cost in USD**	**Data Percentage**
10	1.465$	92.236%	3.970$	95.313%	7.016$	98.444%
20	0.903$	95.81%	2.372$	97.57%	4.277$	99.17%
30	0.716$	97.01%	1.881$	98.26%	3.224$	99.44%
40	0.623$	97.61%	1.635$	98.61%	2.802$	99.55%
50	0.529$	98.20%	1.389$	98.95%	2.381$	99.66%
60	0.482$	98.51%	1.266$	99.13%	2.170$	99.72%
70	0.482$	98.51%	1.266$	99.13%	2.170$	99.72%
80	0.435$	98.81%	1.143$	99.30%	1.960$	99.78%
90	0.435$	98.81%	1.143$	99.31%	1.960$	99.78%
100	0.388$	99.10%	1.020$	99.48%	1.749$	99.83%

Fig.6.5(a) shows the number of smart contract invocations, that is, the number of receipts vs consumer reputation. The cost of these invocations is depicted in Fig.6.5 (b).

Note that the maximum data delivery figures when $Rep^C = 100$ for $GUP = 1.6$ Gwei, 4.2 Gwei and 7.2 Gwei are 99.10%, 99.48%, and 99.83%, respectively. If we assume that the minimum time for a transaction to be confirmed is about 1 second, the maximum number of messages that could be delivered to a consumer is $\leq (ETM - MR * BN)$. Recall that the overhead due to processing the receipts, represented as *RT*, is included in *TATI*, which reduces *ETM* by $RT \cdot BN$.

Although the numbers of receipts as shown in Fig. 6.5 (a) are nearly the same, which led to similar costs for receipts as shown in Fig. 6.5 (b), the fraction of data received, *ATM*, is higher for a higher reputation consumer as expected, as shown in Fig. 6.5 (c).

6.7.3 Transaction Latency Evaluation

The testbed is our Decentralised Application (DAPP), which consists of: (i) the back-end development which is represented by the smart contract and deployed on the Mainnet Ethereum network, and (ii) the front-end development, which was developed by *Python* and *Web3*.

We used Infura [104], which is a cluster hosting an Ethereum node. It is used to interact with our deployed marketplace smart contracts. We used a MetaMask account [105], which has a real Ether currency to deploy the smart contract on the Ethereum network as a back-end development.

Here, we experimentally show the latency of transactions for the Ethereum network to process and confirm our marketplace transactions. Note that here we are only concerned with the confirmation time of processing the transaction of sending receipt *RT*, while the other marketplace transactions are not a concern in this evaluation. The latency we care about is *RT*, where the fast response time means more time out of *TATI* to receive more message batches.

We are recording the best response time our system can achieve for an individual trade, and then evaluating whether this latency is affected by a high workload number of concurrent trades or within long-term contracts.

As the unit gas price *GUP* largely determines the duration of *RT* that is taken to be validated and confirmed by miners, we created a baseline for our experiment with different *GUP* to determine the minimum *RT* our system could achieve while taking into consideration the network status and gas market.

At the current moment of writing, the minimum and maximum *GUP* in the network, as stated in *ETH Gas Station* [89], are explained in Table 6.10. The table shows the maximum time taken by miners to confirm the transaction for each *GUP*. In addition, the gas station provides the median time of transaction confirmation for each *GUP*.

In our evaluation benchmark, we had a baseline with different *GUP* as shown in Table 6.10, for an individual trade with 30 receipts, and recorded the minimum *RT* possible.

As shown in Figure 6.6, the *RT* of the individual trade in Figure (a) with $GUP = 20$ was between a minimum time of between 21 sec and 150 sec, which is quit a long time to postpone sending data and wait to process the receipt. While, on the other hand, Figure (b)

Table 6.10 Gas prices in Gwei and their speeds for transaction latency evaluation

Gas Price (Gwei)	**Speed**	**Median Speed**
20	SafeLow (<3.5 m)	1.3 m
45	Standard (<2 m)	0.8 m
70	Fast (<1.1 m)	0.3 m

with $GUP = 45$ achieved a quick response of between 10 sec and 87 sec. With the increase in *GUP* to 70 as shown in Figure (c), the best *RT* achieved was between 9 sec and 38 sec.

Based on the current *GUP*, *the average RT* figures for gas prices $GUP = 20$, $GUP = 45$ and $GUP = 70$ are 80 sec, 53 sec and 23 sec, respectively, as shown in Figure 6.7. This shows that the minimal acceptable *GUP* a consumer can pay for a lower *RT* is $GUP = 70$. To clarify, the increase in *GUP* contributes to decreasing *RT*, and therefore better use of the *TATI* interval to receive more messages.

Since the focus of our experiment is the relationship between the *RT* of one individual trade and the minimal latency possible, based on these experiments, we have chosen $GUP = 70$ as a abaseline for our benchmark, as this provides a good speed range of *RT* based on the current conditions of the Ethereum network and gas market.

Table 6.11 *RT* in the baseline with *GUP*=70

	Speed
Minimum	9 sec (0.15 m)
Maximum	38 sec (0.6 m)
Median	23 sec (0.3 m)

With $GUP = 70$, we have run 100 concurrent trades with 30 receipt transactions for each, and calculated the avg(*RT*) of every trade.

Figure 6.8 (a) shows that one of the avg(*RT*) of concurrent trades is less than the baseline, at 3 sec faster, while the maximum is 33 sec. Comparing this to the average *RT* of one individual trade as shown in Figure (b), *RT* is affected by high workload number of concurrent 100 trades, such that it is increased by 3 sec from $RT = 23$ sec to $RT = 26$ sec. Although *RT* in concurrent trades has deviated 3 sec from the expected *RT* of the baseline, it is still within the speed range expected for $GUP = 70$, as explained in Table 6.11.

As the response time is affected by how much gas a consumer is willing to pay, the network is attempting to confirm the transactions within the speed range specified in *ETH*

Gas Station [89], as these transactions are tagged as a high priority, and they are picked by miners to be validated. However, the results could be slightly different from one time to another, based on the network conditions and gas market.

6.8 Discussion and Conclusion

6.8.1 Technical Issues

Through the phases of modelling and designing the marketplace, we faced a number of technical hurdles, especially while implementing the marketplace smart contract that lives on the blockchain.

Limitation in Solidity

Solidity is the language that we used to program the smart contract. Programming a smart contract with Solidity is quite different from the nature of the programming for other high-level languages such as Java, C++, and Python, for example, regarding its limitations, *decimal number representation*. It follows a different approach in number representation, where it does not support the decimal numbers, but rather uses four integer digits. To publish and store a decimal number on the blockchain, we need to represent the two integer digits and the two decimal digits as *four* integer digits. Suppose we have a reputation for a participant with value 34.21, it is stored on the blockchain as an one integer with a value of 3421. Therefore, to retrieve any decimal number from the blockchain through the front-end implementation, it should be divided by 100 and multiplied by 100 to be stored again on the blockchain.

Lack of Debugging Tools

Although Remix [103] and Truffle [106] provide a contracts debugger, they debug at the assembly and transactions level, which is too low to debug our smart contract. Instead, we created Events that log data along with the contract to help in tracing variable states after transaction executions.

Slowness in Events Logs in Infura

As Infura does not support listening to events with *HTTPProvider* on Ropsten testnet, we had to use *WebsocketProvider*. Logs and events were sometimes recorded very slowly which hindered the events processed by the front-end, and did not reflect the time that the events had been triggered. Using the same socket connection with Python threading programming in the front-end implementation, it is quite difficult to handle and process many events that are triggered by the smart contract on Ropsten testnet.

Listening to the events of the smart contract in the local network *ganache* [102] is easy, and it immediately logs every transaction and event once triggered. This was difficult both to move to Ropsten testnet and to adapt the code for.

Time Consumption in Synchronising the Local Geth Node

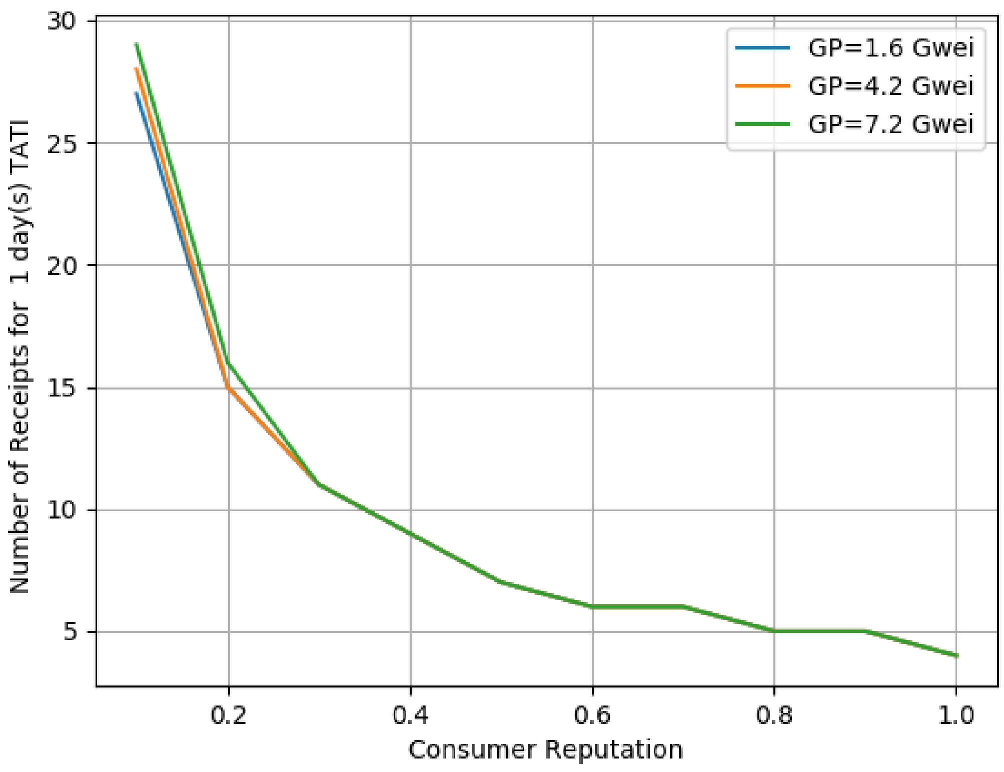
We consumed too much time to synchronise all blockchain records on our machine, rather than using the cluster Infura, so our machine is capable of connecting via *WebsocketProvider* from our local host.

6.9 Conclusion

In this chapter, we have introduced the technical implementation of the marketplace model and its reputation system. It has been explained technically how the model was structured and programmed in both its layers: the data transfer and the blockchain layer. A simulator has been presented, simulating how the model works with the line of scoring participants in different scenarios. The entire model has been evaluated at different implementation levels, with each using different parameters: evaluating the simulator, the trade-off between overhead trade cost and given consumer reputation, and, finally, transaction latency, which shows the time taken for transaction confirmation based on different gas prices in the gas market and the network conditions.

Fig. 6.5 The cost, the number of receipts and the percentage of the data received for three different gas prices.

(a) Number of Receipts



(b) Cost in USD

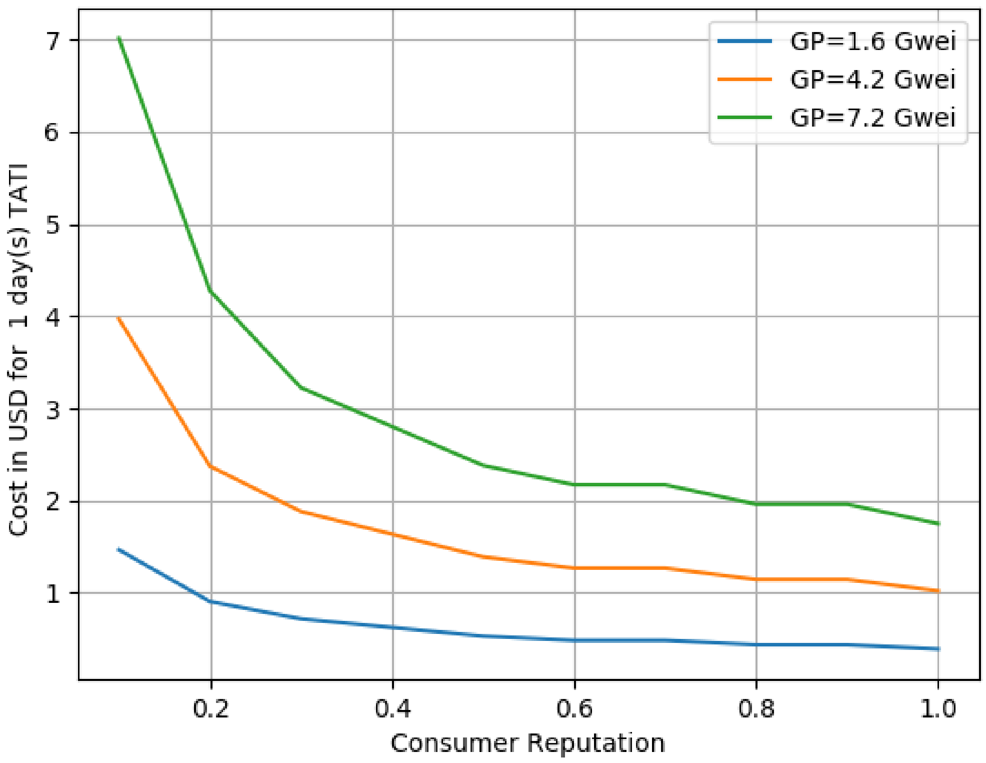


Fig. 6.5 The cost, the number of receipts and the percentage of the data received for three different gas prices.

(c) Data Percentage Received

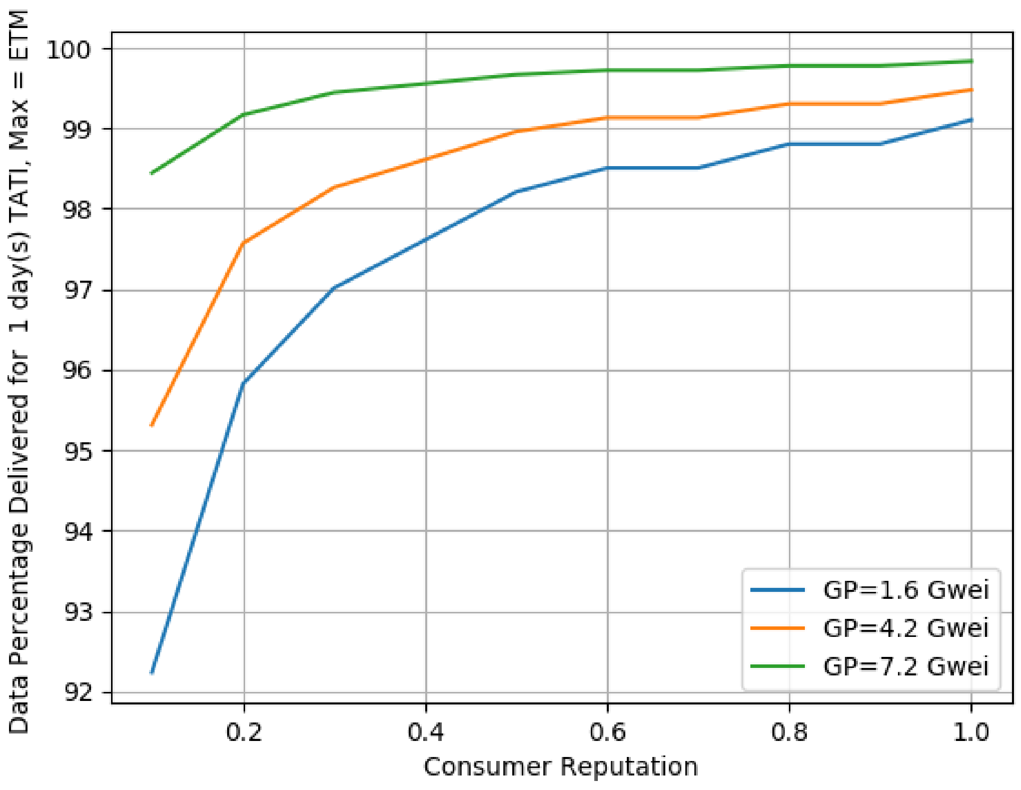
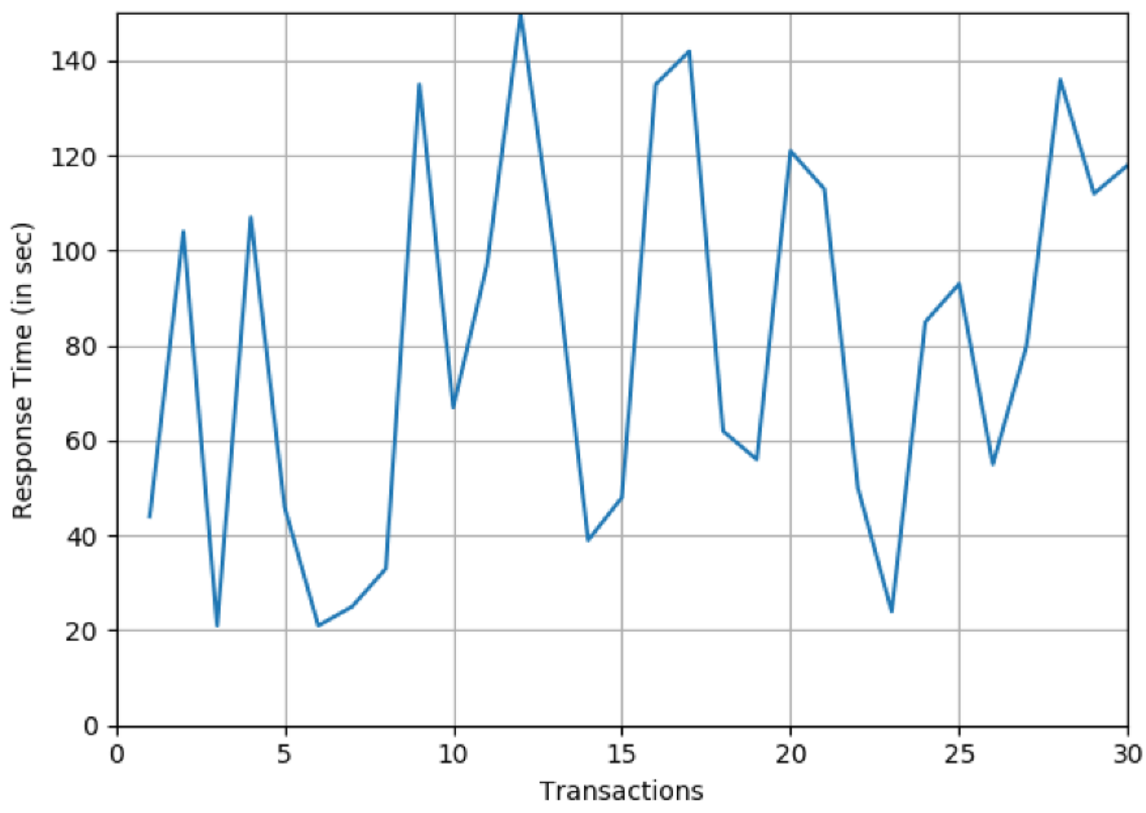


Fig. 6.6 The *RT* speed through 30 receipt transactions with different gas prices

(a) *RT* with *GUP*=20



(b) *RT* with *GUP*=45

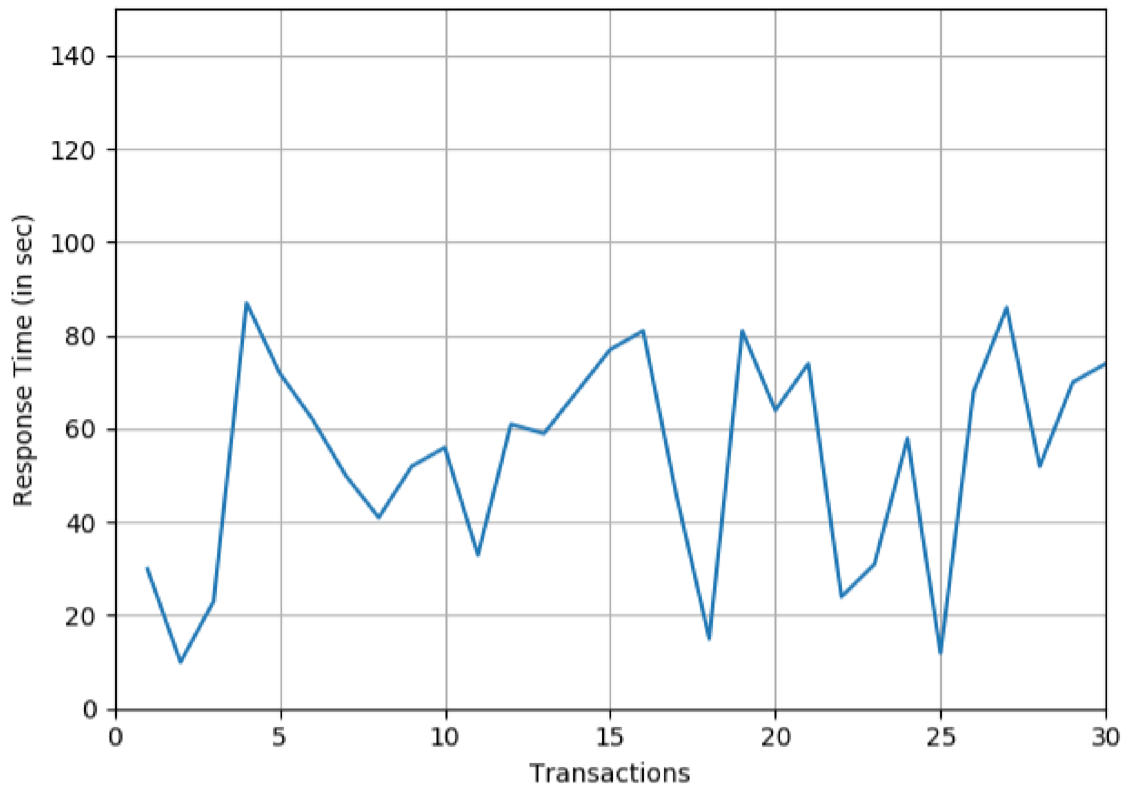


Fig. 6.6 The *RT* speed through 30 receipt transactions with different gas prices

(c) *RT* with *GUP*=70

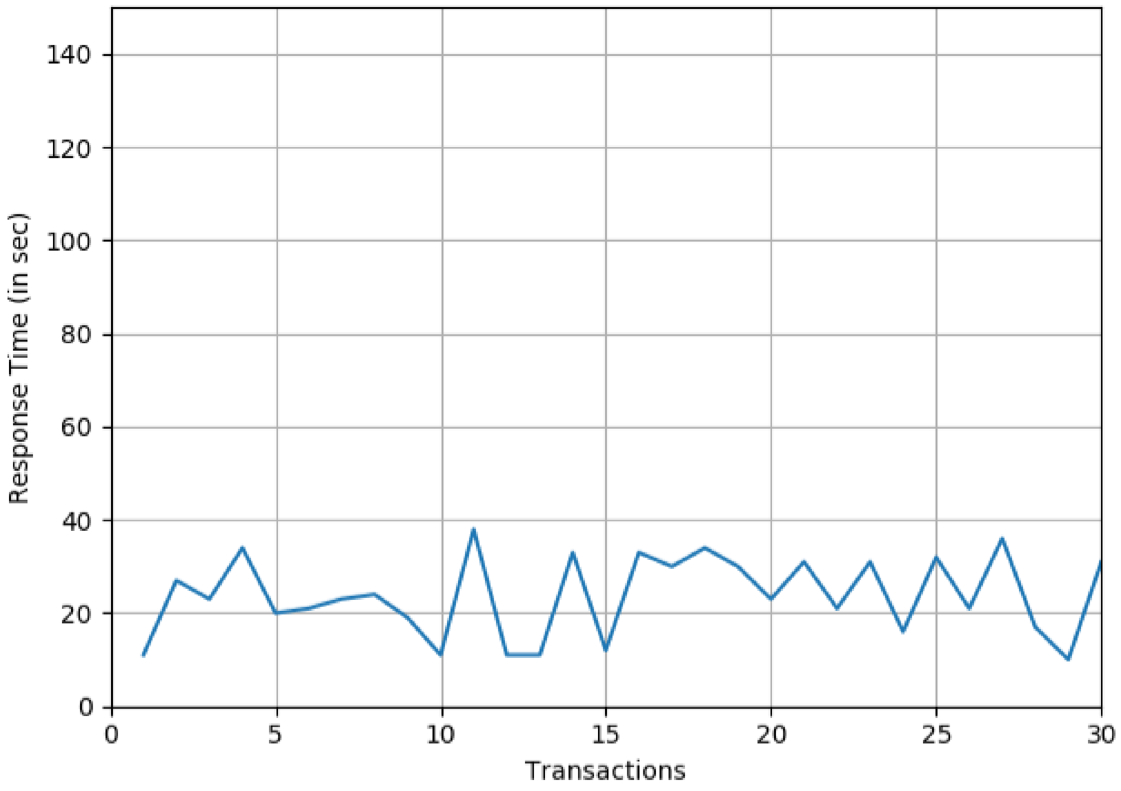


Fig. 6.7 The average *RT* of 30 receipt transactions in every gas price, 20, 45, and 70

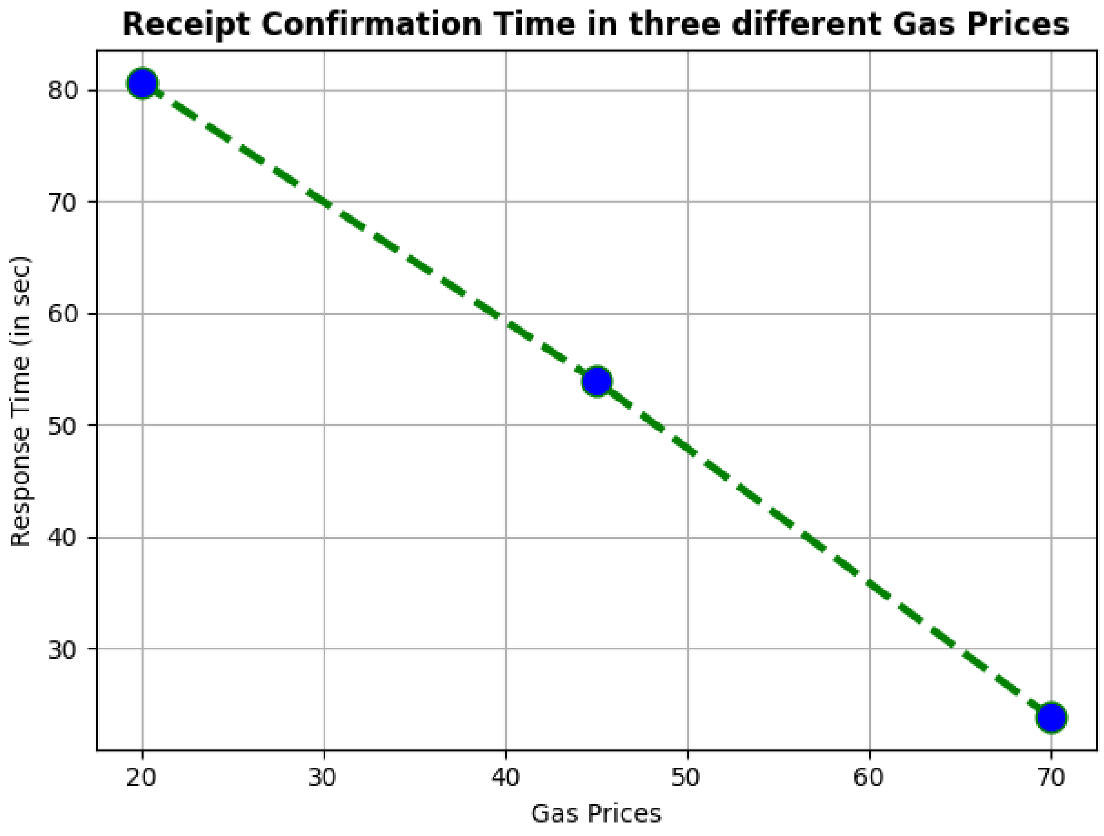
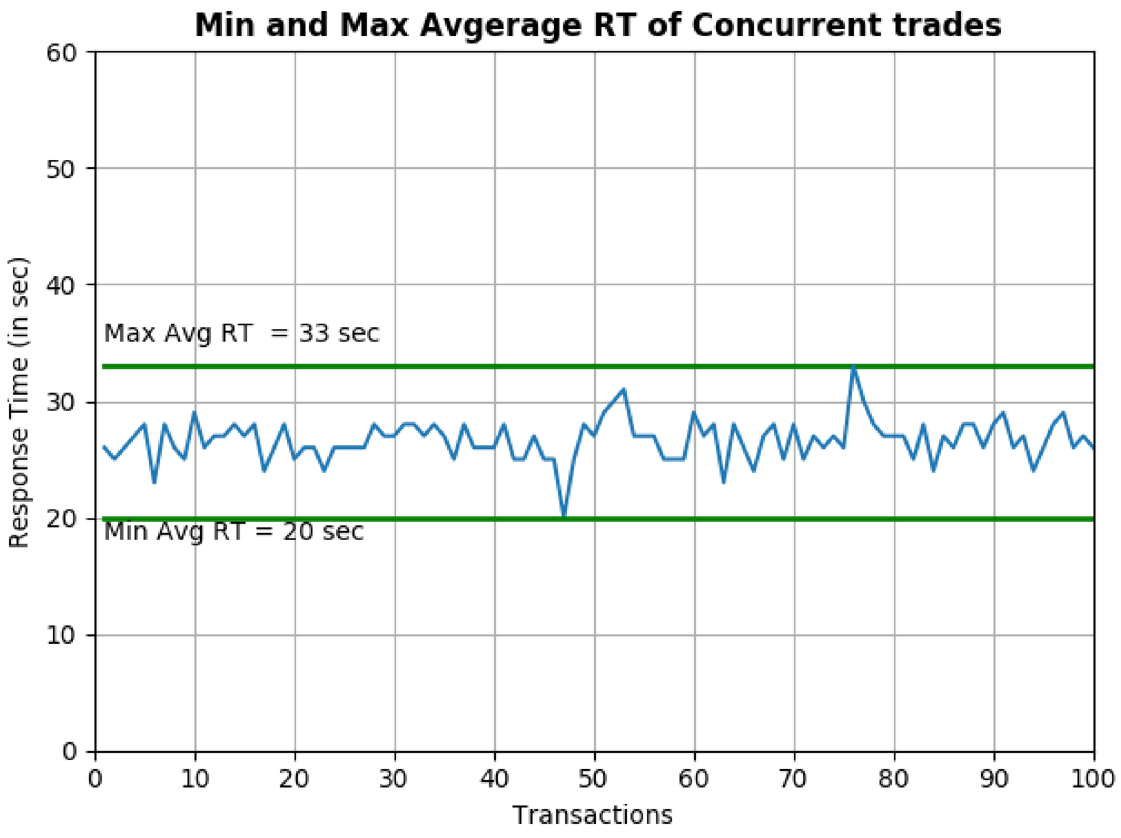
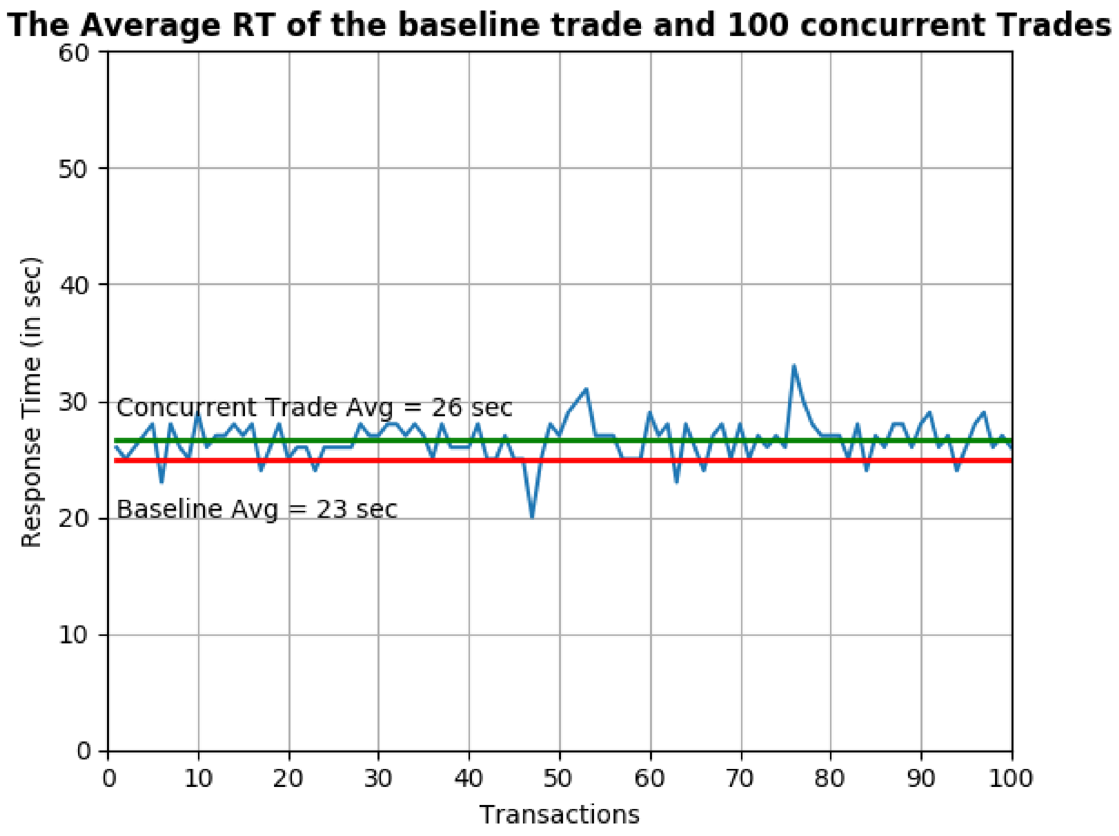


Fig. 6.8 Comparison between the *RT* of the baseline and the concurrent trades.

(a) Min and max avg(*RT*) of 100 concurrent trades



(b) *RT* comparison between the baseline and the concurrent trades



Chapter 7

Discussion and Conclusion

This work has introduced a decentralised marketplace for trading brokered IoT data with no mutual trust amongst participants. We first introduced the protocol of the receipt-based marketplace, which relies on frequent checking points that allow a consumer to send a receipt to acknowledge what he has received. While the model assumes trust in producer delivery, fraudulent behaviour may be expected by consumers. The use of blockchain smart contracts helps to record immutably every receipt sent, as the evidence of consumer trust, in addition to other criteria set based on the reputation system introduced. We then introduced our reputation system, which is tailored to address the reputation management of the marketplace participants.

With no monetary incentives, we have provided our reputation system, which quantifies every participant's trustworthiness at the lowest monetary and storage costs. We have proposed a model that quantifies participants' trustworthiness based on their activities and trade involvement. It only allows participants' scores to be stored on blockchain in a numerical representation, with the absence of textual reviews. We then evaluated this integrated marketplace with the reputation system with our simulator, which showed what we expected.

Based on our experiments, participants' scores - both producers and consumers - fluctuated from high scores to low scores based on their behaviours. Through the experiment, we showed that, in the long run producer reputation criteria used to score producers' attitudes are sufficiently fair to measure their trust in the marketplace'. A producer's behaviour is tracked based on his commitments to current trades, with the absence of withdrawals. Quantifying producer trust with the assumption of delivery trust was a challenge in our model, in which it is measured from different angles. The simulator showed, as expected, that a producer's reputation dropped down and built up again based on his data quality, as rated by consumers, frequent stopping of trades, and multiple disregarded requests.

Alongside the producer, un-trusted consumers can build their trust again with more successful trades, with the absence of withdrawals, and honesty in receipt acknowledgement. The model makes a trade-off between the trade overhead cost, and consumer trust in the marketplace, empowering consumers to negotiate to make a cost balance based on their current reputation. We have shown experimentally that, if a reputation score can be given by our reputation system, then the trade-off can be easily quantified, making the trading overhead cost manageable.

Moreover, we showed experimentally on Ethereum Mainnet the response time that our system can achieve with a high workload number of concurrent trades. The transaction latency, which shows the time taken for transaction confirmation, is based on different gas prices in the gas market and the network conditions. The more gas incurred, the higher the priority of the transaction to be picked by miners to be verified, and therefore confirmed.

7.1 Limitations

This thesis has concentrated on introducing our entire model and its implementation of a decentralised trustless brokered IoT data marketplace on the Ethereum blockchain. It has shown the reputation system designed for the marketplace model, fitting its needs to quantify participants' trustworthiness.

The introduced reputation system can work with other systems that have the same request-response protocol for their marketplace model, regardless of how a trade takes place. It would fit the needs of marketplaces that have the same criteria in quantifying their participants' trustworthiness, where this relies on the participants' expected behaviour more than the way in which a trade is executed.

However, due to the time constraints involved in producing this thesis, the marketplace limitations have the higher priority in our future plan to be addressed. One of the first system limitations is ***testing the system***. We have not tested the system extensively with real producers, real consumers and real-time streams. While our system has passed through various phases of testing, it still needs to be extensively tested with real-time streams, and real, different producers and consumers. We tested the system first in the local network *ganache*, and then moved to the *Ropsten* testnet and traced the communications between the smart contract on Ethereum and the front-end, which was implemented with Python and the library *Web3.py*. Moreover, it has been tested on Ethereum Mainnet, with a simulator for the number of trades between a few producers and consumers, and a real-time stream dataset from the Urban Observatory project [100].

Usability of Front-end development: Although we implemented the front-end of our marketplace with Python, it was implemented for the initial marketplace functionalities. It did not follow any designs or layout principles and it lacks user interface usability.

7.2 Future Work

The future work will focus on continuing to work on improving and enhancing the model functionalities and reputation system reliability through extensive testing. Executing many tests such as pilot tests provides the opportunity for a small-scale study of the marketplace with real participants before conducting the actual one. It is one way to allow a small group of real producers and consumers to test the marketplace functionalities and give a feedback for enhancing the model.

However, before all of that, we will examine in depth the usability of the front-end development and quantify the users' experience, including the system participants' perceptions of utility, ease of use, and efficiency.

References

[1] Zibin Zheng, Shaoan Xie, Hongning Dai, Xiangping Chen, and H. Wang. Blockchain challenges and opportunities: a survey. *Int. J. Web Grid Serv.*, 14:352–375, 2018.

[2] Paolo Missier, Shaimaa Bajoudah, Angelo Capossele, Andrea Gaglione, and Michele Nati. Mind my value: A decentralized infrastructure for fair and trusted iot data trading. In *Proceedings of the Seventh International Conference on the Internet of Things*, IoT '17, pages 15:1–15:8, New York, NY, USA, 2017. ACM.

[3] Charith Perera, Arkady Zaslavsky, Peter Christen, and Dimitrios Georgakopoulos. Context aware computing for the internet of things: A survey. *IEEE communications surveys & tutorials*, 16(1):414–454, 2014.

[4] Kresimir Misura and Mario Zagar. Internet of things cloud mediator platform. In *Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2014 37th International Convention on*, pages 1052–1056. IEEE, 2014.

[5] Fabian Schomm, Florian Stahl, and Gottfried Vossen. Marketplaces for Data: An Initial Survey. *SIGMOD Rec.*, 42(1):15–26, 2013.

[6] Tien-Dung Cao, Tran-Vu Pham, Quang-Hieu Vu, Hong-Linh Truong, Duc-Hung Le, and Schahram Dustdar. Marsa: A marketplace for realtime human sensing data. *ACM Trans. Internet Technol.*, 16(3):16:1–16:21, May 2016.

[7] C. Perera, C. Liu, and S. Jayawardena. The emerging internet of things marketplace from an industrial perspective: A survey. *IEEE Transactions on Emerging Topics in Computing*, 3(04):585–598, oct 2015.

[8] Lara Vomfell, Florian Stahl, Fabian Schomm, and Gottfried Vossen. A classification framework for data marketplaces. Ercis working paper, 2015.

[9] Xiang LIAN Lei CHEN, Mitchell TSENG. Development of foundation models for internet of things. *Frontiers of Computer Science*, 4(3), 2010.

[10] S M R Islam, D Kwak, M H Kabir, M Hossain, and K S Kwak. The Internet of Things for Health Care: A Comprehensive Survey. *IEEE Access*, 3:678–708, 2015.

[11] Charith Perera, Arkady Zaslavsky, Peter Christen, and Dimitrios Georgakopoulos. Sensing as a service model for smart cities supported by Internet of Things. *Transactions on Emerging Telecommunications Technologies*, 25(1):81–93, jan 2014.

[12] Datum, Blockchain Data Storage an Monetization,https://datum.org.

[13] Datapace Marketplace, https://datapace.io.

[14] MQTT Protocol, https://mqtt.org.

[15] Audun Jøsang, Roslan Ismail, and Colin Boyd. A survey of trust and reputation systems for online service provision. *Decision support systems*, 43(2):618–644, 2007.

[16] Erick Owiyo, Yong Wang, Eunice Asamoah, Domnic Kamenyi, and Isaac Obiri. Decentralized privacy preserving reputation system. In *2018 IEEE Third International Conference on Data Science in Cyberspace (DSC)*, pages 665–672. IEEE, 2018.

[17] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008. https://bitcoin.org/bitcoin.pdf,.

[18] Stuart Haber and W. Scott Stornetta. How to time-stamp a digital document. *J. Cryptol.*, 3(2):99–111, January 1991.

[19] Melanie Swan. *Blockchain : blueprint for a new economy*. O'Reilly Media, 2015.

[20] Chris Dannen. *Introducing Ethereum and Solidity: Foundations of Cryptocurrency and Blockchain Programming for Beginners*. Apress, USA, 1st edition, 2017.

[21] F. Tschorsch and B. Scheuermann. Bitcoin and beyond: A technical survey on decentralized digital currencies. *IEEE Communications Surveys Tutorials*, 18(3):2084–2123, 2016.

[22] Dylan Yaga, Peter Mell, Nik Roby, and Karen Scarfone. Blockchain technology overview. *arXiv preprint arXiv:1906.11078*, 2019.

[23] George O. Strawn. BLOCKCHAIN. *IT Prof.*, 21(1):91–92, 2019.

[24] S. Kim and G.C. Deka. *Advanced Applications of Blockchain Technology*. Studies in Big Data. Springer Singapore, 2019.

[25] Guang Chen, Bing Xu, Manli Lu, and Nian-Shing Chen. Exploring blockchain technology and its potential applications for education. *Smart Learning Environments*, 5, 2018.

[26] Jianjun Sun, Jiaqi Yan, and Kem ZK Zhang. Blockchain-based sharing services: What blockchain technology can contribute to smart cities. *Financial Innovation*, 2(1):1–9, 2016.

[27] The economist: The trust machine | the economist, http://www.economist.com/news/leaders/21677198-technology-behind-bitcoin-could-transform-how-economy-works-trust-machine, 2015, 2015.

[28] Roman Beck, Jacob Stenum Czepluch, Nikolaj Lollike, and Simon Malone. Blockchain–the gateway to trust-free cryptographic transactions. 2016.

[29] Stefan Seebacher and Ronny Schüritz. Blockchain technology as an enabler of service systems: A structured literature review. In *International Conference on Exploring Services Science*, pages 12–23. Springer, 2017.

[30] Svein Ølnes. Beyond bitcoin enabling smart government using blockchain technology. In *International conference on electronic government*, pages 253–264. Springer, 2016.

[31] Klaus Schwab. *The Fourth Industrial Revolution*. New York: Croen Business, 2016.

[32] Umair Khan, Zhang Yong An, and Azhar Imran. A blockchain ethereum technology-enabled digital content: Development of trading and sharing economy data. *IEEE Access*, 8:217045–217056, 2020.

[33] Andranik Tumasjan and Theodor Beutel. Blockchain-based decentralized business models in the sharing economy: A technology adoption perspective. In *Business transformation through blockchain*, pages 77–120. Springer, 2019.

[34] Primavera De Filippi. What blockchain means for the sharing economy. *Harvard Business Review*, 15, 2017.

[35] Valentina Gatteschi, Fabrizio Lamberti, and Claudio Demartini. Blockchain technology use cases. In *Advanced Applications of Blockchain Technology*. Springer, 2020.

[36] Daniel Kraft. Difficulty control for blockchain-based consensus systems. *Peer-to-Peer Networking and Applications*, 9(2):397–413, 2016.

[37] Alfonso Panarello, Nachiket Tapas, Giovanni Merlino, Francesco Longo, and Antonio Puliafito. *Blockchain and iot integration: A systematic survey*, volume 18. 2018.

[38] David P Fanning, Kurt; Centers. Blockchain and its coming impact on financial services. *Wiley Blackwell*, 2016.

[39] Fran Casino, Thomas K Dasaklis, and Constantinos Patsakis. A systematic literature review of blockchain-based applications: current status, classification and open issues. *Telematics and informatics*, 36:55–81, 2019.

[40] Jesse Yli-Huumo, Deokyoon Ko, Sujin Choi, Sooyong Park, and Kari Smolander. Where is current research on blockchain technology?—a systematic review. *PloS one*, 11(10):e0163477, 2016.

[41] Zibin Zheng, Shaoan Xie, Hong-Ning Dai, Xiangping Chen, and Huaimin Wang. Blockchain challenges and opportunities: A survey. *International Journal of Web and Grid Services*, 14(4):352–375, 2018.

[42] Philip Treleaven, Richard Gendal Brown, and Danny Yang. Blockchain technology in finance. *Computer*, 50(9):14–17, 2017.

[43] Sarah Underwood. Blockchain beyond bitcoin. *Communications of the ACM*, 59(11):15–17, 2016.

[44] J. Cruz, Yuichi Kaji, and N. Yanai. Rbac-sc: Role-based access control using smart contract. *IEEE Access*, 6:12240–12251, 2018.

[45] Ripple, https://ripple.com/.

[46] Stellar, https://www.stellar.org/.

[47] Ethereum, blockchain app platform, https://ethereum.org/.

[48] Don Tapscott and Alex Tapscott. *Blockchain Revolution: How the Technology Behind Bitcoin Is Changing Money, Business, and the World.* Portfolio, 2016.

[49] Solidity, https://docs.soliditylang.org/en/v0.7.4/.

[50] Cryptaur, Decentralised ethereum-based Ecosystem, https://cryptaur.com.

[51] Fysical, Decentralised Location Data Marketplace, http://fysical.org.

[52] Datawallet, https://datawallet.com.

[53] Florian Stahl, Fabian Schomm, and Gottfried Vossen. The data marketplace survey revisited. Technical report, ERCIS Working Paper, 2014.

[54] BigChainDB, The blockchain database https://www.bigchaindb.com.

[55] IPFS, InterPlanetary File System, https://ipfs.io.

[56] RepuX Marketplace, https://repux.io.

[57] Sia, Decentralised storage, https://sia.tech.

[58] Ahmed Suliman, Zainab Husain, Menatallah Abououf, Mansoor Alblooshi, and Khaled Salah. Monetization of iot data using smart contracts. *IET Networks*, 2018.

[59] Zhiqing Huang, Xiongye Su, Yanxin Zhang, Changxue Shi, Hanchen Zhang, and Luyang Xie. A decentralized solution for iot data trusted exchange based-on blockchain. In *Computer and Communications (ICCC), 2017 3rd IEEE International Conference on*, pages 1180–1184. IEEE, 2017.

[60] Lorenzo Pieri and Bogdan Djukic. Anyledger, embedded wallet for iot devices.

[61] IOTA, An Open, Feeless Data And Value Transfer Protocol, https://www.iota.org.

[62] Serguei Popov. The Tangle, https://assets.ctfassets.net/r1dr6vzfxhev/2t4uxvsIqk0EUau6g2sw0g/45eae33637ca92f85dd9f4a3a218e1ec/iota1_4_3.pdf. 2018.

[63] Streamer, Decentralized platform for real-time data, https://streamr.network.

[64] A Boukerch, Li Xu, and Khalil El-Khatib. Trust-based security for wireless ad hoc and sensor networks. *Computer Communications*, 30(11-12):2413–2427, 2007.

[65] Roberto Di Pietro, Xavier Salleras, Matteo Signorini, and Erez Waisbard. A blockchain-based trust system for the internet of things. In *Proceedings of the 23nd ACM on Symposium on Access Control Models and Technologies*, pages 77–83, 2018.

[66] T. Eder and Daniel Nachtmann. Trust and reputation in the internet of things. Sec.Uni-Passau.De, 2013.

[67] A. Abdul-Rahman and S. Hailes. Supporting trust in virtual communities. In *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences*, pages 9 pp. vol.1–, Jan 2000.

[68] Richard Dennis and Gareth Owen. Rep on the block: A next generation reputation system based on the blockchain. In *2015 10th International Conference for Internet Technology and Secured Transactions (ICITST)*, pages 131–138. IEEE, 2015.

[69] Ji-Sun Park, Taek-Young Youn, Hye-Bin Kim, Kyung-Hyune Rhee, and Sang-Uk Shin. Smart contract-based review system for an iot data marketplace. *Sensors*, 18(10):3577, 2018.

[70] K Salah, A Alfalasi, and M Alfalasi. A blockchain-based system for online consumer reviews. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 853–858. IEEE, 2019.

[71] Davide Carboni. Feedback based reputation on top of the bitcoin blockchain. *arXiv preprint arXiv:1502.01504*, 2015.

[72] Khamila Nurul Khaqqi, Janusz J Sikorski, Kunn Hadinoto, and Markus Kraft. Incorporating seller/buyer reputation-based system in blockchain-enabled emission trading application. *Applied Energy*, 209:8–19, 2018.

[73] Affan Yasin and Lin Liu. An online identity and smart contract management system. In *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*, volume 2, pages 192–198. IEEE, 2016.

[74] Mike Sharples and John Domingue. The blockchain and kudos: A distributed system for educational record, reputation and reward. In *European conference on technology enhanced learning*, pages 490–496. Springer, 2016.

[75] Amjad Aldweesh and Aad VanMoorsel. A survey about blockchain software architectures. In *32nd Annual UK Performance Engineering Workshop & Cyber Security Workshop-2016*. Newcastle University, 2016.

[76] Swarm, Decentralised Storage and Communication System, https://swarm.ethereum.org,.

[77] Salvatore Distefano, Giovanni Merlino, and Antonio Puliafito. Sensing and actuation as a service: A new development for clouds. In *2012 IEEE 11th International Symposium on Network Computing and Applications*, pages 272–275. IEEE, 2012.

[78] Prith Banerjee, Richard Friedrich, Cullen Bash, Patrick Goldsack, Bernardo Huberman, John Manley, Chandrakant Patel, Parthasarathy Ranganathan, and Alistair Veitch. Everything as a service: Powering the new information economy. *Computer*, 44(3):36–43, 2011.

[79] Mosquitto mqtt broker, https://mosquitto.org.

[80] Soumya Sen, Carlee Joe-Wong, Sangtae Ha, and Mung Chiang. Smart Data Pricing: Using Economics to Manage Network Congestion. *Commun. ACM*, 58(12):86–93, nov 2015.

[81] Chao Li, Daniel Yang Li, Gerome Miklau, and Dan Suciu. A theory of pricing private data. *ACM Trans. Database Syst.*, 39(4), December 2015.

[82] Dusit Niyato, Xiao Lu, Ping Wang, Dong In Kim, and Zhu Han. Economics of Internet of Things: an information market approach. *IEEE Wireless Communications*, 23(4):136–145, aug 2016.

[83] Dusit Niyato, Dinh Thai Hoang, Nguyen Cong Luong, Ping Wang, Dong In Kim, and Zhu Han. Smart data pricing models for the internet of things: a bundling strategy approach. *IEEE Network*, 30(2):18–25, mar 2016.

[84] ThingSpeak,Data collection for IoT projects, https://thingspeak.com.

[85] Geth, Ethereum Implementation in Go Language, https://github.com/ethereum/go-ethereum/wiki.

[86] Oraclize, Blockchain Oracle, oraclize.it.

[87] Alexander Schaub, Rémi Bazin, Omar Hasan, and Lionel Brunie. A trustless privacy-preserving reputation system. In *IFIP International Information Security and Privacy Conference*, pages 398–411. Springer, 2016.

[88] HyperLedger, Advancing business blockchain adoption through global open source collaboration, https://www.hyperledger.or.

[89] Eth Gas Station, https://ethgasstation.info.

[90] Sepandar D Kamvar, Mario T Schlosser, and Hector Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *Proceedings of the 12th international conference on World Wide Web*, pages 640–651, 2003.

[91] Paul Resnick and Richard Zeckhauser. Trust among strangers in internet transactions: Empirical analysis of ebay's reputation system. *The Economics of the Internet and E-commerce*, 11(2):23–25, 2002.

[92] Oliver E Williamson. Calculativeness, Trust, and Economic Organization. *Journal of Law and Economics*, 36(1):453–486, April 1993.

[93] Vincent Fremont and Gideon Mekonnen Jonathan. Can blockchain technology solve trust issues in industrial networks? In *17th International Conference Perspectives in Business Informatics Research (BIR 2018), Stockholm, Sweden, September 24-26, 2018*, pages 399–404. CEUR-WS. org, 2018.

[94] Shaimaa Bajoudah, Changyu Dong, and Paolo Missier. Toward a decentralized, trust-less marketplace for brokered iot data trading using blockchain. In *2019 IEEE International Conference on Blockchain (Blockchain)*, pages 339–346. IEEE, 2019.

[95] Paul Resnick, Ko Kuwabara, Richard Zeckhauser, and Eric Friedman. Reputation systems. *Commun. ACM*, 43(12):45–48, December 2000.

[96] Kaidong Wu. An empirical study of blockchain-based decentralized applications. *arXiv preprint arXiv:1902.04969*, 2019.

[97] Python, https://www.python.org.

[98] Web3 library documentation, https://web3py.readthedocs.io/en/stable/.

[99] Paho-MQTT,https://pypi.org/project/paho-mqtt/.

[100] Urban observatory, https://urbanobservatory.ac.uk.

[101] Simpy, https://simpy.readthedocs.io/en/latest/.

[102] Ganache, Local Ethereum blockchain, npmjs.com/package/ganache-cli.

[103] Remix, Ethereum IDE, https://remix.ethereum.org.

[104] Infura, Ethereum and IPFS APIs, https://infura.io.

[105] MetaMask, A crypto wallet and gateway to blockchain apps, https://metamask.io.

[106] Truffle, Ethereum development environment, https://www.trufflesuite.com.

Appendix A

The Marketplace Smart Contract

This section shows the methods of the smart contract, written in Solidity, that represents the marketplace back-end implementation.

A.1 Method 1: Register()

Every participant should register in the marketplace by calling this method with its two arguments. By calling this method successfully, the Ethereum address that make this call will be linked to this participant.

Parameter(s):

1. string nkname: It is a participant name in the marketplace in a string format. It is NOT a unique while it is a nickname for its unique Ethereum address.

2. uint code: It is a numerical integer value for participant role in the marketplace as *P* or *C*. It is only ONE of the following values:

   101: if a participant is *P*.

   202: if a participant is *C*.

Return: No Return.

Allowed to use this method: Everyone.

A.2 Method 2: Offer()

This method is used by a registered *P* to publish data offers specifications to the marketplace as a metadata, where it is visible and available for everyone. *P* invokes this method with a detailed description to data stream he is willing to sell. An event is triggered with a unique *id* given for every new published offer if the invocation executed successfully.

Parameter(s):

1. uint broker:a numerical integer value for broker id where the data will be sent through (off-chain).
2. string topic: the topic name of data stream in the broker in a string format.
3. uint rate: a numerical integer value that represent the expected streaming message rate in (messages/seconds).
4. string start: the start time of the offer time interval. It is date object represented in a string format.
5. string end: the end time of the offer time interval. It is date object represented in a string format.
6. uint up: the message unit price (the cost per message). It represents the cost of a single data unit in USD with 4 integer digits; 2 digits and 2 decimals.
7. bool valid: boolean value that describe the offer state and its availability. It is *true* if the offer is valid and available, and *false* otherwise.

Return: No Return.

Allowed to use this method: Registered *Ps* only.

A.3 Method 3: mkOrder()

It is used by a registered *C* to make a trade request by choosing one of the available published offers. It is a payable method where provides a mechanism allowing a transaction issuer (*C*) to send some funds to the smart contract. This funds is sent in the transaction body in a value field. A registered *C* invokes this method with $P's$ address who offer this data, the offer id, and the offer specifications that represent his needs. It should be noted that C must debit the cost of the data amount he requests from his account as a transaction value in the method invocation. By calling this method, C is giving his consent and fund to create a new TA with P, and it is signed by his Ethereum private key.

This new TA is added into all unconfirmed agreements while waiting for P's signature to approve this TA. In case of new C with $Rep^C = 0$, a value of BS is assigned by P.

By calling this method successfully, an event is triggered with id of the new TA in P's waiting requests to be responded.

Parameter(s):

1. uint of_id: a numerical integer value that represent the offer id of the trade request.
2. address producer: Ethereum address of P that owns the offer.
3. address C: C address (the second party of TA.
4. string top: message topic in a string format.
5. uint rate: message rate per seconds.
6. string start: the start time of the requested TA, specified by C. It is a date object represented in a string format. It should be after or the same time of the offer start time.
7. string _end: the end time of the requested TA, specified by C. It is a date object represented in a string format. It should be before or the same time of the offer end time.
8. uint up: cost per message.
9. uint TA_id: a new unique id for new requested TA_id.
10. uint BS: a numerical integer value for batch size. It is calculated and passed into the argument based on Rep^C. It is set to zero for $Rep^C = 0$.
11. TA time interval $TATI$.
12. uint ETM: a numerical integer value for the estimated total messages based on $TATI$.
13. uint GP: a numerical integer value for the gas price that is willing to pay (in Gwei).
14. uint RT: transaction confirmation time that is estimated based on GP (in seconds).

Return: No Return.

Allowed to use this method: Registered *Cs* only.

A.4 Method 4: AcceptTA()

This method lets *P* confirm the trades requests he has from their waiting requests queues. If *P* is willing to stick with this *TA*, he invokes this method with its id and set *BS* in the case of $Rep^C = 0$, otherwise, its value remain the same. In successful transaction, an event is triggered with the id of the new *TA* that informing it is confirmed and it get in-force in its start time.

Parameter(s):

1. uint TA_id: id of *TA*
2. uint BS: batch size

Return: No Return.

Allowed to use this method: Registered *Ps* only.

A.5 Method 5: RejectTA()

This is the rejection method that let *P* to reject the *TA* if he is no longer willing to stick with it. It takes only one argument which is the id of the rejected *TA*. In successful transaction, an event is triggered with the id of *TA* that informing it is rejected.

Parameter(s):

1. uint TA_id: is the id of *TA*.

Return: No Return.

Allowed to use this method: Registered *Ps* only.

A.6 Method 6: SendReceipt()

This method lets *C* receipt every data batch delivered. It allows *C* ,at the checking point, report how much data unit he received. In this method invocation, *C* must decide either continue to receive the next batch or stop. Moreover, every receipt may have a quality rate for its data batch, where *C* rates how quality data batch is. In successful transactions, an event is triggered with a the id of the *TA* and the receipt that submitted as a notification for *P* that a new receipt is published by *C*.

The marketplace smart contract will stop the trade if the message counts in the receipt is not the same as the batch size that should be delivered. Otherwise, *P* resume sending the next data batch unless *C* decides to stop.

Parameter(s):

1. uint TA_id: $TA's$ id that belongs to receipt.
2. uint Re_id: a receipt id.
3. uint MsgCount: a numerical integer value that represent the total messages delivered.
4. bool CS: a boolean value that express the decision of stop or continue. It says continue trading if it is F́alse,́ or stop trading if it is T́rue.́
5. uint Q: a numerical integer value in range [0,100], to rate the data delivered.

Return: No Return.

Allowed to use this method: Registered *Cs* only.

A.7 Method 7: settle()

This method is called by the smart contract at the end of each trade to settle the payment and ends the trade. It transfers the trade cost to $P's$ account and return the remain back to $C's$ account (if any).

Parameter(s):

1. address p: $P's$ address.
2. address c: $C's$ address.

3. uint forPro: a numerical value that shows the trade cost that must be credited to $P's$ account (in Ether).

4. uint forCon: a numerical value that shows the rest out of the deposit (if any) that must be returned back into $C's$ account (in Ether).

Return: No Return.

Allowed to use this method: The marketplace smart contract ONLY.

A.8 Method 8: conTR()

It is used by the smart contract at the end of every trade to update $C's$ profile in the marketplace. Based on the last trade outcome that C was involved, new values are given to the following variables: (i) Rep^C , (ii) total number of trades involved, (iii) total number of stops decision, and (iv) finally Pnt^C (if any).

Parameter(s):

1. uint TA_id: the $TA's$ id.

2. address c: $C's$ address.

3. uint TrV: a numerical integer value that represent the new total trades involved.

4. uint stops: a numerical integer value that represent the new total stops that made by C.

5. int rep: the new updated value of Rep^C. It should be noted that it is a signed integer which means C could have a score less than zero due to frequent failed practices.

6. uint Rz: a numerical value that represent tolerance fee that imposed to C, and must be paid to resume trading.

7. bool canTrade: a boolean value that shows $C's$ trading status. It is Ìalseíf C is suspended due to the unpaid tolerance fee and must be paid to resume trading, or it is no longer can trade. It is Ťrueótherwise.

Return: No Return.

Allowed to use this method: The marketplace smart contract ONLY.

A.9 Method 9: proTR()

It is used by the smart contract to update $P's$ profile in the marketplace. It updates $P's$ profile for the following variables: (i) Rep^P , (ii) total number of trades involved, (iii) total number of stops decision, (iv) total number of disregarded trades requests, and (v) finally the total number of rejected trades that is requested by new consumers (their first trades in the marketplace).

Parameter(s):

1. uint _TA_id: the $TA's$ id.
2. address p: $P's$ address.
3. uint TrV: a numerical integer value that represent the new total trades involved.
4. uint stops: a numerical integer value that represent the new total stops that made by P.
5. int rep: the new updated value of Rep^P. It should be noted that it is a signed integer which means P could have a score less than zero due to frequent failed practices.
6. uint Q: a numerical integer value that shows the new overall quality for P's generated data, and it is rated by C.
7. uint RNC: a numerical value that shows the new total number of rejected trades that is requested by new consumers.
8. uint UR: an integer value that represent the total number of unreported trades (disregarded requests).

Return: No Return.

Allowed to use this method: The marketplace smart contract ONLY.

A.10 Method 10: Set_PS()

It allows P to stop trading by calling this method with the TA id he would like to stop. An event is triggered with the id of the TA, as a notification to update the trade State that is no longer in-force due to $P's$ stop.

Parameter(s):

1. uint TA_id: $TA's$ id that no longer would like to continue.

Return: No Return.

Allowed to use this method: Registered *Ps* ONLY.

A.11 Method 11: permit()

This method is used only by the smart contract to re-grant the permission for a dishonest C to resume trading in the marketplace once the tolerance fee Rz is paid to the smart contract.

Parameter(s):

1. address c: $C's$ address.

Return: No Return.

Allowed to use this method: The marketplace smart contract ONLY.

A.12 Method 12: pay_Rz()

This method is used by C to pay the tolerance fee imposed on him. It is a payable function that receive the fee amount (in ether) in the transaction value. An event is triggered with $C's$ address as a notification for the smart contract, that Rz^C is paid. Consequently, the smart contract will internally call the method permit().

Parameter(s): No Parameter(s).

Return: No Return.

Allowed to use this method: Registered *Cs* ONLY.

A.13 Method 13: update_offer()

It is called to update an offer that already published. It updates one or all offer clauses where the new entered value will be replace the old one and keep the remain clauses unchanged.

Parameter(s):

1. uint id: offer's id.
2. string topic: offer's topic.
3. uint rate: data rate in seconds.
4. string start: the start time of the offer.
5. string end: the offer expiration time.
6. uint up: the price in USD for a single data unit.
7. bool valid: shows the offer availability, ́Trueíf it is available, ́Falseótherwise.

Return: No Return.

Allowed to use this method: Registered *Ps* ONLY.

A.14 Method 14: updateTA()

This method is used by the smart contract to update *TA* state. It is called if *TA* state has changed in one of the following cases: (i) created, (ii) approved and signed by both parties, (iii) rejected by *P*, (iv) In progress, (v) success, or (vi) failed.

Parameter(s):

1. uint id: *TA* id.
2. uint State:

   a numerical integer value where every state has a numerical representation as the following:

   (a) 0: Created

(b) 1: Approved.

(c) 2: Rejected.

(d) 3: Failed.

(e) 4: Success.

(f) 5: InProgress.

Return: No Return.

Allowed to use this method: The marketplace smart contract ONLY.

A.15 Method 15: setTA()

This method is used by the smart contract to finalize *TA* record where it is then stored on the blockchain.

Parameter(s):

1. uint id: the $TA's$ id.
2. bool CS: *C* stop. It is Ťrueíf yes, Ēalseótherwise.
3. bool PS:*P* stop. It is Ťrueíf yes, Ēalseótherwise.
4. uint ATM: a numerical integer value that shows the actual total data units delivered.
5. uint RM: a numerical integer value that shows the total data units reported by *C* from all his receipts.
6. uint AT: a numerical integer value that shows the actual time was taken from *TA*. It shows how long the trade period was, compared to the period stipulated in the *TA*.

Return: No Return.

Allowed to use this method: The marketplace smart contract ONLY.