Interacting Decentralized Transactional and Ledger Architecture for Mutual Credit

WP2

Iterative Architecture Requirements and Definition

Deliverable D2.2

Iterative Architecture Refinement

Horizon 2020

Project funded by the European Commission

Information and Communication Technologies

FET OPEN Launchpad Project

Grant no. 754494

Contract Number: 754494

Project Acronym: INTERLACE

Deliverable No: D2.2

Due Date: 30/04/2018

Delivery Date: 24/01/2019

Author: Paolo Dini (UH), Giuseppe Littera, Luca Carboni (SARDEX), Eduard Hirsch (SUAS)

Partners contributed: Maria Luisa Mulas (SARDEX), Thomas Heistracher (SUAS), Egon Börger (UNI PASSAU)

Made available to: Public

Versioning		
Version	**Date**	**Name, organization**
1	23/01/2018	Paolo Dini (UH)
2	11/11/2018	Paolo Dini (UH), Eduard Hirsch (SUAS), Luca Carboni (SARDEX)
3	07/01/2019	Paolo Dini (UH), Giuseppe Littera (SARDEX)
4	24/01/2019	Paolo Dini (UH), Giuseppe Littera (SARDEX)

Internal Reviewer: Giuseppe Littera (SARDEX)

Abstract

This report presents a summary of the analysis of a wide range of blockchain frameworks and technologies, comparing six of them in some detail. The mathematical framework underpinning the Stellar Consensus Protocol is retraced in detail in the Appendix. The conclusion of the analysis is that the most suitable framework for the INTERLACE transactional platform is Hyperledger Fabric. The iterative refinement of the functional specification presented in deliverable D2.1 is actually presented in deliverable D3.1.

Table of Contents

Chapter 1

Introduction

Paolo Dini and Giuseppe Littera

The main purpose of this report was originally to provide a refinement to the specification of the transactional platform as given by deliverable D2.1 [6]. That refinement work ended up being integrated in D3.1 [10], because the more urgent challenge of the architecture workpackage became to select an appropriate blockchain technology or framework for the platform. Although this process is being reported now, after the end of the project, the choice was made relatively early on, or in any case early enough for the project to be able to achieve a proof-of-concept implementation. The selection of a blockchain technology for the INTERLACE platform was a long and difficult process because of the wide variety of technologies, architectures, and protocols available; the very rapid rate of innovation in this field; and the evolving requirements of a future vision for the Sardex platform.

Clearly, although INTERLACE is only a first attempt at a blockchain-based transactional platform for the Sardex circuit, it should be as compatible as possible with the future vision of the circuit. In other words, the blockchain framework needs not only to meet certain quality expectations of the technology itself and to satisfy certain business requirements (as described in detail in D3.1), but must also be compatible with the long-term governance and company structure needs of the Sardex initiative as it scales beyond Italy and Europe. In particular, it took a long time to understand how the Sardex mutual credit system could scale to global level while remaining faithful to its principles of support for local economies and of reliance on trust between its users. It is important to realise that the support for local economies and the reliance on trust are not ideological choices. Rather, they should be regarded as a mixture of ethical choice *and* the feature of the network upon which its success is based: that is, the core of the value-generation process, which must therefore be protected.

The fundamental challenge is that trust is most easily built within small social groups and is difficult to maintain in large, long-range systems. Therefore, a global monolithic architecture and company structure could never work because it would lose its value-generating core(s). The answer can only be an architecture that is either distributed or at least hierarchical, with significant delegation and distribution of control to local circuits in local socio-cultural and economic contexts. Since the social and the business dimensions are the starting point for the requirements that lead to choice of technology and architecture definition, we adopted a circular and iterative approach that compared what was possible with what was desirable at each step, and slowly evolved the vision as our understanding of the blockchain space and of our own requirements improved. In this short report we cannot go into too much depth, but it is worth addressing what is arguably the central concept for both Sardex and the blockchain: trust.

1.1 Sardex, Trust and the Blockchain

The blockchain is often described as a 'trustless' technology, since the distribution of the validation and record-keeping function to many independent nodes, together with cryptographic algorithms, removes the need for a central authority that plays the role of record keeper and transaction orderer. According to the prevailing view in Computer Science, therefore, trust in a central authority and bilateral trust between transacting parties are substituted for reliance on a type of technology and protocol. Clearly, such an approach is particularly useful in situations where there is no trust at all between transacting parties.

By contrast, the Sardex circuit relies on trust at a fundamental level. Between the "sociological" trust discussed by Sartori and Dini [12] and the trust in the technology platform lies an "economic" type of trust. In particular, Sardex relies on, and reinforces, "thick trust".[1] Thick trust has been discussed in the literature in the business context (e.g. [15]), but for our purposes it is sufficient to define it as the combination of "Know Your Customer" with "Know Their Products". The role of the Sardex company that runs the circuit, combined with the work of the Sardex brokers, greatly reduces the social cost of trust for the SMEs who participate in the network and achieves and combines both these components, because it is able to make trust transitive: the circuit members trust the Sardex company and the electronic platform, and in the majority of cases this trust extends to bilateral trust between the transacting parties, making the Sardex circuit a particularly strong and stable trading community. One of the drawbacks of such an approach is that the communities and companies involved in the network ultimately depend on one actor to facilitate credit and trade among participants, rendering the network as a whole highly efficient yet vulnerable and not as inclusive and socially/financially adoptable as might be preferred.

Thus, Sardex interested in the blockchain for two reasons. First, as the company operations grow beyond Sardinia and Italy to other countries in Europe and beyond they will involve interactions with other circuits whose legal personality, business relationship with Sardex, and proprietary structure may vary along a range of options depending on the context and stakeholders. Therefore, from a functional and organisational point of view it may be more expedient to build in some flexibility at the level of the architecture: for example, each circuit could run a separate node of the blockchain. This could enable inter-circuit trade via agreements recorded on decentralised public ledgers, more transparency of the overall network, and regional and local clusters of SMEs – which in turn reduces informational asymmetries.

Second, this organisational flexibility requirement, which is essentially functionalist, is reinforced by the social and cultural requirement of respecting local community identity to the extent possible. This is not so much a matter of institutional or governance efficiency as a question of shared values built around reciprocal respect between different communities who identify with different regions or localities. We feel it is easier to meet such expectations with an articulated and decentralised[2] architecture than through a monolithic platform like Facebook or Google.

1.2 Overview of Report

This brief report provides a high-level discussion of the main blockchain technologies we examined during the course of the project, together with a rationale for choosing Hyperledger Fabric as the core component with possible extensions towards Ethereum and Holochain. The discussion in the rest of the report assumes familiarity with the basic concepts and terminology of the main blockchain technologies as can be found, for example, in [14].

Chapter 2 discusses a few candidate Distributed Ledger Technologies (DLTs) that we assessed in the process of deciding which satisfied the INTERLACE and Sardex requirements best. The next step in the process was going to be an ASIM specification and CoreASIM modelling of the transactional platform, which would have been reported in a third chapter, but lack of funding complementary to the INTERLACE budget made this plan impossible. We therefore decided that it made more sense to focus the remaining time and resources on implementing a proof-of-concept transactional platform based on the requirements specified in deliverable D2.1 [6] and updated in deliverable D3.1 [10]. The implementation work is reported in deliverable D3.2.

[1] Richard Simmons, economist, private communication, 2018.

[2] In contrast to what we wrote in the INTERLACE proposal, we have adopted the definition of 'decentralised' as an architecture where control is distributed, as opposed to 'distributed', which we take to mean only a distribution of functional aspects but leaves the control central.

1.3 Table of Acronyms

Table 1 shows the definition of the acronyms used in this report.

AML	Anti Money Laundering
ASM	Abstract State Machine
ASIM	Abstract State Interaction Machine
B2B	Business-to-Business
B2C	Business-to-Consumer
BFT	Byzantine Fault Tolerance
BTC	Currency symbol for Bitcoin
Dapp	Distributed App
DB	Database
DHT	Distributed Hash Table
DLT	Distributed (or Decentralised) Ledger Technology
DoS	Denial of Service
ETH	Currency symbol for Ether, the Ethereum token
EVM	Ethereum Virtual Machine
FDAS	Federated Distributed Agreement System
FPML	Financial Products Markup Language
GDPR	General Data Protection Regulation
ICO	Initial Coin Offering
JS	Javascript
JVM	Java Virtual Machine
KYC	Know Your Customer
LE	Leader Election
PBFT	Plenum Byzantine Fault Tolerance
PoA	Proof of Authority
PoS	Proof of Stake
PoW	Proof of Work
QI	Quorum Intersection
REST	Representational State Transfer
SC	Smart Contract
SCP	Stellar Consensus Protocol
SMR	State Machine Replication
SQL	Structured Query Language
SRD	Currency symbol for Sardex credits
Tx/s	Transactions per second
UML	Unified Modelling Language
UTXO	Unspent Transaction Output
XLM	Currency symbol for Lumen, the Stellar token

Table 1: **Table of acronyms used in the report**

Chapter 2

Analysis of Possible Blockchain Technologies for INTERLACE

Paolo Dini and Giuseppe Littera

2.1 Introduction

As stated in the INTERLACE proposal, we are not interested in relying on Bitcoin, due its very low performance efficiency (< 10 transactions per second, or $[Tx/s]$) and the wastefully high energy requirements of the Proof of Work (PoW) consensus protocol. Bitcoin, however, remains a useful reference point for many blockchain properties and parameters. It is therefore assumed that the reader is already familiar with the basic concepts of the Bitcoin blockchain, which are explained very well in [2].

In the next section we present a few important concepts around which design trade-offs, and in our case selection trade-offs, have been made to arrive at the architecture that at the time of this writing we feel is most suitable for INTERLACE and for the Sardex circuit, Hyperledger Fabric. A few more specific details are then presented that concern a small subset of possible Distributed Ledger Technologies (DLTs) in common use, which are summarised at the end of the chapter in a summary table. Thus, the analysis in this chapter forms the basis for the architectural design decisions described in deliverable D2.3 [7].

2.2 Basic Concepts

2.2.1 The Blockchain as a Distributed Applications Platform

It may appear that the most important innovations of Bitcoin are the creation "out of thin air" of a new currency and the substitution of a central authority for validating a ledger of transactions in this new independent currency with a distributed architecture based on a consensus protocol. These two innovations were the main motivation for creating Bitcoin, but as has been noted by many the potentially more impactful innovation is the blockchain itself. Figure 2.1 proposes a reason why by highlighting the role of the second-generation blockchains such as Ethereum as a distributed, permanent, and immutable memory in enabling a new type of distributed – yet very expensive and slow – application. The fact that this memory substrate is shared among completely unrelated applications and user groups is interesting from the point of view of the sharing economy, which could be regarded as a third innovation, but the (fourth) innovation that opens up a new space for computer science and software engineering is the fact that the memory is *distributed*, *permanent*, and *immutable*.

As shown in Figure 2.1, even allowing for block and wallet explorers,[3] the Bitcoin blockchain supports only the simplest kinds of interactions between users and the permanent, distributed, and immutable memory. Second-generation blockchains, on the other hand, have introduced what looks like a familiar stack: user interface, code, persistence layer. However, the persistence layer is writable only once. A single permanent and immutable *shared* history will remain "on the *shared* record" forever – or at least as long as there will be at least one running server hosting the blockchain. In smart contract-enabled distributed ledgers it is possible to perform also shared calculations and self-enforcing agreements based on user and/or machine or data input. Clearly, these two elements combined create a very special type of application framework that has never been possible before. If, as Edward De Bono said long ago, memory is the basic mechanism of intelligence [5], this kind of persistence layer could be the

[3] For example: http://www.bitcoin-en.com/uploads/1/0/8/6/10861361/2620769.jpg?623

start of a collective intelligence with properties and capabilities that are impossible to predict today. For the moment, therefore, we focus on the most obvious use case, a financial information system – i.e., in the INTERLACE case, the blockchain as the "mind" and memory of an additional real-economy network layer meant to complement the existing real economy. In particular, we wish to define the most appropriate properties of the financial information system and transaction engine for the Sardex circuit.

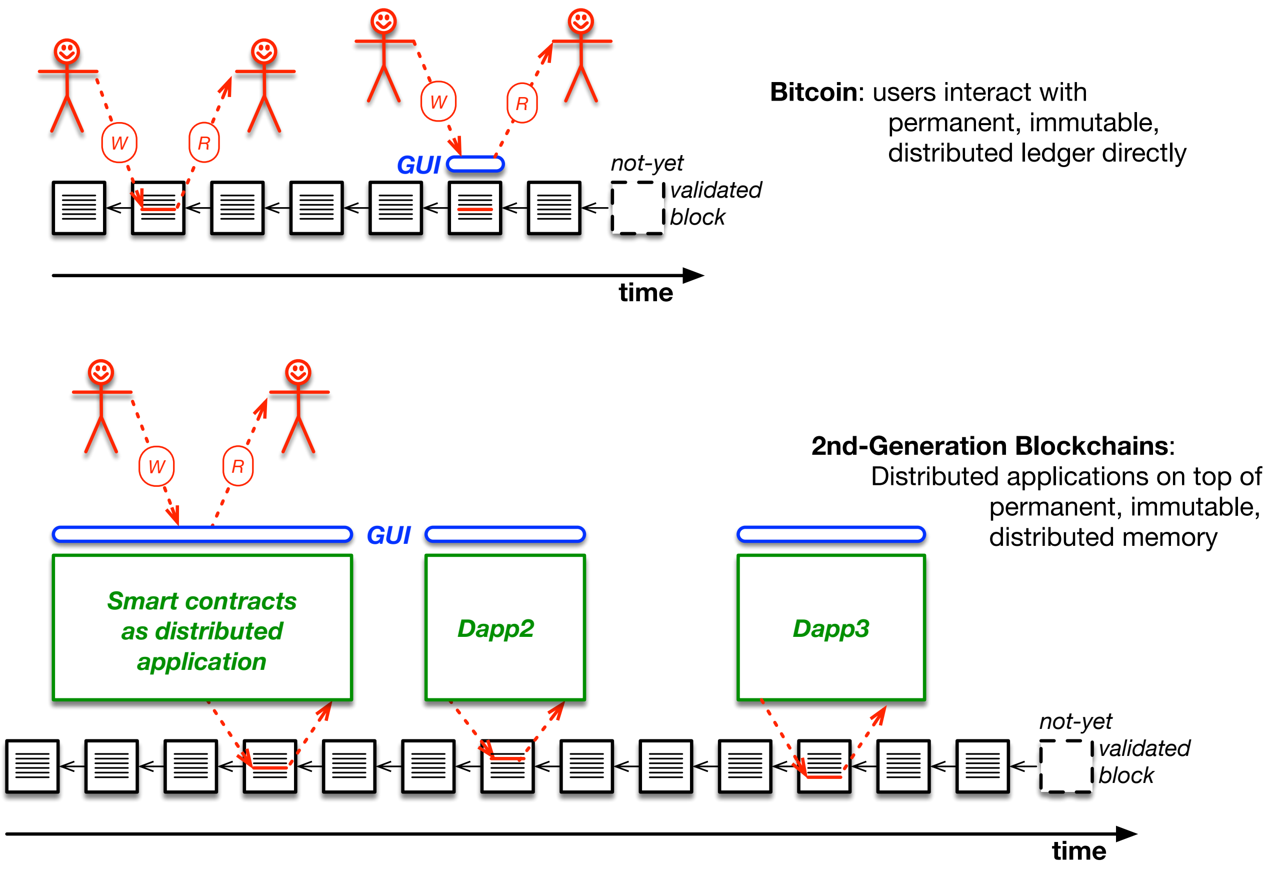
W
R
W
R
GUI
not-yet validated block
time
Bitcoin: users interact with permanent, immutable, distributed ledger directly
W
R
GUI
Smart contracts as distributed application
Dapp2
Dapp3
2nd-Generation Blockchains: Distributed applications on top of permanent, immutable, distributed memory
not-yet validated block
time

Fig. 2.1: **Second-generation blockchains enable a peculiar type of distributed application**

As explained in the Corda technical white paper ([9]: 30), the need for organising the transactions into blocks is a consequence of the fact that the rate at which transactions are performed is greater than the rate at which they can be validated by all the peers of a large distributed system with network latencies. The block structure permits the validation of a set of transactions at a time. Since Corda is a permissioned distributed database, it does not need to achieve global consensus and therefore does not need to employ blocks.

2.2.2 Deterministic Execution

As discussed in the Hyperledger technical white paper [1], distributed systems have relied on the state machine replication (SMR) paradigm [13] for a few decades. The SMR approach is motivated by the need for redundancy in the provision of services to a given client as a strategy to offset possible faults. Each service request from a client is executed in an identical, deterministic, and asynchronous way by a set of servers, which means that the same operations are executed in the same order by each server, even though not necessarily at the same time. By definition, the correct response is whatever the majority of the servers calculates. Therefore, SMR is resistant also to Byzantine faults, which are malicious and not merely technical. More precisely, an SMR system is t-resistant as long as no more than t servers are affected by faults (Byzantine or otherwise) for a total set comprising $2t + 1$ servers. Most blockchains have taken SMR as a starting point, where the set of operations in this case is one block of transactions. In particular, most global consensus algorithms are centred around reaching agreement on the ordering of the transactions in a given block – hence the name 'order-execute' for this type of distributed architecture.

For example, in Bitcoin order-execute involves deterministic sequential execution by each peer, after the first peer has ordered a block, solved the PoW puzzle, and broadcast the block by gossip. The validation of the new block is achieved once every peer has completed and validated the execution. In a public blockchain such as Ethereum a denial-of-service (DoS) attack could be mounted by embedding an infinite loop in the smart contract of one of the transactions. Since it is not possible to determine in general whether or not an algorithm completes ('halting problem'), such a loop could go undetected, leading to a block that cannot be validated and stopping the ability of the blockchain to support future transactions. Ethereum solves this problem by using "gas" which, once converted in the cryptocurrency of that blockchain (ether), results in a charge for the execution of transactions. Gas is essentially a mix between an anti-spam/sybil attack and pay-as-you-write mechanism.

The low efficiency of sequential execution can be improved upon with parallel execution of unrelated transactions. However, detecting the possible interdependencies is not trivial. Stellar and Holochain seem to have been able to do that.

Holochain solves the problem by giving up on global consensus for mutual consensual validation which can be optionally gossiped to a global DHT. This is an agent-centric approach where memory and computation are not shared by default. Stellar uses XLM as Ethereum uses gas, yet manages consensus without PoW but via SCP and a set of known validators. This is a data-centric approach where all transactions need to be on the same single replicated ledger maintained by each node participating in the consensus quorum.

2.2.3 Non-Deterministic Execution

Non-deterministic execution of smart contracts can lead to forks in the blockchain and is therefore avoided when possible. This can be achieved by means of smart contract languages that are not Turing-complete: they should be expressive enough for the purposes of the blockchain they serve (like Solidity for Ethereum) but not so general as to render complete avoidance of non-determinism impossible. For example, Androulaki et al. [1] mention a map iterator in Go that is a deterministic operation at the level of the command but hides a non-deterministic implementation in the Go language itself.

2.2.4 Confidentiality

Performing chaincode on all peers may expose details that some peer would rather be kept private. This issue is particularly important in B2B contexts. One possible solution is to replicate the end-state of the calculation (passive replication) rather than the whole calculation (active replication) [1].

Recent developments in both Hyperledger Fabric, Burrow, and on Ethereum itself indicate that possible future solutions may enable confidentiality at more granular levels via so-called Zero-Knowledge Proofs.[4]

2.2.5 Native Currency

The Bitcoin blockchain provides the simplest example of what a native currency is and how it is created. This is done by a miner by writing a transaction to self of (currently) 12.5 BTC at the beginning of the block he/she will then try to validate by finding the nonce that satisfies the puzzle requirement (i.e. the hash of the miner's current block with the winning nonce) for the current block.

[4] See e.g. work from ING: https://github.com/ing-bank/zkproofs

If the miner wins the current PoW puzzle the 12.5 BTC to self will become "real" and will be credited to her private key.

Many permissioned blockchains, like Hyperledger, do not have a native currency because there is no need to reward nodes (which can be anonymous) for participating in the consensus and chain security computations. Given the requirements of INTERLACE, there seems to be no need for a native protocol currency. Indeed mutual credit works by moving from the paradigm of liquidity to that of clearing and therefore from viewing money as a commodity to money as unit of information meant to facilitate trade and local economic development. The important concept is not whether or not an *asset* is present but what value someone's credit *balance* has – a value that can be negative as well as positive.

2.2.6 Latency vs. Performance in Consensus

As shown in Figure 2.2, the consensus objective can be cast as a trade-off between network latency vs. blockchain performance expressed in transactions per second $[Tx/s]$ ([16], cited in [1]). Stellar here claims the middle ground.

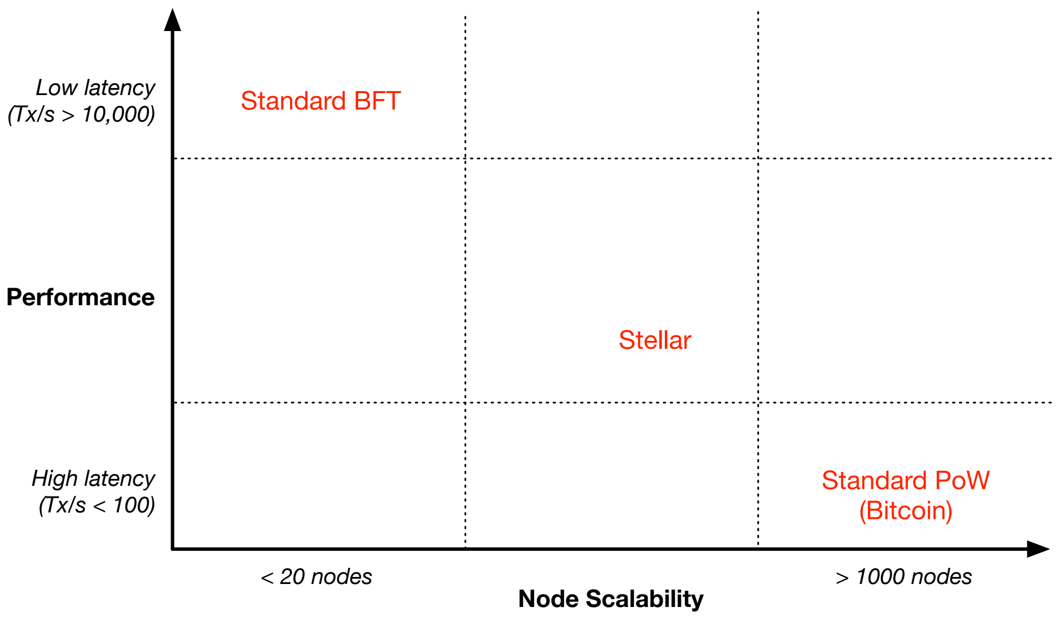

Fig. 2.2: **Network latency-node scalability trade-offs, after [16]**

2.3 Brief Summary of Some DLT Technologies

The INTERLACE team has looked at a number of blockchain technologies, listed here in order of decreasing openness:

- Holochain
- Ethereum
- Stellar
- Quorum (permissioned Ethereum)
- Hyperledger Fabric
- Corda

In this chapter we analyse and discuss them briefly in turn, emphasising the algorithmic, architectural, mathematical, or financial aspects that are most pertinent to INTERLACE. The frameworks that have made it into this short list are all interesting for one reason or another, and at different points each of them was seriously taken into consideration for adoption. Stellar has an interesting mathematical foundation which is therefore studied and discussed in some detail in the Appendix since mathematical understanding can be transferred to other frameworks. The one that comes closest to the requirements of the Sardex mutual credit system is Hyperledger, as will be shown when discussing the summary table at the end fo the chapter.

2.3.1 Holochain

Holochain is the only DLT we have examined that is agent-centric. This means that it does not require global consistency on the same data on a global ledger, but keeps track of individual and mutual peers' data histories, with global replication as an optional feature achieved through DHT and gossiping.

Holochain is composed of two parts: Holochain proper is the underlying technology – not a global blockchain but in any case a persistence layer composed of many individual chains [8], and Holo, the interaction, governance, and financial framework that runs on Holochain.

2.3.2 Ethereum

Ethereum is a data and computation-centric distributed application platform. It is currently under heavy development on both its core components (Layer 1) as well as on higher layers such as payment channels or collective funding (ICO). Interestingly, Ethereum as a platform can function with various degrees of openess of partecipation. The public main net is open to anybody without any verification as long as the willing participating nodes follow the protocol rules and provide the required resources. Apart from the public mode, Ethereum can work also in private consortium mode (as discussed below for Quorum) as well as with side-chains (see Proof of Authorithy where a known-to-random validator set takes care of consensus). It is the platform with the widest developer adoption; yet, in its public version it is too slow to support the full transactional engine needed for current and future Sardex operations

2.3.3 Stellar

Stellar is a system for international currency transfer and exchange. Thus, although it has its own native token (Lumens, XLM), this is similar to Ethereum gas in the sense that its main function is to prevent spamming. Thus, the transaction fees to be paid in XLM are very small. The actual assets being exhanged are "credits" acquired from Stellar Anchors (agents performing and holding the funds which are issued as digital credits on the Stellar ledger). The credits correspond to fiat currency that the counterparties wish to trade or exchange. In other words, the units being recorded on the Stellar ledger chain are analogous to the "statistical Euros" discussed in deliverable D3.1 [10].

2.3.4 Quorum

Quorum is a private and permissioned version of Ethereum developed by J.P. Morgan. It is optimised for enterprise consortia.[5] [6]

2.3.5 Hyperledger

The main points of interest of Hyperledger Fabric are [1]:

[5] https://github.com/jpmorganchase/quorum-docs/blob/master/Quorum_Architecture_20171016.pdf
[6] quorumwhitepaper

- It supports modular consensus protocols, which allows the system to be tailored to particular use cases and trust models.
- Fabric is the first blockchain system that runs distributed applications written in standard, general-purpose programming languages, without systemic dependency on a native cryptocurrency. This stands in sharp contrast to existing blockchain platforms that require smart contracts to be written in domain-specific languages or rely on a cryptocurrency.
- Fabric realizes the permissioned model using a portable notion of membership, which may be integrated with industry-standard identity management.
- Fabric achieves end-to-end throughput of more than 3500 transactions per second in certain popular deployment configurations, with sub-second latency, scaling to well over 100 peers.

Hyperledger Indy-Plenum Byzantine Fault Tolerance (PBFT).[7] PBFT is based on RBFT: Redundant Byzantine Fault Tolerance [3].

2.3.6 Corda

Corda [9] is a permissioned distributed database for banking networks which does not use a blockchain. As explained in [9] (pp 29-30) and mentioned above, the structuring of a blockchain into blocks is a consequence of the difference in time needed to perform a transaction and to reach consensus on that transaction. If in a blockchain like Bitcoin or Ethereum each transaction needed a proof of its validity by global consensus, separately, the number of circulating but not yet proven transactions would be huge and the stable, validated ledger would grow even more slowly. By grouping the transactions into blocks that need to be validated the number of things the network needs to reach consensus on is greatly decreased, and the system reaches some level of efficiency (although much less than VISA – 7 Tx/s for Bitcoin vs. 57,000 max Tx/s for VISA, for example [2]).

Since Corda is permissioned and does not require global consensus, it does not need to group transactions into blocks or to run a consensus protocol. The transactions still need to be verified and validated, which is done by so-called Notaries, after which they are written into the ledger. The ledger is distributed and each node has a copy, but not all nodes see the same information. Visibility of the data is strictly related to whether or not the data is directly related to a particular peer or counterparty in the transaction.

2.4 Summary and Comparison Table

Tables 2 and 3 provide a summary of some of the aspects we have analysed when comparing different blockchain frameworks. The rows are in the same order as the sub-sections of the previous section, from the most open to the most closed frameworks. The order of the topics in each table could perhaps be organised better; they were added on as they were encountered during the literature review. There are two tables due to page layout constraints, and each table has two sets of rows because the number of properties we examined was too large to fit in a single row. So Tables 2 and 3 are in practical terms a single comparison table. The six frameworks listed represent the best candidates out of the much larger set that was examined.

We ended up choosing Hyperledger, so this table could be used as a way to explain the different trade-offs we encountered in making this choice. For example, as already discussed from the proposal stage of the project, the first choice was to use a private, permissioned platform as the most logical transition from the current centralised relational database to a blockchain for business and trade, where privacy matters.

[7] https://github.com/hyperledger/indy-plenum/wiki

However, we examined also several public chains because some aspects of the system may benefit from greater transparency. For example, once circuits are established beyond the Eurozone it will become necessary to set up a credit-currency conversion protocol for inter-circuit trades, since each circuit's credit currency will be pegged 1-1 to the local fiat currency of the country where it is located. It may be better to use a public chain for such a currency exchange function, which we envisage as involving a *single* rate of exchange in both directions and not two separate rates, one for buying and one for selling as is done normally by the banks and exchange bureaus. In other words, speculation on the currency markets would go against the principles of the circuits, which are built on and in support of the (local) real economy and not the financial economy. Therefore, the greater transparency afforded by a public, permissionless chain would be more appropriate for this function than a private chain.

DLT	**Focus**	**Read Access**	**Write Access**	**Data/ Agent**	**Smart Contracts**	**Smart Contract Language(s)**	**Consensus Model**
Holochain	General-purpose platform	Public DHT	Permissionless DHT	Agent-centric	Yes	Lisp, JS, Rust	Local
Ethereum	General-purpose platform	Public Chain	Permissionless Chain	Data-centric	Yes	Solidity on EVM	Global, PoS
Stellar	Cross-border exchange	Public	Permissionless	Data-centric	Limited	Limited	SCP (FBAS)
Quorum	Public and banking sectors	Private	Permissioned	Data-centric	Yes	Solidity on EVM	Configurable, voting-based (PoS, LE, BFT)
Hyperledger	Enterprise, B2B, & supply chain	Private	Permissioned	Data-centric	Yes	JS, Go, Java	Configurable, voting-based (PoS, LE, BFT)
Corda	Business agrmts between fin. institutions	Private	Permissioned	Data-centric	Yes	Bytecode subset on JVM	Local state (Notary pools), pluggable

DLT	**Backup System**	**Interfaces**	**Monetary Model**	**Blockchain**	**Immutable**	**Transaction Validation**	**Architecture**
Holochain	No	Rust, WebAssembly	Mutual Credit	Individual chains	No (link to repl. code)	Local to the parties	
Ethereum	External mirror DB	SQL	Native and token Assets	Yes	Yes	Each peer	Order-Execute
Stellar	MySQL, PostGres	Horizon	Native and token Assets	Chain of ledgers	Yes	Global	Execute-Order-Validate
Quorum	No	Constellation	Token assets	Yes (Ethereum)	Yes	Every node. Data local to the parties	Order-Execute
Hyperledger	Key-value store DB	SQL NoSQL	Adaptable to Mutual Credit	Yes	Yes	Validator peers	Execute-Order-Validate
Corda	H2, Postgres	SQL FPML	Token assets	Global ledger locally visible	Yes	Local to the parties	Execute-Order-Validate

Table 2: **Summary table comparing the main properties of different types of blockchain**

The next property of some interest and relevant is the difference between data-centric and agent-centric blockchains. Holochain is currently the most well-known agent-centric platform in the monetary space, although other agent-centric system that have emerged recently include the so-called DWEB space. Notable examples are Scuttlebutt,[8] WebTorrent,[9] BeakerBrowser,[10] IPFS,[11] Aragon,[12] Matrix,[13] IndieAuth,[14] and ActivityPub.[15]

[8] https://hacks.mozilla.org/2018/08/dweb-social-feeds-with-secure-scuttlebutt/
[9] https://hacks.mozilla.org/2018/08/dweb-building-a-resilient-web-with-webtorrent/
[10] https://hacks.mozilla.org/2018/08/dweb-serving-the-web-from-the-browser-with-beaker/
[11] https://hacks.mozilla.org/2018/08/dweb-building-cooperation-and-trust-into-the-web-with-ipfs/
[12] https://hacks.mozilla.org/2018/09/aragon-ethereum-dweb/
[13] https://hacks.mozilla.org/2018/10/dweb-decentralised-real-time-interoperable-communication-with-matrix/
[14] https://hacks.mozilla.org/2018/10/dweb-identity-for-the-decentralized-web-with-indieauth/
[15] https://hacks.mozilla.org/2018/11/decentralizing-social-interactions-with-activitypub/

Although the development of Holochain is not yet completed, it is interesting because it allows the platform to run on each terminal (e.g. phone) separately and independently, an architecture which has been dubbed "fog computing".[16] In Holochain, each agent has its own individual chain and applications "DNA"). This could be useful for scalability purposes, in the long term, and could also be important as a censorship-resistant alternative to the Bitcoin and Ethereum architectures. In the short term the data-centric approach is sufficient for the objectives of INTERLACE.

Smart contracts are obviously of central importance in a business environment where the business logic needs to be formalised. The consensus model is not as important in a permissioned environment, where the rate of transaction processing and completion takes precedence. Different backup systems and interfaces are available on different chains, but they are not as important discriminating properties.

DLT	**Regulatory/ Supervisory nodes**	**Explicit links of SCs to legal prose**	**Business flow modelling**	**Computational Model**	**Turing-Complete**	**Contract Object**	**Inter-Node Comm.**
Holochain	Random Peers	No	No		Yes		Local
Ethereum	No	No	No	Virtual (global) Computer	No	Stateful	Global
Stellar	Compliance Server	No	No	Ledger + headers	No	Ledger-based	
Quorum	No	Yes	No	Virtual (global) Computer	No	Stateful	Local
Hyperledger	Yes	Yes, with Accord/Clause	No	UTXO + Worldstate	Yes		Global
Corda	Yes	Yes	Yes	UTXO	Yes	Stateless	Local (Flows)
DLT	**Native Token**	**Transaction Fees**	**Backing**	**Convertibility**	**AML Compliance**		
Holochain	Yes (Holo Fuel + HOT Token on Ethereum)	Yes (Holo Fuel)	Fiat currency, Hosting CPU	Yes	Up to app providers		
Ethereum	Yes (Ether: ETH)	Yes (Gas)	No	Yes	No		
Stellar	Yes (Lumens: XLM)	Yes, in XLM (Very small)	No	N/A	MySQL PostGres		
Quorum	No	Custom-izable	N/A	Yes	Identity service/node		
Hyperledger	No	Custom-izable	N/A	N/A	Identity service/node		
Corda	No	Custom-izable	N/A	N/A	Identity service/node		

Table 3: **Summary table comparing the main properties of different types of blockchain (contd.)**

The most important property after the private/public choice is that the INTERLACE blockchain must support mutual credit. Out of the frameworks we examined, only Holochain and Hyperledger are compatible with mutual credit. Holochain was designed from the ground up as a mutual credit system, whereas Hyperledger can be adapted to become a balance-centric rather than asset-centric chain. In all the others the "asset" cannot be a balance that can be either positive or negative: it is assumed to be a positive amount, or a deed, or some quantity of something.[17] Because the Hyperledger

[16] https://en.wikipedia.org/wiki/Fog_computing

[17] In the latest release of Corda it is possible to use an 'obligation' as a debt amount linked to an account (https://docs.corda.net/releases/release-M8.2/api/kotlin/corda/net.corda.contracts.asset/-obligation/). Although this might not be easy to code for us and although Corda is more suitable for networks of banks than for our purposes, their latest developments at the company structure level are extremely interesting for us and are discussed briefly in the final architecture deliverable (D2.3).

assets could be programmed, through smart contracts code, to act as the balance of a mutual credit account, it becomes the obvious choice for INTERLACE.

Whether or not the technology is a blockchain or a more general DLT was not really an issue. Immutability is important for a public chain but not as much for a private chain since as a matter of course we need to mirror everything onto a relational DB as backup. However, it does imply that private user data cannot be stored on-chain, otherwise we would be violating GDPR requirements (for example, the right to be forgotten). Since Hyperledger is immutable, private data[18] can only be stored in a separate DB, either SQL or key-value. Finally, the execute-order-validate architecture appears to be more suitable for permissioned chains and much faster than most consensus algorithms of the public chains.

The next important feature of interest is Turing-completeness. From a security point of view Turing-completeness is not necessarily a good feature because it implies a larger 'attack surface' for malicious hackers. However, it can also imply the use of an established language for the smart contracts. This is the case of Hyperledger, which supports JS, Go, and more recently also Java. In addition, languages that have been developed specifically for smart contracts can have bigger problems. For example, Ethereum's Solidity is not regarded as a properly structured language, and also has security issues of its own.[19] For these reasons we decided to stick with Hyperledger and the interoperability of well-established languages.

Because Sardex has its own currency unit, a native token was not important for us. In fact, it would create problems since it would interfere with the financial, economic, and *social* functions of mutual credit [12]. Anti Money Laundering (AML) and regulatory compliance, on the other hand, are very important for INTERLACE and for Sardex. This was another point in favour of Hyperledger since it exposes a REST interface that could be fed into a regulator's dashboard or data explorer.

[18] 'Personal information', as per the GDPR definition.
[19] See, for example, `https://news.ycombinator.com/item?id=14691212`

Chapter 3

Conclusion

In conclusion, this deliverable reported on the main blockchain technologies that we have investigated, and concluded that Hyperledger Fabric is the best option for INTERLACE. The mathematical basis of the Stellar consensus algorithm were studied in some detail, and reported in the Appendix, in order to get a better understanding of the level and type of rigour that can be obtained about truth statements related to variable-node networks under possible Byzantine attacks. The iterative refinement of the specification is actually reported in Chapter 2 of D3.1 [10].

References

References

1. E Androulaki, A Barger, V Bortnikov, C Cachin, K Christidis, A De Caro, D Enyeart, C Ferris, G Laventman, Y Manevich, S Muralidharan, C Murthy, B Nguyen, M Sethi, G Singh, K Smith, A Sorniotti, C Stathakopoulou, M Vukolic, S W Cocco, and J Yellick. Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains. In *EuroSys 18: Thirteenth EuroSys Conference, April 23-26, 2018, Porto, Portugal*, New York, NY, USA, 2018. ACM. URL: https://doi.org/10.1145/3190508.3190538.
2. A Antonopoulos. *Mastering Bitcoin*. O'Reilly, Sebastapol, CA, USA, 2015.
3. P. L. Aublin, S. B. Mokhtar, and V. Quma. RBFT: Redundant Byzantine Fault Tolerance. In *2013 IEEE 33rd International Conference on Distributed Computing Systems*, pages 297–306, July 2013.
4. P Cameron. *Introduction to Algebra (2nd Ed.)*. Oxford University Press, Oxford, 2008.
5. E De Bono. *The Mechanism of Mind*. Cape, 1969.
6. P Dini, E Börger, E Hirsch, T Heistracher, M Cireddu, L Carboni, and G Littera. *D2.1: Requirements and Architecture Definition*. INTERLACE Deliverable, European Commission, 2017. URL: http://www.interlaceproject.eu/.
7. P Dini, G Littera, L Carboni, and E Hirsch. *D2.3: Final Architecture*. INTERLACE Deliverable, European Commission, 2018. URL: http://www.interlaceproject.eu/.
8. E Harris-Brown, N Luck, and A Brock. Holochain: Scalable agent-centric distributed computing, 2018. DRAFT(ALPHA 1) 15 February. URL: https://github.com/Holochain/holochain-proto/blob/whitepaper/holochain.pdf.
9. M Hearn. Corda: A distributed ledger, 2016. Version 0.5. URL: https://docs.corda.net/_static/corda-technical-whitepaper.pdf.
10. E Hirsch, T Heistracher, P Dini, E Börger, L Carboni, M L Mulas, and G Littera. *D3.1: First Demonstrator Implementation*. INTERLACE Deliverable, European Commission, 2018. URL: http://www.interlaceproject.eu/.
11. D Mazières. The Stellar Consensus Protocol: A Federated Model for Internet-level Consensus, 2016. Stellar Development Foundation, https://www.stellar.org/papers/stellar-consensus-protocol.pdf.
12. L Sartori and P Dini. Sardex, from complementary currency to institution. A micro-macro case study. *Stato e Mercato*, 107:273–304, 2016.
13. F Schneider. Implementing Fault-Tolerant Services Using the State Machine Approach: A Tutorial. *ACM Computing Surveys*, 22(4):299–319, 1990.
14. P Tasca, T Thanabalasingham, and C J Tessone. Ontology of Blockchain Technologies: Principles of Identification and Classification. *CoRR*, abs/1708.04872, 2017.
15. E Vosselman and J van der Meer-Kooistra. Accounting for control and trust building in interfirm transactional relationships. *Accounting, Organizations and Society*, 34:267–283, 2009.
16. Marko Vukolić. The quest for scalable blockchain fabric: Proof-of-work vs. BFT replication. In *International Workshop on Open Problems in Network Security*, pages 112–125. Springer, 2015.

Basic Mathematical Framework for Stellar

Paolo Dini

A.1 Introduction

In this Appendix we provide a mathematical summary of the Stellar Consensus Protocol (SCP) [11]. The presentation involves a sequence of definitions interspersed with theorems and their proofs. Understanding the theorems and their proofs is essential to understanding the Stellar system and the SCP. Therefore, although we follow Mazières's paper [11] very closely, in some cases we provide more elementary explanations of the mathematical formulation, relying on figures and diagrams created ad hoc as needed.

SCP is based on a new decentralized agreement system, also defined and developed in [11], called Federated Byzantine Agreement System (FBAS). FBAS and SCP, together, provide an alternative to Bitcoin's Proof of Work (PoW) [2] or Ethereum's Proof of Stake (PoS)[20] to achieve consensus. FBAS is a generalization of Byzantine agreement,[21] where the latter is analogous to a permissioned system. Although the early implementation of the new Sardex INTERLACE platform will be permissioned, we wish to develop an architecture that can easily replace centralized control with local consistency between transacting parties. At the level above in the network hierarchy, we will also need to manage a federated network of circuits. Also here the capability to transition to a scalable decentralized architecture is preferable to a traditional centralized approach, although at this level Corda may be a more appropriate Distributed Ledger Technology (DLT) framework.

A.2 Federated Byzantine Agreement System

A.2.1 Basic Components

A Stellar network relies on a FBAS to achieve consensus in the absence of central control. The consensus is on the update of replicated states such as ledger records corresponding to transactions. The purpose of the consensus protocol is for a set of nodes to reach agreement on a given update. Each update is identified with a unique *slot* which also encodes information on inter-update dependencies – for example as consecutively numbered positions in a transaction ledger. A FBAS runs a SCP that ensures that the nodes agree on slot contents. Agreement is defined in terms of safety of operation; the definition is somewhat recursive so it may need to be refined later:

Definition 1. *We say that a node v can* **safely** *apply update x in slot i when*

- *it has safely applied all the updates in all the slots upon which i depends, and*
- *it believes all correctly functioning nodes will eventually agree on x for slot i.*

Definition 2. *When node v has safely applied update x in slot i we say that v has* **externalized** *x for slot i.*

The reason for this term is that once the contents of a slot have been accepted by other nodes they (the other nodes) could perform irreversible actions as a consequence, which v cannot do anything about: the contents are now outside or *external* to the control of the originator node v.

A challenge for FBA is that malicious nodes can outnumber honest ones, such that determining a quorum by simple majority is not sufficient to guarantee safety. FBA selects quorums in a decentralized way, leading to a layering of the network into a hierarchy such that different structures are relevant at different levels, as shown in Figure .1. The top level is provided by a set of nodes V. The second level is called a 'quorum', denoted by U and defined as a set of nodes sufficient *for those nodes* to reach agreement. The level below is a set of 'slices' of a given node v, written $Q(v)$, where a **slice** is a subset of a quorum whose nodes agree with that *single* node v. As the figure shows, a node v may belong to more than one slice, with the cardinality of $Q(v)$ (i.e. $|Q(v)|$) providing the quantification. We now provide formal definitions and mathematical relations between these quantities.

Using the usual notation 2^X for the power set[22] of the set X, 2^V is the power set of V, i.e. the set of all possible quorum slices. At the next level, 2^{2^V} is the power set of quorum slices, i.e. the set of all possible *sets* of quorum slices. Figure .2 shows a visualization of the elements of these kinds of power sets.

[20] https://github.com/ethereum/wiki/wiki/Proof-of-Stake-FAQ

[21] https://en.wikipedia.org/wiki/Byzantine_fault_tolerance

[22] The power set of a set X is the set of all possible subsets of X [4]. For a set of N elements, its power set turns out to have 2^N elements, hence the notation.

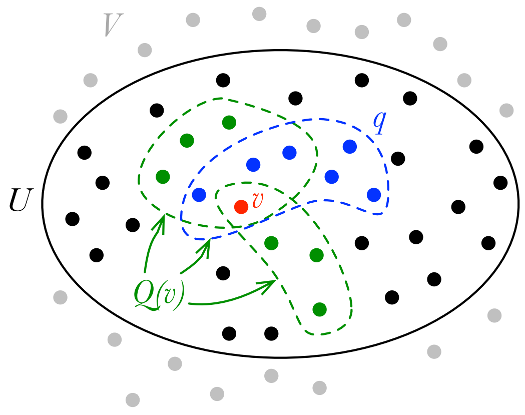

Fig. .1: **The five levels of a Stellar network or FBAS**

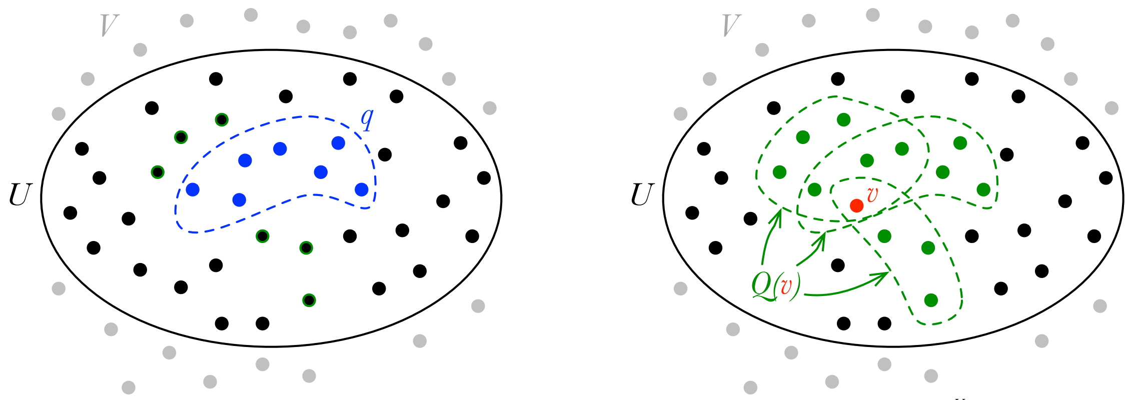

Fig. .2: **L: a)** q **is an element of** 2^V**. R: b)** $Q(v)$ **is an element of** 2^{2^V}**.**

The function Q assigns to each node $v \in V$ a set of slices $\{q_1, q_2, \cdots, q_k\}$. Here the curly brackets denote set and k is the number of slices for a given node v. Formally,

$$Q\colon V \to 2^{2^V} \setminus \emptyset, \tag{1}$$

where $\setminus\emptyset$ means that the empty set is not in the range of this function ($Q(v)$ can never be empty).[23] In general,

$$\forall v \in V, \forall q \in Q(v), \qquad v \in q, \tag{2}$$

which reads 'For all nodes v members of the set V and for all slices q members of the set $Q(v)$, node v is a member of some slice q'.

Definition 3. *A Federated Byzantine Agreement System* (**FBAS**) *is a pair* (V, Q).

Definition 4. *A set of nodes* $U \subseteq V$ *in a FBAS* (V, Q) *is a* **quorum** *iff* $U \neq \emptyset$ *and* U *contains a slice for each member:*

$$\forall v \in U, \exists q \in Q(v) \mid q \subseteq U. \tag{3}$$

In English: for all v members of a quorum U, there exists a slice q, member of $Q(v)$, such that the slice is a subset of, or equal to, the quorum U.

Definition 5. *A* **quorum slice** q *is a subset of a quorum* U *sufficient to convince a particular node* v *of agreement.*

[23] It is not immediately clear whether this is a consequence of how this function operates or whether it is a requirement that we are adding arbitrarily to ensure that it does what we need. TBD.

Therefore, a quorum slice is usually smaller than a quorum, as shown in the figures above.

The 'convincing' is depicted graphically by an arrow that indicates dependence between nodes in a manner analogous to inheritance in UML class diagrams. For example, as shown in Figure .3a, v_2 can convince v_1 but not vice versa, i.e. v_1 depends on itself and on v_2 while v_2 depends only on itself. Figure .3b shows the corresponding slices. Note that in this simple case the quorum of this set of nodes $V = \{v_1, v_2\}$ equals the slice for v_1: $V = U = q(v_1) = \{v_1, v_2\}$, whereas the set of slices $Q(v_1)$ is written $Q(v_1) = \{\{v_1, v_2\}\}$, where the outer curly brackets indicate the set of slices $Q(v_1)$ and the inner curly brackets indicate the slice $q_1 = q(v_1)$.

A.2.2 Examples

Figure .4 shows a more complex interdependence between a set of 4 nodes, as Example 1. The arrows imply that v_2, v_3, and v_4 each has the same slice. Thus, $q(v_2) = q(v_3) = q(v_4) = \{v_2, v_3, v_4\}$. In this case we also have that $Q(v_2) = Q(v_3) = Q(v_4) = \{\{v_2, v_3, v_4\}\}$. In this example, although $\{\{v_2, v_3, v_4\}\}$ is a quorum, the smallest quorum involving v_1 must involve all four nodes.

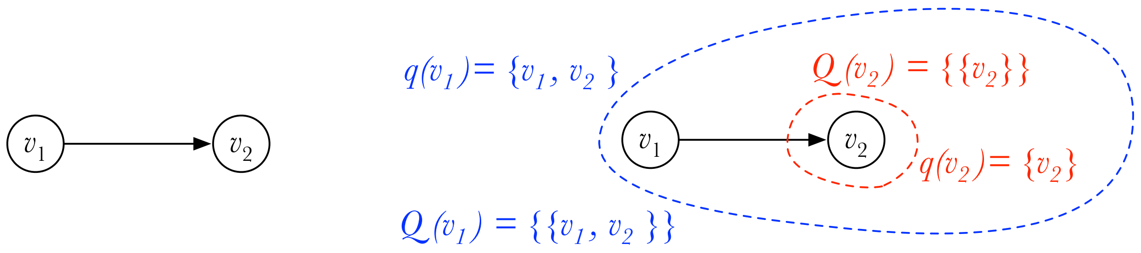

Fig. .3: **L: a) v_2 can convince v_1 but not vice versa. R: b) The slices of v_1 and of v_2.**

Figure .4 also provides a clear example of the difference between a slice and a quorum. The slice of v_1 is $q(v_1) = \{v_1, v_2, v_3\}$, which means that these three nodes are all that's needed to convince v_1. However, this set of nodes cannot be a quorum because the slices of v_2 and v_3 include nodes (actually only one node, v_4) that are outside of this set, which contravenes Definition 4.

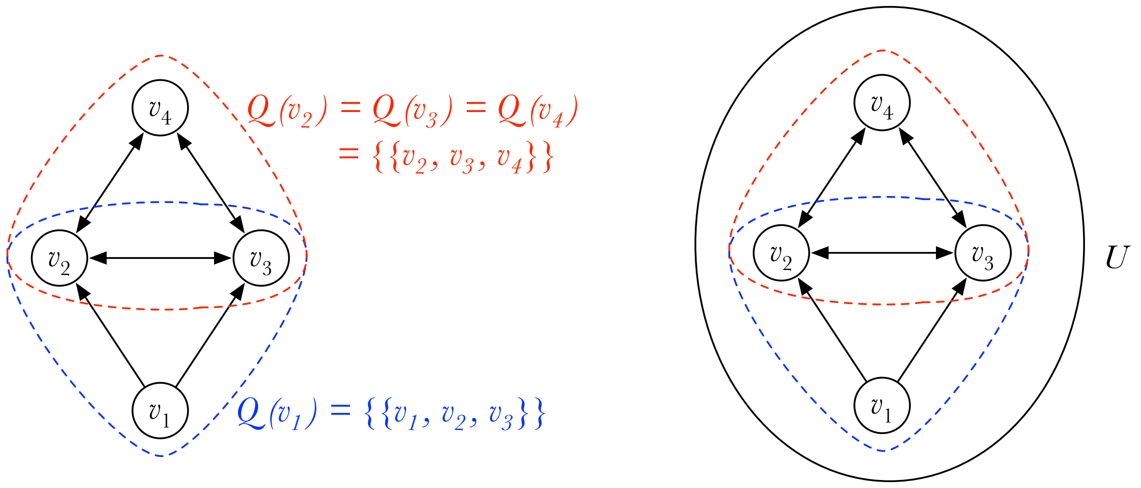
v_4
v_2
v_3
v_1
$Q(v_2) = Q(v_3) = Q(v_4)$
$= \{\{v_2, v_3, v_4\}\}$
$Q(v_1) = \{\{v_1, v_2, v_3\}\}$
v_4
v_2
v_3
v_1
U

Fig. .4: **Example 1. L: a) The slices of this network. R: b) The smallest quorum involving v_1. (After [11])**

In traditional, non-federated Byzantine systems all nodes have the same slices. Therefore, non-federated systems do not distinguish between slices and quorums:

$$\forall v_i, v_j \in V, \qquad Q(v_i) = Q(v_j). \qquad \text{[BAS]} \tag{4}$$

In our case, instead, in general we have that

$$\forall v_i, v_j \in V, \qquad Q(v_i) \neq Q(v_j). \qquad \text{[FBAS]} \tag{5}$$

The important point made by Mazières is that whereas BAS is not scalable, as exemplified by the huge CPU overhead the Bitcoin consensus protocol requires, FBAS is scalable.

Example 2 It is worth discussing a second example from Mazières, as shown in Figure .5. The figure shows three new notational concepts:

- a reflexive dependence arrow can be drawn explicitly
- sets of nodes can be grouped explicitly into separate sets or tiers, and
- the dependence arrows can carry specific labels.

For example, the label on the reflexive arrow on the top tier indicates that at least 3 nodes are needed for consensus. According to the definitions above for the arrow directions, the top tier does not depend on the tiers below, and it is precisely for this reason that it is the 'top' tier. The label on the arrow between the middle and top tiers indicates that the slice of any one node from the middle tier is composed of itself plus any two of the tier above. The slice of either v_9 or v_{10} is composed of itself plus any two nodes from the middle tier. Since each of these, in turn, depends on any two of the nodes of the top tier, neither such slice can be a quorum. A quorum involving either v_9 or v_{10} will have at least 5 and at most 7 nodes. A quorum involving *both* v_9 and v_{10}, on the other hand, will have at least 6 and at most all 10 nodes.

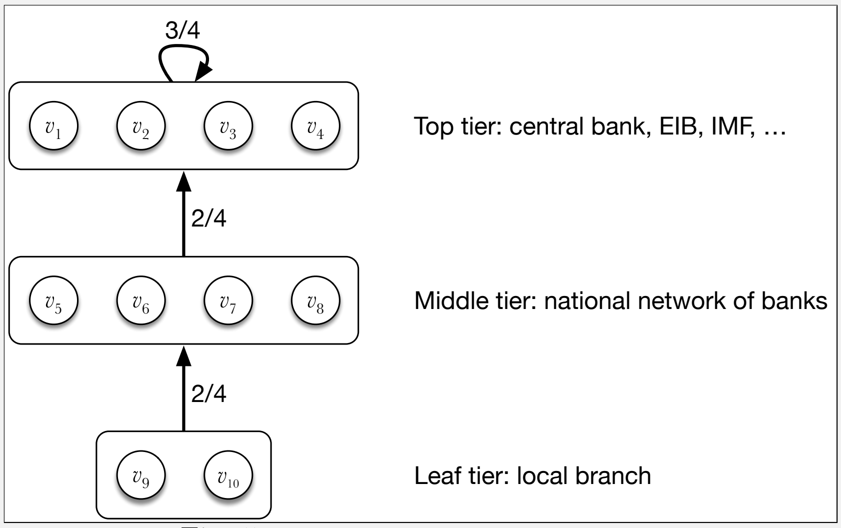

Fig. .5: **Three-tier network (After [11])**

In Figure .5 the top tier according to Mazières could be composed by a number of global financial institutions. The middle tier could be sets of banks operating at national level, with a single such set corresponding to a single country shown here. The bottom tier would be a single branch office, with the nodes inside representing individual customer accounts.

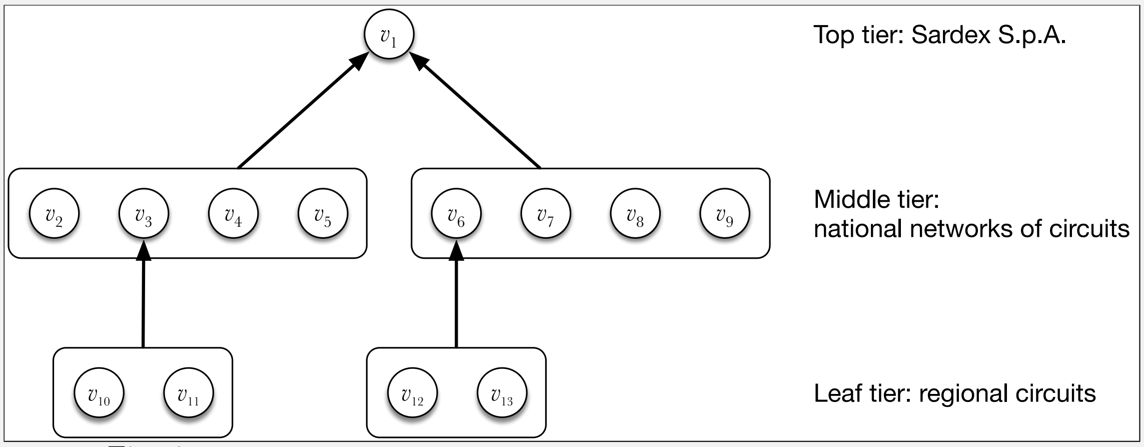

Fig. .6: **Possible mapping to INTERLACE circuit architecture. (After [11])**

In Figure .6, on the other hand, the top tier would be only the single Sardex S.p.A. company. The middle tier could be composed by different national jurisdictions, each of which containing a set of circuits; the figure shows two such jurisdictions corresponding to two different countries. The bottom tier would then correspond to separate individual circuits, with the nodes corresponding to individual company accounts. There are no labels on the arrows because the dependence is to a specific node in every case, generating a centralized hierarchical network.

Although the architecture of Figure .6 may correspond to an initial implementation of the INTERLACE platform, it does not make use of the distribution capability of the Stellar consensus protocol, so let's proceed with the development of the protocol to see what will become possible.

A.2.3 Safety and Liveness

Nodes can be either well-behaved or ill-behaved. A well-behaved node chooses sensible quorum slices and obeys the protocol, including responding (eventually) to all requests. Ill-behaved nodes suffer Byzantine failure, meaning that they behave arbitrarily due for example to a malicious modification of the software, or may have crashed. The goal of FBAS is to ensure that all well-behaved nodes externalize the same values in spite of such ill-behaved nodes.

Definition 6. *Two nodes in a set V are* **divergent** *if they externalize different values to the same slot. A set V of nodes in a FBAS is* **safe** *if no two of its elements (nodes) externalize different values for the same slot.*

Definition 7. *A node $v \in V$ is* **blocked** *if it is in some dead-end state from which consensus is no longer possible. A node $v \in FBAS$ has* **liveness** *if it can externalize new values without the participation of any failed or ill-behaved nodes.*

Definition 8. *Nodes that enjoy both safety and liveness are called* **correct***. Nodes that are not correct have* **failed**.

Thus, nodes without liveness are blocked, whereas nodes without safety are divergent. Figure .7 shows the set inclusion relationships between these different kinds of nodes in a set V. The reason a divergent or blocked node is shown under the well-behaved category even though it has failed is that, unlike ill-behaved nodes, it may have started as a correct node but may fail if its choice of slice causes it to wait indefinitely for messages from an ill-behaved node (this makes it **blocked**) or, worse, if its state is maliciously modified by messages from an ill-behaved node (this makes it **divergent**). Thus, the reason divergent nodes are a subset of the blocked nodes is that an attack that violates safety is strictly more powerful that one that violates only liveness.

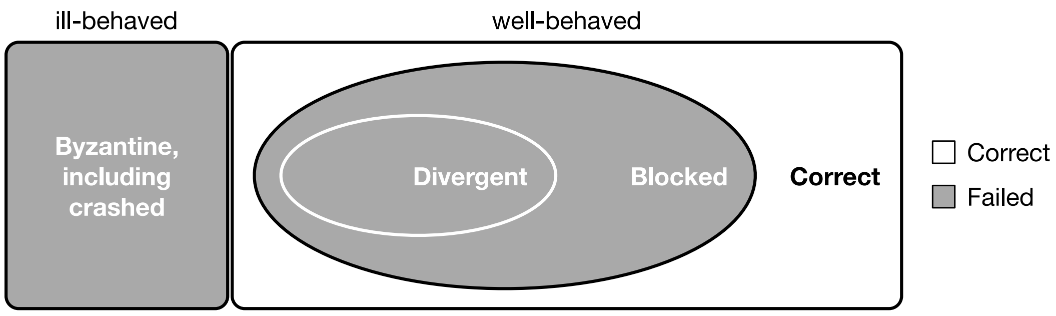

Fig. .7: **Set inclusion relationships between node categories. (After [11])**

Mazières explains that the above definition of liveness is weak because it only says that a node *can* externalize, not that it *will* or *must*. As a consequence, there is a possibility for consensus to be forever delayed by, for example, preemptive reordering of the transactions. It is not clear whether this issue matters in the case of INTERLACE since we will not implement a completely decentralized system: some level of centralized control is likely to remain. Mazières in any case provides relevant references that can be consulted if this issue needs to be analysed in depth and resolved in some robust way.

Optimal Resilience: Quorum Intersection and Dispensable Sets

As is commonly assumed for asynchronous systems, messages between well-behaved nodes are eventually delivered, but can otherwise be delayed indefinitely or reordered arbitrarily. This section starts becoming more involved because it addresses the following non-trivial question: Given a specific (V, Q) and a subset of V that is ill-behaved, what are the best levels of safety and liveness that a FBAS can guarantee in an arbitrary network?

Definition 9. *A FBAS has* **quorum intersection** *iff any two of its quorums share at least one node.*[24] *In other words,* $\forall U_i, U_j \in FBAS$, $U_i \cap U_j \neq 0$, *where* i *and* j *range over the number of quorums for a given FBAS.*

Therefore, when a FBAS has many quorums, quorum intersection (QI) fails when ***any two*** do not intersect. We remind the reader that $Q(v_i)$ is just the set of slices of a given node v_i and, depending on the slices of all the other nodes $v_j \in Q(v_i)$, may or may not be a quorum $U \subseteq V$. Thus, although in the simple examples that follow each $Q(v)$ is a quorum, this is by no means the case in general.

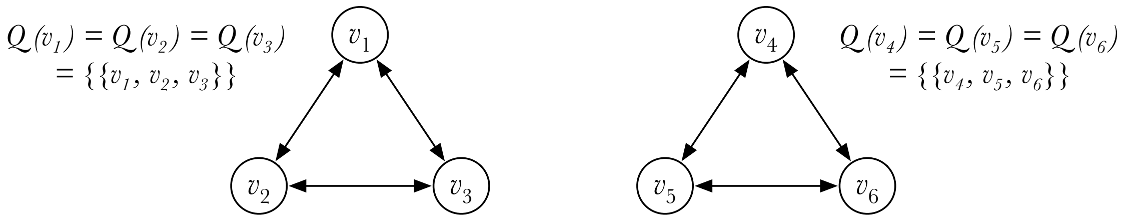
$Q(v_1) = Q(v_2) = Q(v_3)$
$= \{\{v_1, v_2, v_3\}\}$
v_1
v_2
v_3
v_4
v_5
v_6
$Q(v_4) = Q(v_5) = Q(v_6)$
$= \{\{v_4, v_5, v_6\}\}$

Fig. .8: **A set of nodes without QI. (After** [11]**)**

Figure .8 shows a system of 6 nodes that lacks QI, since the function Q allows two quorums on the set of 6 nodes that do not intersect. Mazières says that the two quorums can separately agree on contradictory statements. In other words, since v_5 cannot communicate with v_1, it could say something different from what v_1 is saying for a given slot. Mazières says that the whole set of all nodes $\{v_1, \cdots, v_6\}$ is *also* a quorum, which in this case we can call U, and it intersects the other two. Is this true? If we look back at the definition of a quorum (Definition 4), we can verify that yes, it is correct because each node has a slice that is part of U. So this is an example of a system that has multiple quorums but two of them do not intersect and, therefore, the system does not have QI. We can believe that there is no certainty that v_1 can contradict or agree with v_5 even though they both belong to U.

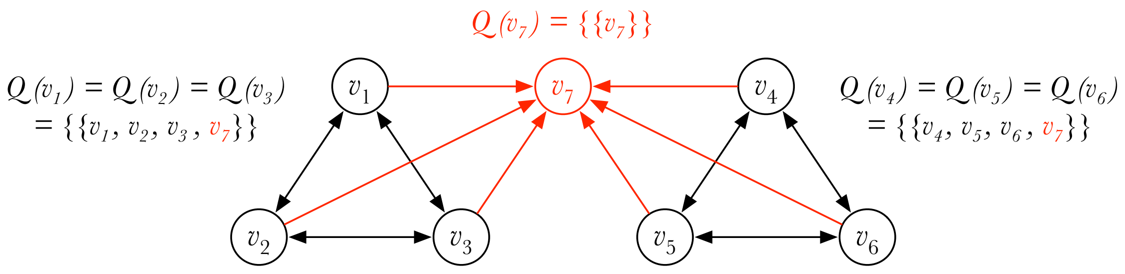
$Q(v_7) = \{\{v_7\}\}$
$Q(v_1) = Q(v_2) = Q(v_3)$
$= \{\{v_1, v_2, v_3, v_7\}\}$
v_1
v_7
v_4
v_2
v_3
v_5
v_6
$Q(v_4) = Q(v_5) = Q(v_6)$
$= \{\{v_4, v_5, v_6, v_7\}\}$

Fig. .9: **A set of nodes with QI. (After** [11]**)**

Safety cannot be guaranteed because such a system can operate like two separate FBAS systems that do not communicate. However, safety may not be possible even in a system with QI, like the one shown in Figure .9, if the intersecting node is ill-behaved. If v_7 makes inconsistent statements to the left and right quorums they are as good as disconnected. Therefore, safety can be guaranteed only if the well-behaved nodes have QI; or, put otherwise, a FBAS (V, Q) can survive Byzantine failure by a set $B \subseteq V$ iff (V, Q) has QI after deleting the nodes in B from V and from all slices in Q.

More formally,

Definition 10. *If* (V, Q) *is a FBAS and* $B \subseteq V$ *is a set of nodes, then to* **delete** B *from* (V, Q), *written* $(V, Q)^B$, *means to compute the modified* $(V \setminus B, Q^B)$, *where*

$$Q^B(v) = \{q \setminus B \mid q \in Q(v)\}, \qquad \forall v \in V \setminus B. \tag{6}$$

It is the responsibility of each node v to ensure that $Q(v)$ does not violate QI. From Mazièress paper it is not clear **how**, however. Be that as it may, assume that Figure .9 evolved from the 3-node FBAS $\{v_1, v_2, v_3\}$ to a system that includes

[24] The term 'iff' means 'if and only if'. It implies that the causal dependence works in both directions, in contrast to merely 'if' which works only in the reverse direction. In symbols, 'A iff B' is written $A \Leftrightarrow B$, whereas 'A if B' is written $B \Rightarrow A$ and is also read as 'B implies A'.

also $\{v_4, v_5, v_6\}$. Assume now that $\{v_4, v_5, v_6\}$ are malicious such that they choose slices that do not satisfy QI. But $Q(v)$ is meaningless for a malicious node. That's why the necessary condition for safety, QI after deleting ill-behaved nodes, is unaffected by the slices of ill-behaved nodes. The system $(V, Q)^{\{v_4,v_5,v_6\}}$ restores QI for $\{v_1, v_2, v_3\}$. Note that we have not yet said how such deletion takes place. For now we just say that the protocol must guarantee safety for $\{v_1, v_2, v_3\}$ without these nodes having to know that $\{v_4, v_5, v_6\}$ are malicious.

Turning now to dispensable sets or DSets, the safety and liveness of the nodes outside a DSet can be guaranteed regardless of the behaviour of the nodes inside the DSet.

Definition 11. *Let (V, Q) be a FBAS and $B \subseteq V$ be a set of nodes. We say that B is a dispensable set or a* **DSet** *iff:*

- *$(V, Q)^B$ has QI* *(Intersection)*
- *$V \setminus B$ is a quorum OR $B = V$* *(Availability)*

As explained by Mazières, availability protects against nodes in B not replying to requests or impeding other nodes' progress. QI protects against nodes in B making contradictory assertions that enable other nodes to externalize inconsistent values for the same slot. These two threats depend on slice size in opposite ways: greater slices increase the chance for QI, but also the chance that they will contain failed nodes, impacting availability. Smaller slices decrease the chance of failed nodes, but also the likelihood of having QI.

The smallest DSet containing all ill-behaved nodes may contain also well-behaved nodes, if they depend on the ill-behaved ones. For example, in Figure .5 if v_5 and v_6 are ill-behaved the smallest DSet would need to include also v_9 and v_{10} since the lowest tier depends on any 2 of the nodes in the middle tier, and so it may depend on the two that are ill-behaved.

Definition 12. *A node $v \in FBAS$ is* **intact** *iff $\exists$ DSet B containing all ill-behaved nodes and such that $v \notin B$.*

Definition 13. *A node $v \in FBAS$ is* **befouled** *iff it is not intact.*

Theorem 1. *Let U be a quorum in FBAS (V, Q). Let $B \subseteq V$, so that B is not necessarily a slice of U. And let $U' = U \setminus B$. If $U' \neq \emptyset$, then U' is a quorum in $(V, Q)^B$.*

Proof. First, as shown in Figure .10A, since U is a quorum, every $v \in U$ has a slice $q \in Q(v)$ such that $q \in U$:

$$\exists q \in Q(v) | q \in U, \qquad \forall v \in U. \tag{7}$$

Second, as shown in Figure .10B, since $U' \subseteq U$, every $v \in U'$ has $q \in Q(v)$ such that $q \setminus B \subseteq U'$:

$$\exists q \in Q^B(v) | q \in U', \qquad \forall v \in U'. \tag{8}$$

Therefore, U' is a quorum in $(V, Q)^B$. $\square$

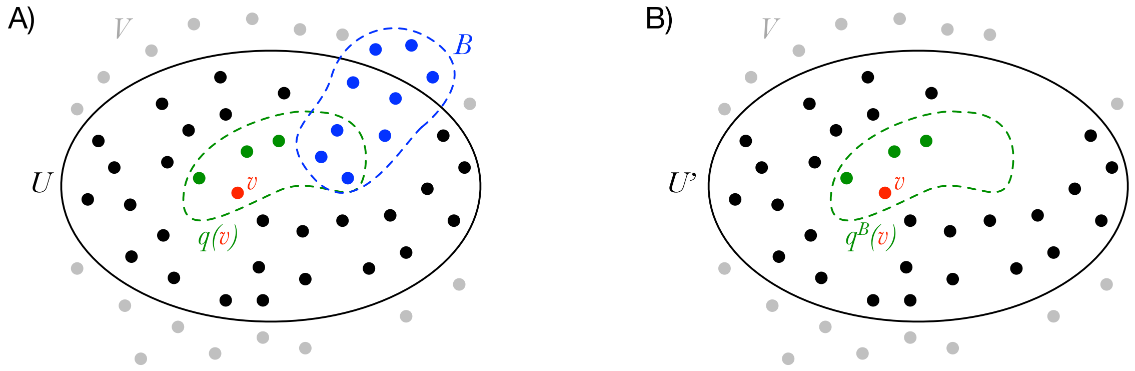

Fig. .10: **A) Quorum $U \subseteq V$ with set $B \subseteq V$. B) $U' = U \setminus B$ and $q^B(v) = q(v) \setminus B$.**

The next theorem is far from obvious.

Theorem 2. *If B_1 and B_2 are DSets in a PBAS (V, Q) that has QI, then $B = B_1 \cap B_2$ is a DSet too.*

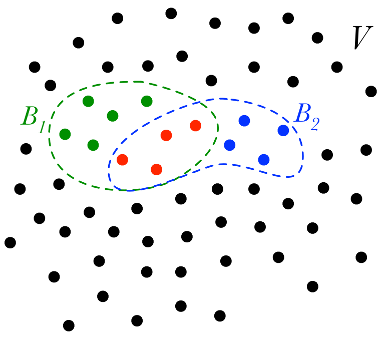

Fig. .11: **Visualization of a set of nodes V with two DSets B_1 and B_2 and their intersection in red**

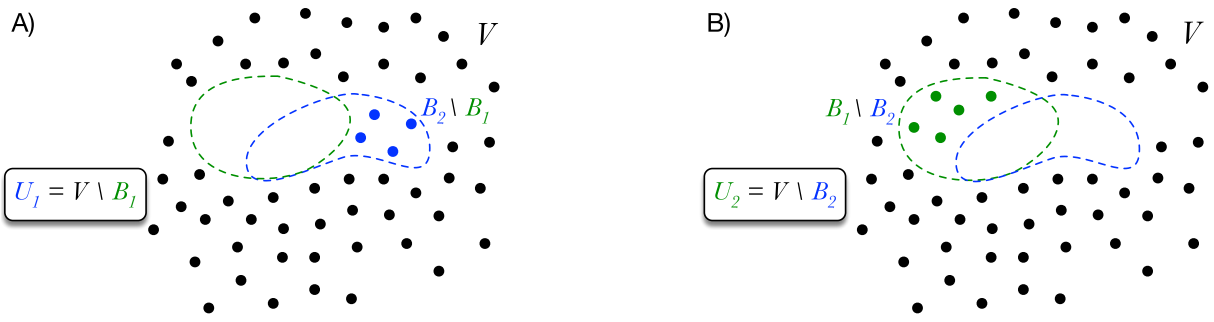

Fig. .12: **The two subsets U_1 and U_2**

Proof. Let $U_1 = V \setminus B_1$ and $U_2 = V \setminus B_2$ be two subsets of V defined by taking away the two DSets in turn, as shown in Figure .12. Note that by Definition 11 both subsets have QI. By the same definition, both subsets are also quorums.

Case 1. If $U_1 = \emptyset$, then $B_1 = V$ and $B = V \cap B_2 = B_2$, which is a DSet.

Case 2. Similarly, if $U_2 = \emptyset$, then $B_2 = V$ and $B = B_1 \cap V = B_1$, which is a DSet.

Case 3. If we are not at these two extremes, as just stated above by quorum availability U_1 and U_2 are both quorums in (V, Q). Also, the quorum definition 4 implies that $U_1 \cup U_2$ (which also equals $V \setminus B$) is also a quorum (see Figure .13A). Therefore, $V \setminus B$ is a quorum. Therefore, $B_1 \cap B_2$ satisfies availability in V despite B. This proves the **availability** condition of a DSet, i.e. availability despite B. Proving **QI** despite B requires quite a bit more work.

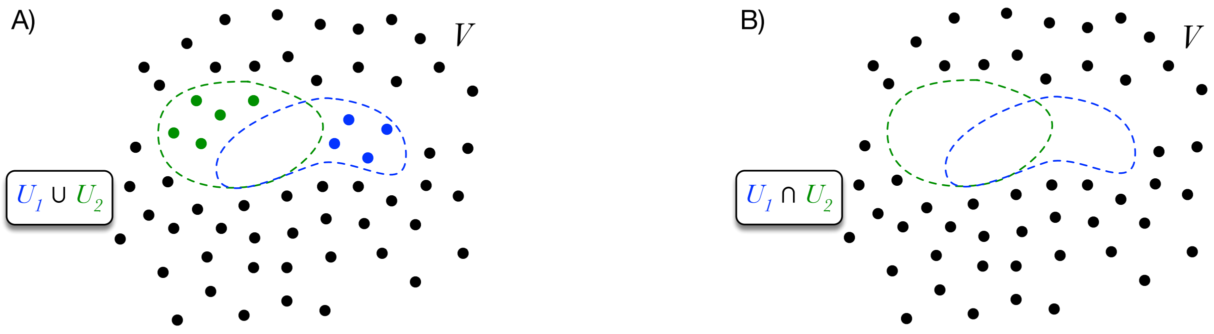

Fig. .13: **Union and intersection of the two subsets U_1 and U_2**

Let $U = U_1 \cap U_2$ (see Figure .13B). Note that U also equals $U_2 \setminus B_1$ and $U_1 \setminus B_2$. By the definition of QI (Definition 9), $U_1 \cap U_2 \neq \emptyset$. Then by Theorem 1 $U_1 \cap U_2 = U_2 \setminus B_1$ is a quorum in $(V, Q)^{B_1}$. In other words, Figure .13B is a quorum in Figure .12A.

∗Let U_a and U_b be two quorums in $(V, Q)^B = U_1 \cup U_2$. We begin the analysis by working with U_a, and repeat the argument later for U_b.

- If $U_a \setminus B_1 = \emptyset$, U_a must be at most $\subseteq B_1$ (but not include B, by construction)
- Similarly, If $U_a \setminus B_2 = \emptyset$, U_a must be at most $\subseteq B_2$ (but not include B, by construction)
- If $U_a \setminus B_1 = \emptyset$ **AND** $U_a \setminus B_2 = \emptyset$, then $U_a = \emptyset$.

Now whether or not B_1 and B_2 are quorums is not relevant to the argument since they are DSets by assumption. They are sets we wish to ignore. Therefore, the cases above are not of interest and are irrelevant. Therefore, we are only interested in the two cases

$$U_a \setminus B_1 \neq \emptyset \qquad \textbf{OR} \qquad U_a \setminus B_2 \neq \emptyset. \tag{9}$$

This implies that there are now three possibilities for U_a:

1. $U_a \setminus B_1$ is a quorum in $\left((V,Q)^B\right)^{B_1} = (V,Q)^{B_1}$
2. $U_a \setminus B_2$ is a quorum in $\left((V,Q)^B\right)^{B_2} = (V,Q)^{B_2}$
3. Both, i.e. the quorum is outside of both B_1 and B_2.

What we show next is that if 1) is true, then 2) is true too, and vice versa. Therefore, since we have ruled out all other possibilities, the next few lines will show that the *only* possibility left is 3).

If 1) is true, then $(U_a \setminus B_1) \cap U \neq \emptyset$, by QI of $(V,Q)^{B_1}$. In fact, since B_1 is a DSet by construction, $(V,Q)^{B_1}$ has QI. And since $(U_a \setminus B_1)$ and U are both quorums, by definition of QI their intersection is not empty. Now, $(U_a \setminus B_1) \cap U$ happens to equal $(U_a \setminus B_1) \setminus B_2$. Therefore, $(U_a \setminus B_1) \setminus B_2 \neq \emptyset$. If that is the case, then even more $U_a \setminus B_2 \neq \emptyset$. Therefore, by Theorem 1 $U_a \setminus B_2$ is a quorum in $(V,Q)^{B_2}$.

Similarly, if 2) is true, then $(U_a \setminus B_2) \cap U \neq \emptyset$, by QI of $(V,Q)^{B_2}$, since B_2 is also a DSet. Since $(U_a \setminus B_2) \cap U = (U_a \setminus B_2) \setminus B_1$, $(U_a \setminus B_2) \setminus B_1 \neq \emptyset$, and $U_a \setminus B_1 \neq \emptyset$ either. Therefore, by Theorem 1 $U_a \setminus B_1$ is a quorum in $(V,Q)^{B_1}$.

Therefore, 3) is the only possibility left. Knowing this helps a global understanding of all the possibilities, but we have actually done more work than was strictly necessary to prove the theorem. All we need to focus on is 1): $U_a \setminus B_2$ is a quorum in $(V,Q)^{B_2}$.

Now go back to $*$ and follow the same argument for U_b. Look at the possibility 1) that $U_b \setminus B_1$ is a quorum. Following the same argument we can show that this implies that $U_b \setminus B_2$ is a quorum in $(V,Q)^{B_2}$. But then, by QI despite B_2, we necessarily have that

$$(U_a \setminus B_2) \cap (U_b \setminus B_2) \neq \emptyset. \tag{10}$$

If that's true, $U_a \cap U_b \neq \emptyset$ is even more so. Since U_a and U_b are any two quorums in $(V,Q)^B$, we have proven QI despite B. Hence B is a DSET. □

The above theorem may seem "much ado about nothing", but in fact it has shown a rather important general fact, namely that DSets are closed under intersection. We use this fact immediately, in the next theorem.

Theorem 3. *In a FBAS with QI, the set of befouled nodes is a DSet.*

Proof. In the definition of DSet we did not specify whether the nodes contained in a DSet were ill-behaved, blocked, divergent, or correct. We only said that what's left after we take a DSet is a quorum and, in addition, it has QI (if it happens to contain other quorums).

Now let a FBAS contain an arbitrary number of DSets, such that each DSet may contain a number of ill-behaved nodes. Let B_{min} be the specific intersection of these DSets that contains all the ill-behaved nodes. According to Definition 12, a node v is intact iff $v \notin B_{min}$. Therefore, B_{min} is the set of befouled nodes. By Theorem 2, B_{min} is a DSet. □