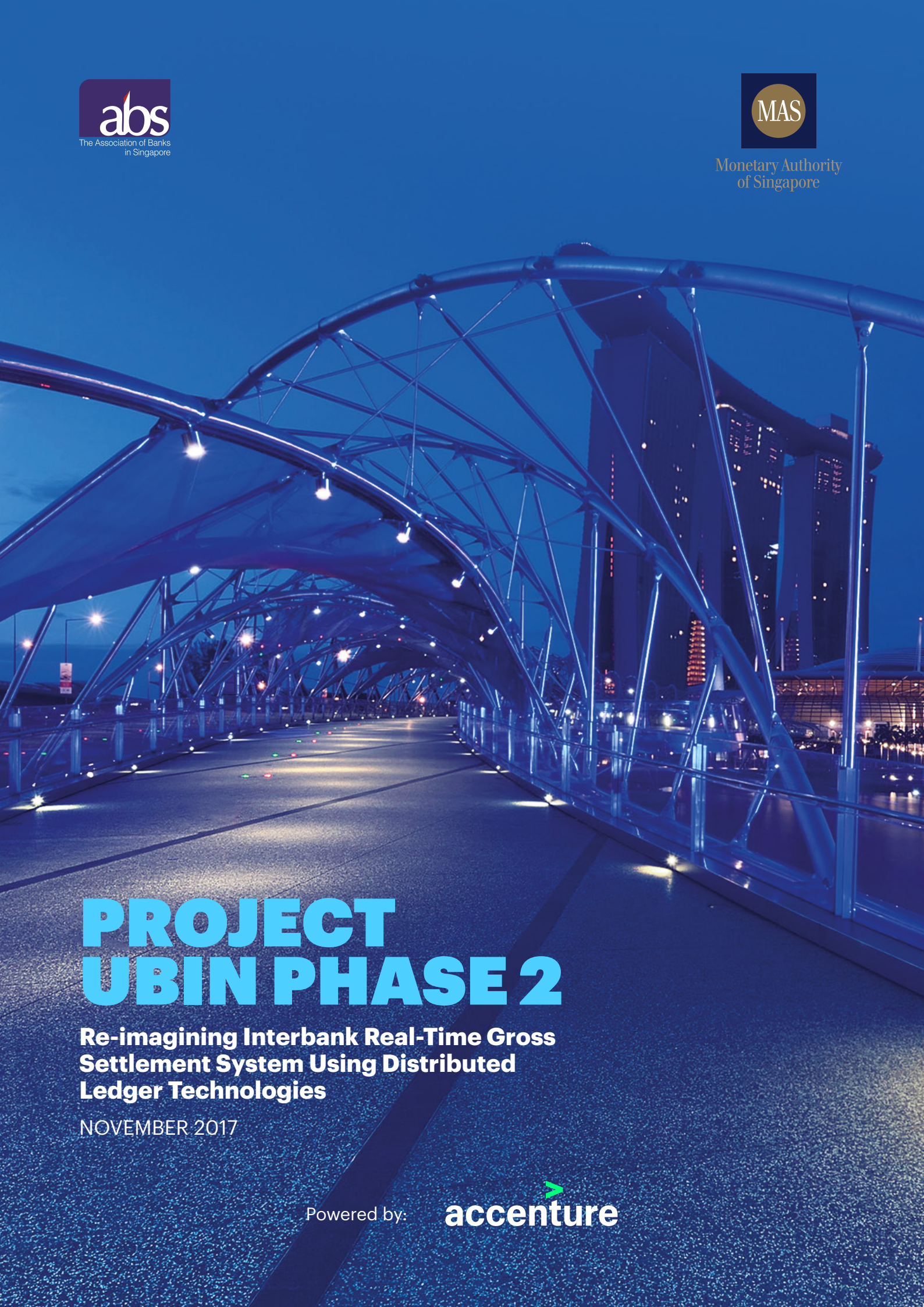


abs
The Association of Banks in Singapore

MAS
Monetary Authority of Singapore

PROJECT UBIN PHASE 2

Re-imagining Interbank Real-Time Gross Settlement System Using Distributed Ledger Technologies

NOVEMBER 2017

Powered by: accenture

FOREWORD

The Monetary Authority of Singapore (MAS) and The Association of Banks in Singapore (ABS) are pleased to present the report "Re-imagining Interbank Real-Time Gross Settlement System using Distributed Ledger Technologies".

The report shares the findings from Project Ubin Phase 2, a collaborative industry project with 11 financial institutions, four technology partners and Accenture.

Project Ubin's journey started a year ago with the development of a basic prototype on Ethereum in Phase 1. This first step in the exploration of Distributed Ledger Technology (DLT) opened up a world of opportunities, while also uncovering new challenges and obstacles.

Phase 2 is focused on solving the key challenges identified around the need for transactional privacy and deterministic finality, and most critically, the ability to perform multilateral netting capabilities in a decentralised manner while preserving transactional privacy.

Prototypes were developed on three leading DLT platforms: Corda, Hyperledger Fabric and Quorum, to explore the different models made possible by the varied capabilities and features of the three DLT platforms.

The successful completion of the project solved a major challenge faced by the DLT community, and opens up opportunity for wider adoption of DLT-based settlement systems. We are sharing our learnings and knowledge from Project Ubin to encourage greater experimentation amongst central banks and financial institutions.

We hope that you would gain a better understanding of developing solutions on the three commonly used DLT platforms, and be inspired to challenge the traditional thinking of centralised architecture, and re-imagine the design of future financial systems.

We would like to express our gratitude and thanks to the 11 participating financial institutions, four technology partners and Accenture, who came together in an open and transparent manner to work on the project. We see immense benefits from a successful industry collaboration that creates a vibrant ecosystem.

We commend this report to you and hope that you will be as excited as we are in its potential to help us provide better services to our customers in a faster, more secure and less costly manner.

Sopnendu Mohanty

Co-Chair, Project Ubin Phase 2
Chief FinTech Officer
Monetary Authority of Singapore

Ong-Ang Ai Boon, Mrs

Co-Chair, Project Ubin Phase 2
Director
The Association of Banks in Singapore

Project Ubin Phase 2 is a collaborative design and rapid prototyping project, exploring the use of Distributed Ledger Technologies (DLT) for Real-Time Gross Settlement (RTGS) System.

Ubin Phase 2 is led by the Monetary Authority of Singapore (MAS) and The Association of Banks in Singapore (ABS). It is managed and delivered by Accenture, with a consortium of 11 financial institutions in Singapore: Bank of America Merrill Lynch, Citi, Credit Suisse, DBS Bank Ltd, HSBC Limited, J.P. Morgan, Mitsubishi UFJ Financial Group, OCBC Bank, Singapore Exchange, Standard Chartered Bank and United Overseas Bank.

CONTENTS

EXECUTIVE SUMMARY

01

Ubin Phase 2 is a collaborative project led by The Monetary Authority of Singapore (MAS) and The Association of Banks in Singapore (ABS).

It is managed and delivered by Accenture, with participation from 11 financial institutions. The 13 week project explores the use of Distributed Ledger Technology (DLT) for specific Real Time Gross Settlement (RTGS) functionalities. Particularly, it focuses on the feasibility of decentralising Liquidity Saving Mechanisms (LSM), while maintaining privacy of banking transactions.

Leveraging the capabilities of the Accenture Liquid Studio and its Liquid Delivery Methodology with Microsoft Azure as the cloud platform, three prototypes were developed by three workstreams on three different DLT platforms: Corda, Hyperledger Fabric and Quorum. The prototypes successfully demonstrate several points. Firstly, that key functions of a RTGS system such as fund transfer, queueing mechanism and gridlock resolution can be achieved through different techniques and solution designs. Secondly, decentralising the key functions of a RTGS system may not only mitigate the inherent risks of a centralised system, such as single point of failure, but may also affirm the promised benefits of DLT, for example cryptographic security and immutability.

Given that privacy is paramount in an interbank payment system, this project validates that privacy of RTGS transactions may be ensured by all workstreams with their distinct methods. Specifically, Corda with its Unspent Transaction Output (UTXO) model and confidential identities, Hyperledger Fabric leveraging its Channels design, and Quorum using Constellation and zero knowledge proofs (ZKP).

Other observations and findings from this project include the scalability and resilience of the three designs. Significantly, this project concludes that all three workstream designs have successfully demonstrated the feasibility of removing a central infrastructure operator in a DLT-based RTGS system. Therefore, with the feasibility of DLT in a RTGS system, the role of MAS as an infrastructure operator in facilitating interbank payments needs to be re-evaluated.

Ubin Phase 2 not only successfully demonstrates that RTGS functions may be decentralised without comprising privacy, but also marks the success and significance of an industry-wide collaboration in laying the foundation for future innovation.

INTRODUCTION 02

2.1 BACKGROUND

2.1.1 Project Ubin

In late 2016, in line with the vision for Singapore to become a Smart Financial Centre, the Monetary Authority of Singapore (MAS) commenced a collaborative project with 11 leading financial institutions and 5 technology providers. It explores the use of Distributed Ledger Technology (DLT) for clearing and settlement of payments and securities. The goal of this endeavour, known as Project Ubin, was for MAS and the financial industry to gain a better understanding of DLT and the feasibility of developing more resilient and efficient alternatives to today's financial market operations and systems.

In Phase 1, a proof of concept was conducted on Ethereum, testing the feasibility of using a central-bank-issued digital currency (SGD equivalent) for interbank payments. Ubin Phase 2, managed and developed by Accenture, assesses the potential implications of deploying DLT for specific RTGS functionalities by focusing on Liquidity Saving Mechanism (LSM). It is also an objective of Phase 2 to understand how real-time gross settlement privacy can be ensured using DLT.

Figure 1: Overall Journey of Project Ubin

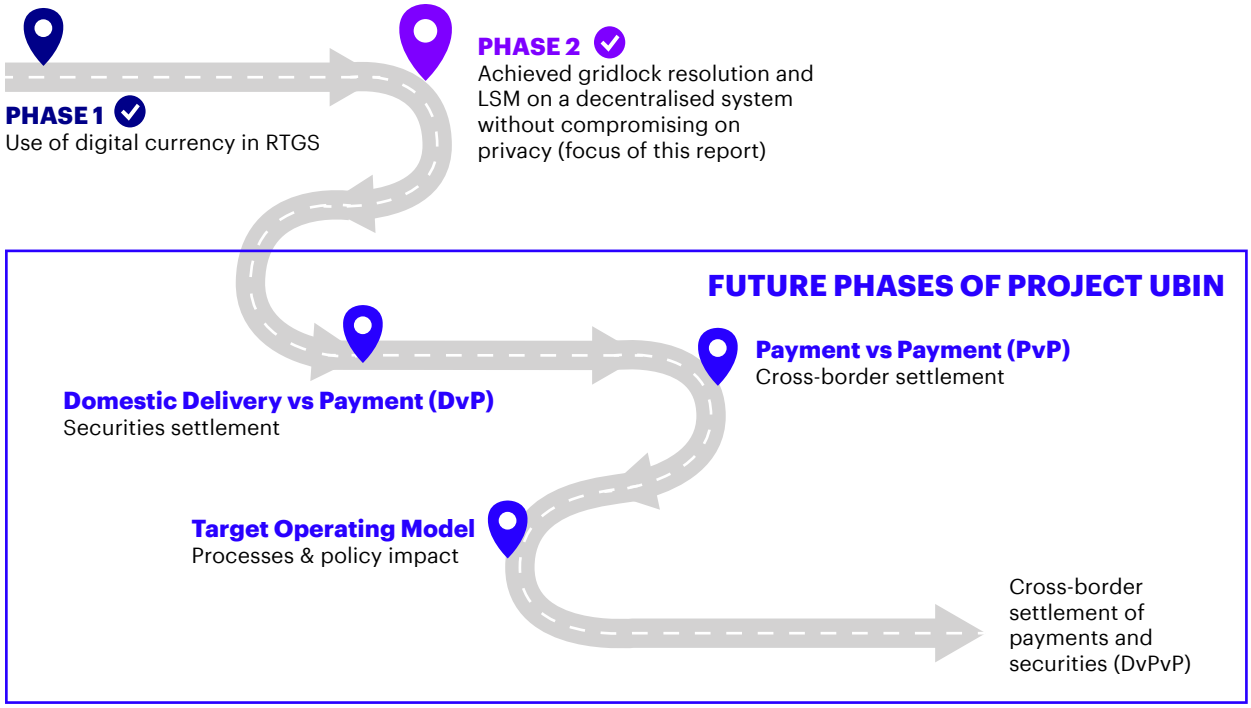
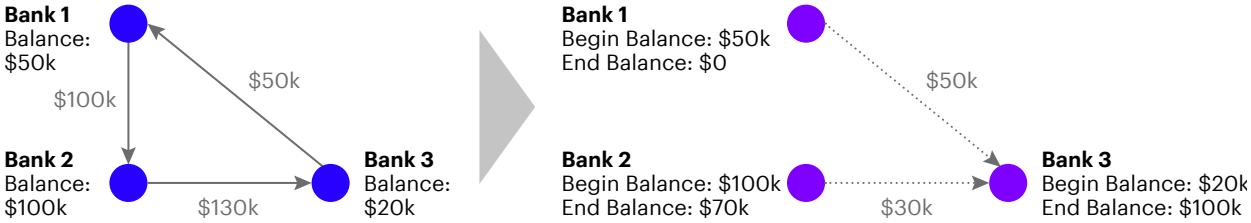


Figure 2: Illustration of a simple gridlock scenario



2.1.2 Real-Time Gross Settlement System

Real Time Gross Settlement (RTGS) systems are typically used for high-value transactions requiring immediate settlement. MAS operates a RTGS system called the MAS Electronic Payment System (MEPS+). MEPS+ plays an integral role in the functioning of Singapore's financial market. It processes about 25,000 transactions a day, with each transaction valued up to SGD 1 billion, and a total daily transaction value of up to SGD 70 billion.[1]

Most RTGS systems around the world are operated on a centralised infrastructure, which is subject to risks such as a single point of failure. The high-value nature of RTGS transactions also demand the system to process transactions seamlessly to reduce intraday liquidity gridlocks in the financial market. A key function in RTGS systems is the ability to resolve payment gridlock through a Liquidity Saving Mechanism (LSM).

Figure 2 illustrates a simple gridlock scenario where three payment instructions, $50,000, $100,000, and $130,000 are unable to settle (in a sequential manner) as each sender bank does not have sufficient funds. The purpose of a LSM is to resolve the gridlock scenarios when these transactions are executed on a net basis. LSM is traditionally performed by a centralised system as the algorithms for LSM typically requires a consolidated, single view of all payment instructions in the system.

Today, MAS plays both the roles of an infrastructure operator and an overseer of the MEPS+. The latter sections in the report refer to MAS as having both roles but mostly as the infrastructure operator.

2.1.3 Distributed Ledger Technology (DLT)

At its most basic, Distributed Ledger Technology (DLT) is a general purpose data technology that allows different actors to share access to the same data with confidence. Participating actors can trust that the data has not been tampered with and can control access to the data. DLT was arguably born out of 1920s/30s' cryptography and has been widely popularised by the introduction of the Bitcoin in 2009. Today, the technology has evolved to solve different business problems.

[1] http://www.mas.gov.sg/singapore-financial-centre/payment-and-settlement-systems/clearing-and-settlement-systems/meps/meps-plus-statistics.aspx as reference for section 2.1.2 "It processes about 25,000 transactions a day, with each transaction valued up to SGD 1 billion, and a total daily transaction value of up to SGD 70 billion

DLT is of particular interest to the financial sector and traditional centralised clearing and settlement systems for several reasons. Firstly, it has the potential to increase the reliability and traceability of information stored in a decentralised network. This decentralised processing and storing of information potentially mitigates the single point of failure present in the current centralised system.

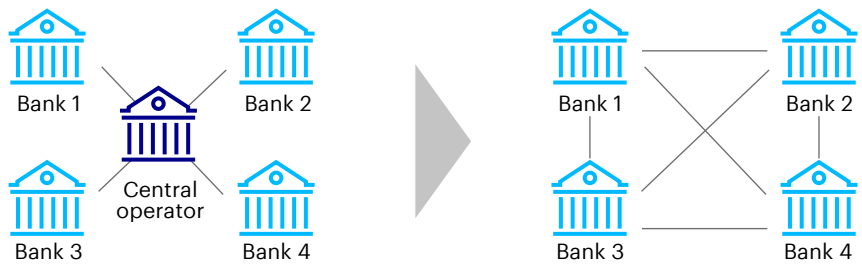
Moreover, strict rules are embedded within DLT on how ledgers are governed and recorded. Multiple parties must come to a consensus on the legitimacy of a transaction before it can be recorded in the distributed ledger. This helps to reduce or eliminate the need to reconcile transactions, since the data has been agreed and attested to by all or multiple parties.

Lastly, where privacy and confidentiality are paramount in the financial sectors, additional protocols and enhancement of the technology will better enable the decentralised information to be private or restricted. Figure 3 is an illustration of how DLT may potentially disintermediate the central operator in a centralised RTGS system as the settlement processes and ledger are distributed among the different banks in the network.

Implementing a Real-Time Gross Settlement (RTGS) system on Distributed Ledger Technology (DLT) poses the challenge of preserving transactional privacy while processing a traditionally centralised Liquidity Saving Mechanism (LSM) in a decentralised manner.

LSM is a key functionality in a RTGS system which eliminates transaction gridlocks to maximise overall liquidity in the payment network. LSM is typically processed centrally as the algorithm requires a consolidated view of all payment instructions in the system.

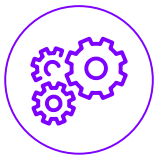
Figure 3: Transition from a RTGS with central operator to a decentralised RTGS



2.2 OBJECTIVES AND APPROACH OF UBIN PHASE 2

The objective of Ubin Phase 2 was to develop three prototypes with specific RTGS functionalities. Each prototype is developed on a different DLT platform: Corda, Hyperledger Fabric and Quorum running on a common cloud platform – Microsoft Azure. A key functionality showcased in Ubin Phase 2 is the ability to execute Liquidity Saving Mechanisms (LSM) without compromising privacy in a decentralised network. The prototypes are developed to address the following six key criteria:

Digitalisation of Payments
Central Bank Digital Currency (CBDC) with real-time gross settlement capabilities



Decentralised Processing
Distributed and resilient infrastructure with no single point of failure

Payment Queue Handling
Uniform queueing system with prioritisation, holding and cancellation facilities

Privacy of Transactions
Only relevant parties will have visibility to transaction details

Settlement Finality
Final and irrevocable settlement of payment instructions with deterministic finality

Liquidity Optimisation
Implement netting and gridlock resolution algorithms to maximise liquidity efficiency

Ubin Phase 2 started with design workshops in July 2017. There were three workstreams with each representing a platform: Corda workstream, Hyperledger Fabric workstream and Quorum workstream. The workstreams were tasked to design a solution on a common set of functionalities, with the goal of conducting an objective assessment on the three platforms. It is not the intent of this proof of concept to draw a quantitative comparison among the three workstreams.

Based on the different workstream designs, Accenture's rapid prototyping team from Singapore Liquid Studio developed and delivered three prototypes over the course of three Agile sprints. A common user interface for these prototypes was also developed in line with the goal of an objective evaluation across the three platforms.

This report summarises the outcome from this project and is structured as follows: Section 3 provides the overall scope and approach of the project, as well as highlights key characteristics and design principles of the three platforms. Section 4 describes the functional design, focusing on three key functional areas: Fund Transfer, Queue Mechanism and Gridlock Resolution. Section 5 summarises the findings on the three workstreams in relation to privacy, scalability and performance, resiliency and finality. Section 6 describes six future considerations noted by the project participants throughout the project. Finally, the report closes with a conclusion in Section 7.

In addition to this report, a detailed technical documentation with the technical design of all three prototypes is published along with the source code.

SCOPE AND SYSTEM DESIGNS

03

3.1 OVERALL FUNCTIONAL SCOPE

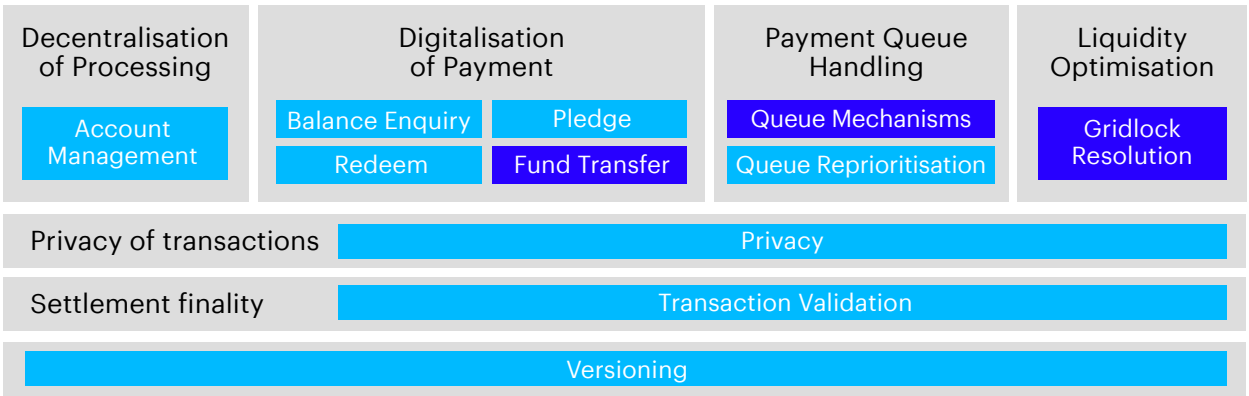
The functional scope of this proof of concept is categorised into 11 epics, addressing the specific key criteria as described in Section 2.2. This section describes the key functions of applying DLT to the specific RTGS functionalities.

A key functional requirement in Ubin Phase 2 is to execute fund transfer in a DLT network that enables decentralisation and digitalisation of payments. This includes a queueing mechanism that allows payment instructions to be prioritised for processing, as well as determining whether payment instructions can be immediately settled, deferred for future processing or cancelled. In a straightforward scenario, eligible payment instructions are settled as debits from the sender and credits to the receiver. However, complex situations may arise when payment instructions are ineligible due to insufficient funds and these are transferred to the waiting queue, where they can be reprioritised. These complex situations may result in a gridlock situation, whereby outgoing payments cannot be fulfilled unless an incoming payment is received or a gridlock resolution is triggered.

All the transactions performed within the DLT require strict privacy, and are validated to ensure the transaction is final and legitimate.

Figure 4 is an illustration of the functional scope of Ubin Phase 2. This report focuses on three key features (i.e. Fund Transfers, Queue Mechanism and Gridlock Resolution) as highlighted. Detailed design considerations for all epics and user stories are available in the technical documentation that is published along with the source code.

Figure 4: Functional scope of Project Ubin Phase 2



3.2 WORKSTREAM CHARACTERISTICS AND DESIGN CONSIDERATIONS

This section describes some of the key characteristics and components of each DLT platform used in the three workstreams. Subsequent sections will discuss the implications of these characteristics and components on the design of the overall solution.

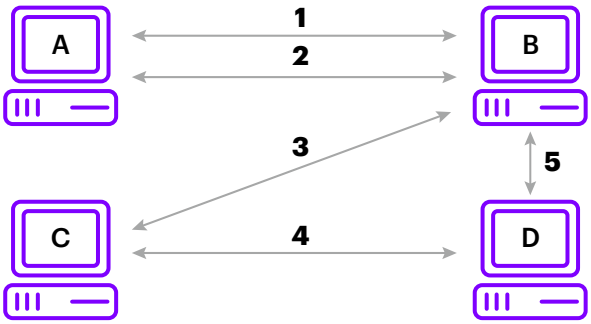
3.2.1 Corda

Corda is a distributed ledger technology platform designed for use with regulated financial institutions. It is inspired by blockchain systems and is designed for recording, managing and synchronising commercial agreements between known and identified parties at scale without compromising privacy.

The key differentiating factor for Corda is the UTXO (unspent transaction output) model where data records, representing asset units or deals, reference to previous versions of that data by transaction hash. Hence, creating an immutable chain of provenance and lineage. When a transaction is committed to the ledger, the "consumed" UTXOs are considered historic (i.e. “Spent”), and one or more new UTXOs are created. When a data record is involved in a transaction, the Corda node will ‘soft lock’ that record so that it will not be able to be used in another transaction at the same time, therefore allowing parallelisation. In addition, Corda uniquely leverages on a Notary service to prevent double spending of a UTXO state in a distributed ledger environment.

Corda distributes the ledger based on a need-to-know basis instead of a global broadcast, similar to a multiple peer-to-peer model. For example, in the illustration below, transactions 1 and 2 are only visible to node A and B, transaction 3 is only visible to node B and C, transaction 4 is only visible to node C and D, and transaction 5 is only visible to node B and D.

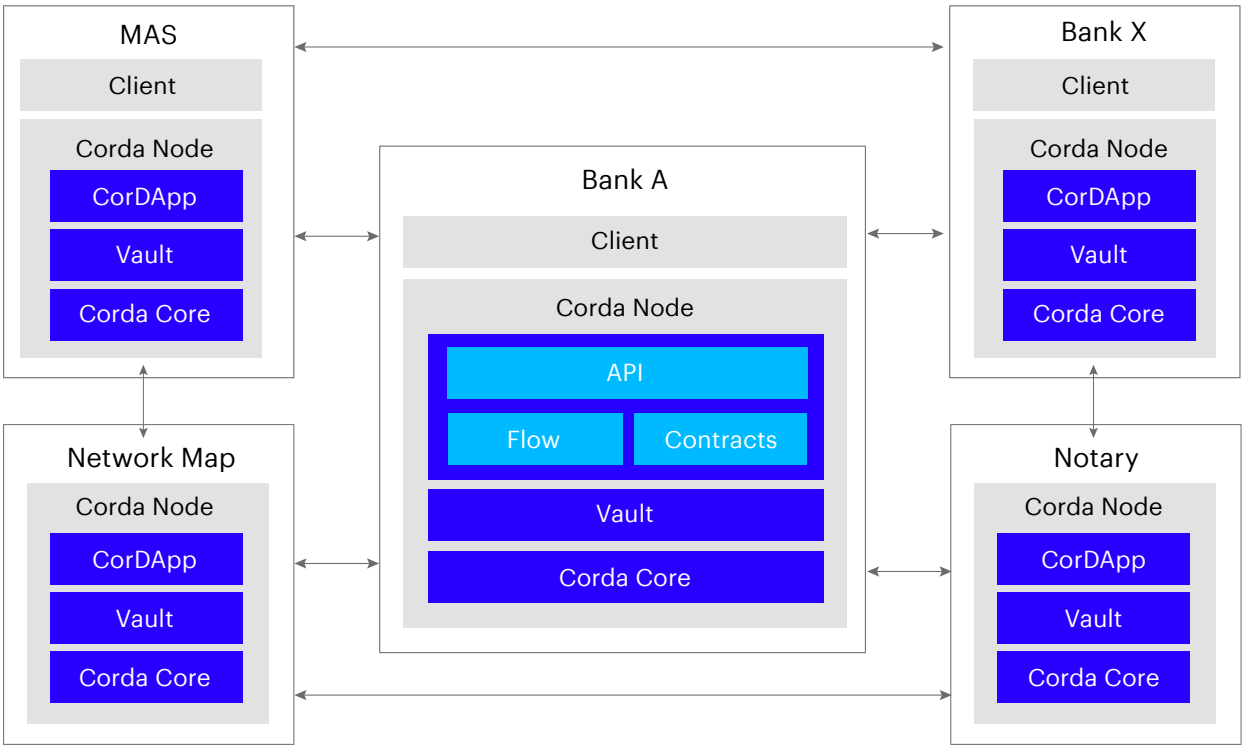
Figure 5: Illustration of five transactions between four banks in a Corda network



In this project, the prototype on Corda was developed using *Corda version 1.0*, leveraging Kotlin as the main programming language. Some of the design concepts employed in Ubin Phase 2 are as follows:

- A representation of Cash for fund transfer whenever a sender has sufficient funds
- A representation of an Obligation from a sender to a receiver to be paid in cash in the future. This is used to represent pending or queued payment instructions when a sender has insufficient funds to settle at the point of initiating the fund transfer
- A vault for each node to hold all of its Cash and Obligation states
- Both Cash and Obligation use the UTXO state model to represent the disposition of the object
- Confidential Identities create a new unique pair of public key and certificate to be exchanged between the sender and receiver for every transaction, ensuring transaction anonymity.

Figure 6: Architecture components of Corda workstream design



For the Corda prototype, each Corda node (banks, MAS and Notary) is hosted in individual Azure virtual machines as illustrated in Figure 6.

Key components in the Corda design include:

- **Corda Distributed App (CorDApp):** A Distributed Application installed at the node level which leverages on Corda's platform to handle business logic and process. CorDApps are made up of Flows, Contracts, States and APIs.
- **Network Map Service:** Manages and publishes the well-known public keys and corresponding physical IP addresses, so nodes on the network can be identified and reached. This Network Map service can be distributed and run by an independent party.
- **Notary Service:** Provides uniqueness and/or validating consensus on received transactions by providing signature to indicate transaction finality. In Ubin Phase 2's prototype, the notary only provides uniqueness consensus. Notary node (or cluster of nodes) can be run by an independent party and not necessarily the central bank.

Corda distributes the ledger based on a need-to-know basis instead of a global broadcast, similar to a peer-to-peer model. Corda's UTXO (Unspent Transaction Output) model creates an immutable lineage chain of transaction states (history), by referencing one or more inbound transaction by its hash for every outgoing transaction.

3.2.2 Hyperledger Fabric

Hyperledger Fabric is a platform for distributed ledger solutions, underpinned by a modular architecture aimed to deliver high degrees of confidentiality, resiliency, flexibility and scalability. It is intended to allow for pluggable executions of various parts and suits the multifaceted nature and complexities that exist across the economic ecosystem. For this project, the prototype was developed using *Hyperledger Fabric version 1.0.1*, utilising the Go programming language for smart contracts (chaincodes) and Node.js for the application layer.

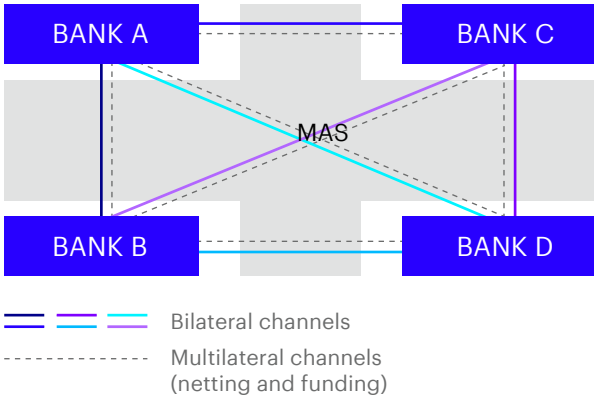
Hyperledger Fabric offers the capability to create channels, enabling a gathering of participants to share a ledger of transactions that are only privy to these participants. This allows isolation of confidential data and ensures the shared ledger is accessible only among the need to-know participants. This is an alternative to systems where a few members may be contenders and do not need transactions to be known to competing members.

Hyperledger Fabric prevents double spending by having peers (i.e. participation nodes) validate the transactions against the endorsement policy to ensure correct allotment and authentication of the signatures. The endorsement policy is defined per chaincode to determine the number of endorsements and signatures (from the endorsing peers) required per transaction. Peers will also perform a versioning check to ensure data integrity. The Orderer receives endorsed transactions, packages them into blocks and broadcasts to all participants in the channel. The participants in the channel then validate these transactions before committing them to the ledger.

To maintain the privacy of fund transfers between banks, a bilateral channel is created for every pair of banks present in the network. In the example of a four banks network, Bank A would have three bilateral channels, one each with Bank B, Bank C and Bank D. In each bilateral channel, the bank will maintain its channel-level account which allows for the flexibility to assign a fixed amount of liquidity for transacting between a particular counterparty. For ensuring transactional validity, the endorsement policy for all bilateral transactions is set to require both participating banks in each bilateral channel to endorse before committing into the ledger.

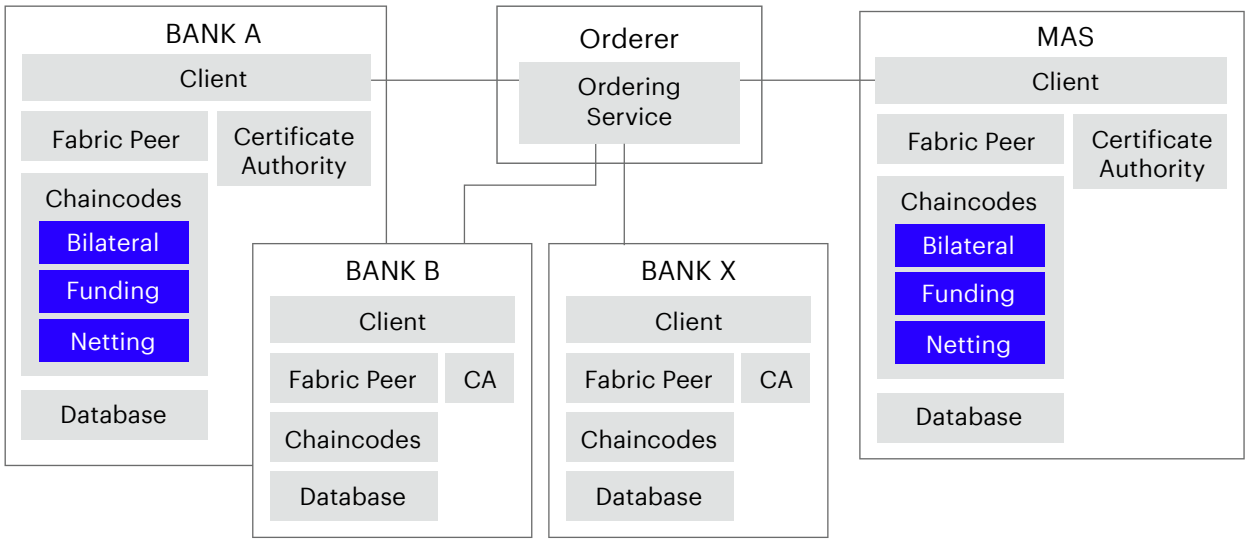
Additionally, all banks participate in two multilateral channels: funding channel and netting channel. An illustration of how a network with four banks would look like is as follows:

Figure 7: Example of bilateral and multilateral channels in a Hyperledger Fabric network with four banks



The funding channel is used for banks to perform fund movements across the channel-level accounts in their bilateral channels. This multilateral channel will allow for auditability and traceability of such transactions. Both participating banks from source and target channels are required to provide their endorsement for cross-channel fund movement to ensure the legitimacy of the fund.

Figure 8: Architecture components of Hyperledger Fabric workstream design



The netting channel facilitates gridlock resolution. Given that gridlock resolution may involve all network participants and all must agree on the final outcome, the endorsement policy is set to require endorsement from all network participants.

The Hyperledger Fabric prototype includes MAS as a participant in all channels, both bilateral and multilateral, providing MAS with the ability to audit and track all transactions in the Fabric network. Although MAS is not directly involved in any bilateral transactions, MAS plays a special role in the settlement of a gridlock resolution cycle to process and endorse the settlement transactions once a successful gridlock resolution solution is found.

For the Hyperledger Fabric prototype, each Hyperledger Fabric node (banks, MAS and Orderer) is hosted in individual Azure virtual machines as illustrated in Figure 8.

Key components in the Hyperledger Fabric design includes:

Chaincode: A set of codes that defines the business logic and is executed against the ledger's current state database

Orderer: One or many orderers form the ordering service which provides a shared communication channel to clients and peers

Certificate Authority: A component that governs identity registration and certificate management

Fabric Peer: Receives ordered state updates in the form of blocks from the ordering service and maintains the state and the ledger

Hyperledger Fabric offers the capability to create channels, enabling a gathering of participants to share a ledger of transactions that are only privy to these participants. A bilateral channel is created for every pair of banks present in the network to maintain the privacy of the transactions to only the sending and receiving banks.

3.2.3 Quorum

Quorum is created for the financial services industry as an Ethereum-based distributed ledger that supports transaction and contract privacy. Quorum shares private information in a point to point manner on a need-to-know basis. In addition to privacy, Quorum adds further enterprise-centric features on top of Ethereum such as transaction finality, performance benefits and network permissioning. Since it is designed to operate in permissioned networks, Quorum removes Ethereum's Proof-Of-Work and Proof-Of-Stake consensus mechanisms, replacing thesewith a selection of voting-based consensus mechanisms that users can choose from. It is a fork of the Go Ethereum client(geth), and is designed to be developed in line with future geth releases.

In Ubin Phase 2 design uses Quorum's Raft consensus mechanism, a formally verified consensus mechanism that is based on the etcd Raft implementation that underpins widely used software such as Kubernetes. Raft provides faster blocktimes (in the order of milliseconds instead of seconds) and transaction finality (the absence of forking). In addition to Raft, Quorum Constellation provides transaction privacy. Further the design implements Zero Knowledge Security Layer(ZSL), a protocol designed by the team behind Zcash, that leverages zk-SNARKS – a variant of zero-knowledge proofs (ZKP) – to enable the transfer of digital assets on a distributed ledger without revealing any information about the Sender, Recipient, or quantity of assets that are being transferred, and without requiring any central party to affect the transfer.

For this project, the prototype was developed using *Quorum version 1.5 (Ethereum version 1.5 with Constellation version 0.1.0)* with Solidity as the programming language and Node.js for the application layer.

All nodes in a Quorum network run the same set of components; there are no centralised or function-specific nodes that require different configurations. This ensures that the network is truly decentralised with no single point of failure.

Key components in the Quorum design includes:

- **Quroum Node:** A fork of the Go Ethereum client (geth)
- **Constellation:** Manages transaction privacy and holds the inventory of the encrypted "Unconfirmed" and "Confirmed" payment transactions
- **Smart Contracts:** Network-wide (i.e. public) smart contracts and private smart contracts
  - Public contracts (available to all participants): Netting & ZKP validation
  - Private contracts (for involved parties only): Payment transfer and bank balance visibility
- **Zero Knowledge Security Layer (ZSL):** Provides decentralised privacy during Liquidity Saving Mechanism execution and payment transfers
- **Quorum Decentralised App (DApp):** The DApp orchestrates various payment execution functions in the smart contracts and generates zero knowledge proofs (ZKP). It also functions as a RESTful API layer and a smart contract event listener

For Ubin Phase 2, each Quorum node (banks and MAS) is hosted in individual Azure virtual machines as illustrated in Figure 9.

Zero knowledge proofs (ZKP) is a key feature introduced as part of Ubin Phase 2 in terms of preserving privacy in the absence of a central party. ZKP proves that a bank has sufficient balance to make a payment without exposing the actual balance to the rest of the network. The entire network will validate this fact, all whilst ensuring that no information about the sender, receiver or the assets that are being transferred is ever revealed to the network. This network-wide, privacy-preserving validation removes the need for a central authority to validate balances and transactions.

These proofs are created off chain by submitting hashed values of the initial balance, transaction amount, and final balance to a proof generator. After the proofs are created, they are submitted for verification on chain. Once both the sender's and receiver's proofs have been verified by the other nodes on the Quorum network, the transaction is then confirmed by updating the balances, and the payment instruction is moved into a completed state.

Quorum uses zero knowledge proofs (ZKP) to enable the transfer of digital assets on a distributed ledger without revealing information about the Sender, Recipient, or quantity of assets. Additionally, Quorum uses a voting-based Raft consensus mechanism, replacing Ethereum's Proof-Of-Work and Proof-Of-Stake consensus mechanisms.

Figure 9: Architecture components of Quorum workstream design

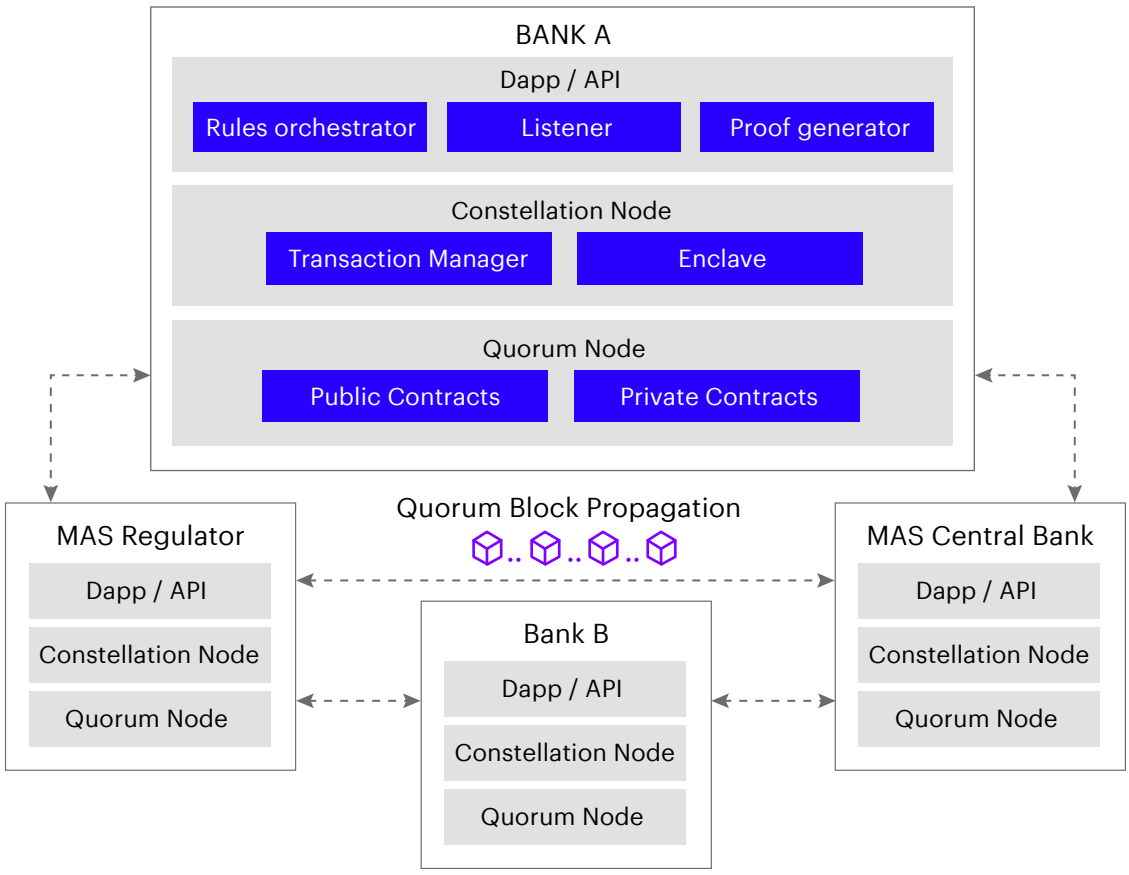
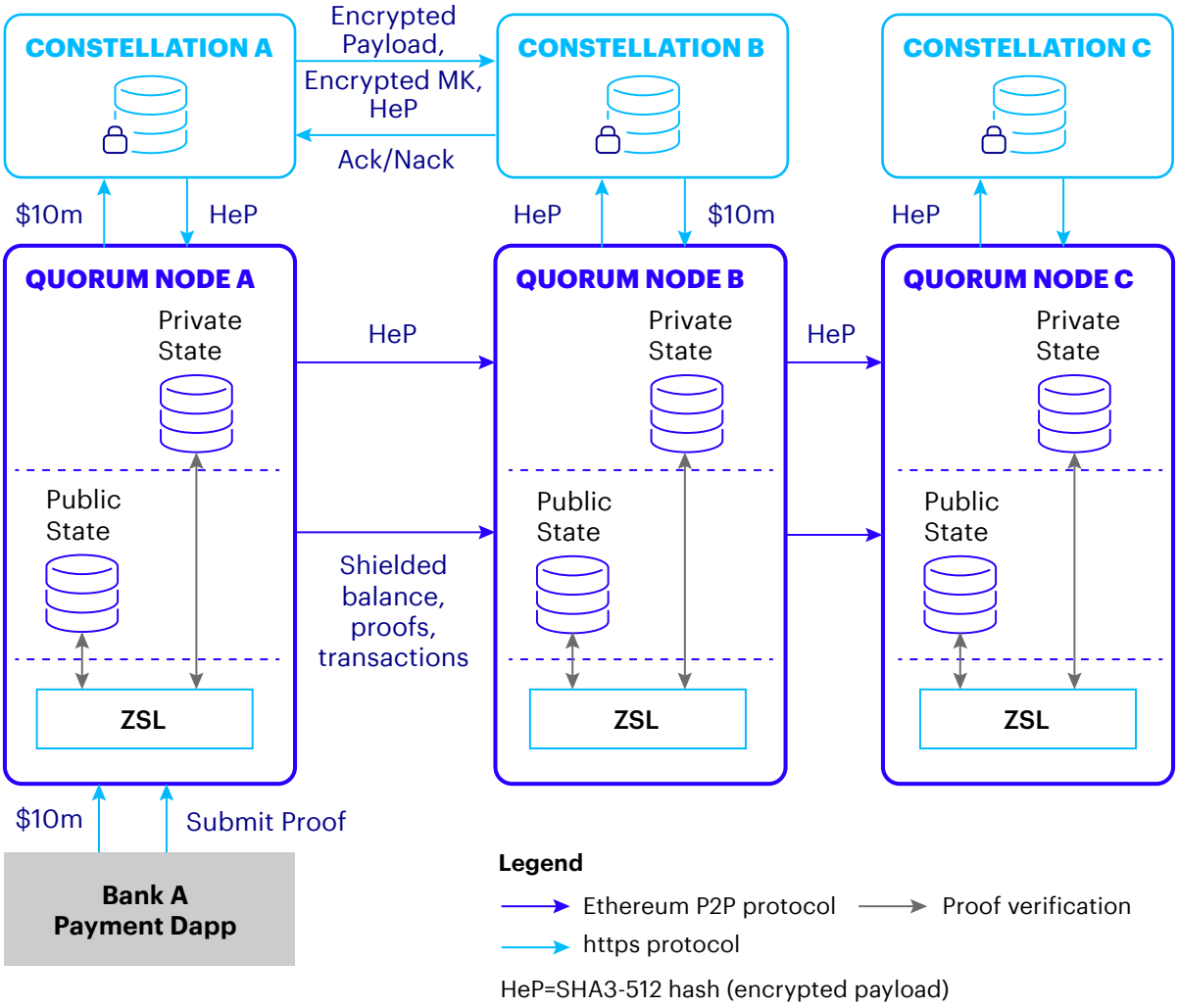


Figure 10: Illustration of Quorum transaction privacy



KEY FUNCTIONAL DESIGN

04

4.1 FUND TRANSFER

In Ubin Phase 2, fund transfer refers to a payment instruction to send funds from one bank to another. The payments are settled immediately on the basis that the sender has sufficient liquidity and has no pending payment instructions in its outgoing queue (refer to Section 4.2).

4.1.1 Corda

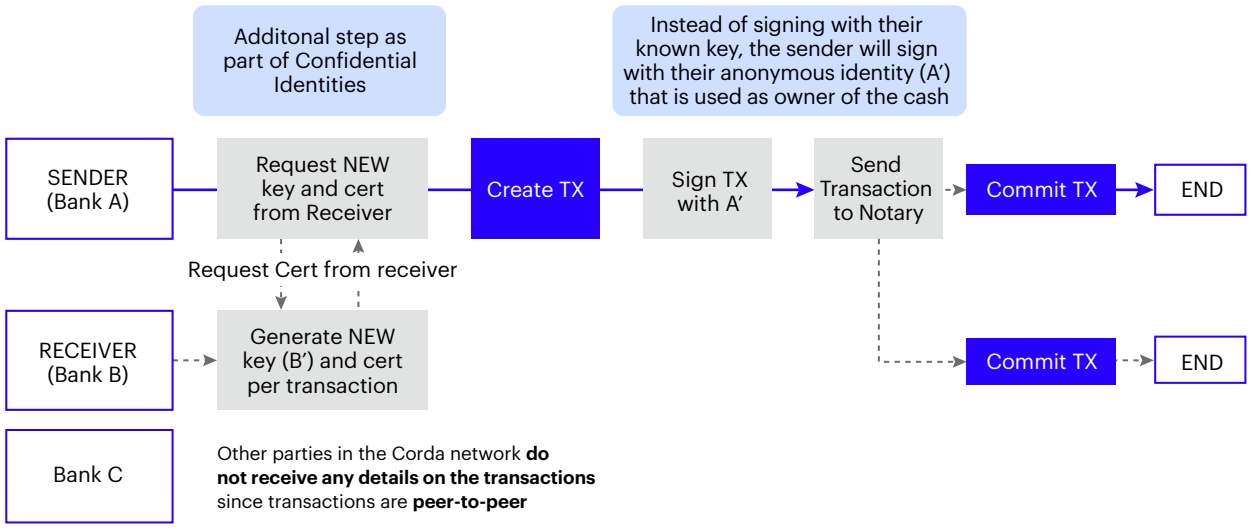
In Corda's design, a fund transfer executes a 'Transfer' flow through a peer-to-peer approach where only the sender and receiver banks will process, validate and record the transaction.

With Confidential Identities, the sender would request for a new, unique pair of public key and certificate from the receiver of the payment instruction.

This anonymous identity is only known to both sender and receiver. This helps to shield the parties that are involved in payment instruction so that future owners of the asset are not able to identify the previous owners. This is important to preserve privacy in the UTXO model where the chain of custody of the asset needs to be validated all the way back to issuance of the asset by MAS. With Confidential Identities, the receiver can validate the provenance of the funds all the way back to issuance, but is not able to ascertain the real-world identities of the historical owners.

In Figure 11, a sender initiating the transfer of funds requests a key and certificate from the receiver, before constructing a new transaction. Both the sender and receiver involved in the transaction would use their anonymous identities in the transaction.

Figure 11: Transaction flow of Fund Transfer in Corda



The public keys of these Confidential Identities are used in the transaction when generating the output states, commands and signing of the transaction. After the sender signs the transaction with its anonymous identity that is currently used as the owner of the cash, the notary verifies the uniqueness of the states and imprints its signature as well. Upon notarisation, both the sender and receiver would commit the final transaction with its output states to its respective ledgers. This approach allows the privacy of the transaction to be maintained in real time and in future transactions where the lineage of states used as input states do not reveal the identities of the past participants.

In this process, confidential identities ensure that only the sender and receiver are aware of the parties in the payment instruction, while allowing the transaction to remain visible.

Alternatively, in the 'Transfer' flow, if a sender party does not have sufficient funds for an immediate settlement of funds transfer, an Obligation state will be created in an alternative flow. Obligations will be registered into a persistent queue (refer to Section 4.2.1) maintained by the sender party which can be cancelled, re-prioritised, or otherwise settled when funds are available or processed through gridlock resolution (refer to Section 4.3.1).

The Notary functions as a service that accepts transactions submitted to them for uniqueness validation. The Notary would either return an accepted transaction with a signature or a rejection error that states a double spend has occurred.

4.1.2 Hyperledger Fabric

A fund transfer is executed in the bilateral channel between the sender and the receiver. If there are sufficient funds in the sender bank's channel-level account and no queued outgoing payment instructions with equal or higher settlement priority than the newly submitted one (refer to Section 4.2.2), the payment instruction will be settled immediately i.e. decrease the balance of sender's channel-level account and increase the balance of receiver's channel-level account. Otherwise, the sender bank will attempt to execute bilateral netting against the payment instructions in its incoming queue. Bilateral netting can be illustrated as such:

- Bank A performs a fund transfer of $5,000 (Transaction 1) to Bank B. However, its channel-level balance is $1,000 and so it has insufficient funds to settle the payment instruction
- In the meantime, Bank A has an incoming payment instruction of $5,000 (Transaction 2) from Bank B
- Given Bank B has a $5,000 payment instruction in queue to Bank A, bilateral netting of both Transactions 1 and 2 result in a net value of 0 and both transactions can settle accordingly

In the bilateral channel design, both banks have visibility over both of their channel-level accounts, as well as the queues between them. By maintaining a channel-level account, each bank can control the amount of liquidity dedicated per counterparty for transactions. As such, the bilateral channel design inherently supports bilateral netting to attempt to settle queued payment instructions bilaterally prior to attempting to settling multilaterally in the network with gridlock resolution.

If there are insufficient funds and bilateral netting is not possible, the payment instruction is added to the queue which is described further in Section 4.2.2.

Figure 12: Transaction flow in Hyperledger Fabric

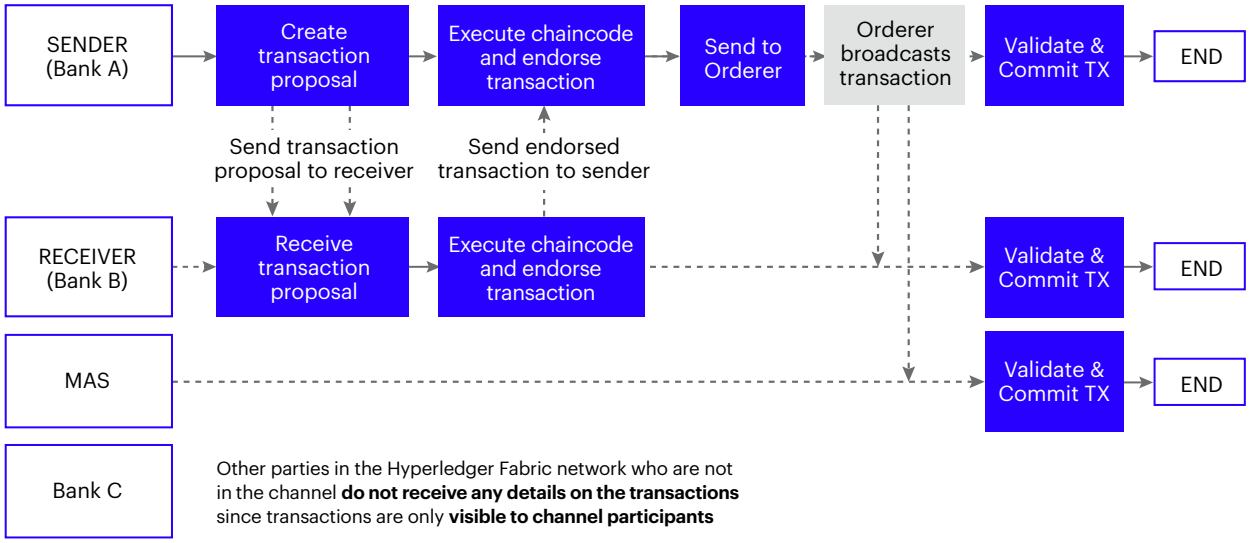


Figure 12 illustrates the transaction flow in Hyperledger Fabric – this transaction flow is similar for other functionalities in the Hyperledger Fabric prototype.

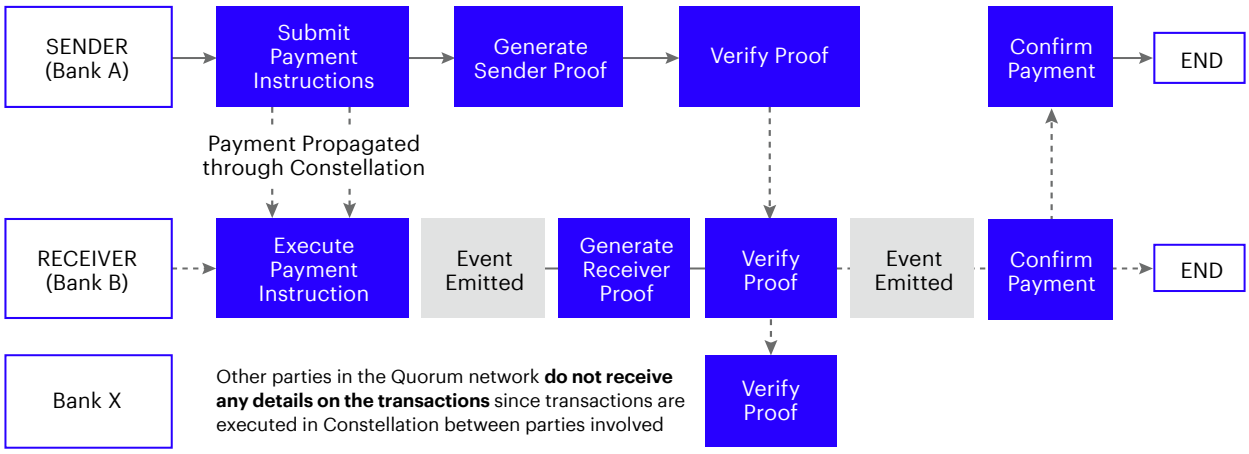
As per the diagram, the sender sends the transaction to the receiver. Both the sender and receiver endorse the transaction before the sender sends it to the orderer for broadcasting. All participants in the channel (sender, receiver and MAS) will receive a block to validate and commit the transaction to their ledgers.

4.1.3 Quorum

In Quorum's design, funds transfer is executed privately between two parties, with no other party seeing the details of this. Balance validation is performed by the entire network through the use of zero-knowledge proofs. To achieve this, both private and public smart contracts are utilized: private contracts allow for the bilateral payment transactions between two parties. Public contracts store shielded salted balances for each of the participants in the network. For every transaction, the sender and receiver generate Zero-Knowledge-proofs and submit the same for verification by the entire network. Once the network verifies the proofs, the shielded salted balances for the sender and receiver are updated in the public contract. The transaction is marked as complete on the private contract only after the completion of the proof verification and update of shielded salted balance. The decentralised application (DApp) functions as the orchestrator of the payment process.

In Figure 13, a payment instruction for fund transfer is initiated from the sender's DApp. The DApp invokes the private contract to generate a private transaction. The sender's DApp then invokes a public transaction which will be executed by all nodes on the Quorum network. The public transaction is created with the hash of payment instruction amounts which will be used as inputs to generate and verify proofs. The hashing of payment instruction amounts leverages on SHA-256 algorithm with a unique salt dynamically generated per transaction. Hashing is done to maintain data privacy since public transactions are propagated to all nodes in the network.

Figure 13: Transaction flow of Fund Transfer in Quorum



The validity and integrity of the public transaction is verified by using zero knowledge proofs (ZKP). Both sender and receiver generate proofs to show that no unauthorised funds have been introduced or taken out of the system. The proofs are generated based on the parties' balances and amount transferred. Upon submitting a payment instruction, the sender's DApp triggers its Quorum node to generate proof that takes in a simple mathematical formula of *starting balance – amount sent = ending balance*. At the same time, the receiver's DApp also begins generating its proof with a different mathematical formula of *starting balance + amount received = ending balance*.

Verification of proofs are public and performed by all nodes on the network. Because of ZSL, this activity does not require any data to be revealed. Once verification of the sender's and receiver's proofs is complete, the receiver's DApp will execute a private function to move the transfer amount from sender and receiver in a single atomic transaction.

To achieve Fund Transfer functionality while maintaining privacy:

- Corda workstreams leverages Confidential Identities to generate and exchange a new unique pair of public keys between the sender and receiver for each transaction. This helps to shield both sender and receiver identities from other participants including past participants.
- In Hyperledger Fabric design, fund transfer is executed in the bilateral channel between the sender and the receiver. The bilateral channel also allows banks to maintain channel-level balance with each counterparty bank, as well as enable bilateral netting of gridlock transactions between each pair of transacting banks.
- Quorum design requires both private and public smart contracts where private contracts allow bilateral transactions between two parties and public transaction with the hash of transaction amounts to generate and verify using zero knowledge proofs (ZKP).

4.2 QUEUE MECHANISM

When a bank creates a payment instruction for a fund transfer but has insufficient liquidity, the payment instruction is put into a queue. The bank has visibility of all its outgoing and incoming payment instructions in the queue.

When the bank obtains sufficient liquidity, the queued transactions are settled automatically, based on the following sequence:

Priority – There are two levels of priority—Normal and High—for each payment instruction. Payment instructions with High priority will be settled ahead of any payment instruction with Normal priority, regardless of creation time.

First-In, First-Out (FIFO) – For payment instructions within the same priority level, the oldest payment instruction based on creation time will be settled first.

The following are options for the sending bank to perform on the payment instructions in its outgoing queue:

- **Change** the priority of an unsettled payment from Normal to High, and vice-versa
- **Cancel** an unsettled payment – Removes payment instruction from the outgoing queue
- Put an unsettled payment **on-hold** – Payment instruction will remain in the outgoing queue and will not be settled or included in gridlock resolution
- **Activate** an on-hold payment – Payment instruction will be settled upon sufficient liquidity or included in gridlock resolution

If the outgoing queue contains Active payment instructions of any priority, a new payment instruction of Normal priority will automatically be appended to the outgoing queue. This is regardless of the bank's liquidity at the point of creation of this new payment instruction. However, if a payment instruction of High priority is created and there is no other High priority payment instruction in the outgoing queue, it will be settled immediately if the bank has sufficient liquidity. If there are other High priority payment instructions in the outgoing queue, this new High priority payment instruction will be added between the newest High priority and the oldest Normal payment instructions, if any.

4.2.1 Corda

During a 'Transfer' flow, a payment instruction is initiated. Where there are insufficient funds in the sender's balance, there will be an issuance of an Obligation state in the ledgers of both the sender and the receiver. Similar to a fund transfer, Confidential Identities are used where a new key and certificate would be created for the transaction. This key and certificate would then be exchanged between the sender and receiver. The new keys generated would be used to identify the participants of the Obligation state. The Obligation state would be issued as output of the transaction, with the transaction signed and sent to the receiver. If the receiver responds with the verified and signed transaction, the Obligation state details (i.e. linearId) would be put in the persistent queue of the sender party. If not, the transaction would be cancelled and the Obligation state would not be issued.

Each Obligation state represents a "pending" fund transfer or an outgoing unsettled payment instruction that will be settled in the future. These Obligation states' details are also replicated and maintained by the sender in its persistent priority-queue to ensure that the node will be able to recover the information in the event of any node restart. Each of the Obligation states in the queue is tagged with a priority level that can only be changed by the sender. The priority level is visible only to the sender as the queue is local to the sender party, and not shared with other nodes to ensure privacy.

The queue maintains a FIFO (first-in, first-out) order of sequence. Each virtual machine contains a scheduled processing function that periodically triggers a settlement API to attempt to settle Obligation states, if possible, based on the current funds available. The queue settlement logic will iterate through the queue of Obligation states to settle in sequence by the next active, highest priority and oldest obligation first. By changing the priority level, the order by which the Obligation states are settled changes as well, according to the settlement logic. The queue also maintains the statuses of each of the Obligation states. An Obligation can be put on-hold to exclude them from participating in any settlement or gridlock resolution. They could be reactivated by changing their status back to 'active' in the queue again, allowing them to participate in any future settlements and gridlock resolution.

All three platform designs allow maintenance of a transaction queue including changing priorities, cancelling, putting on hold and reactivating:

- In Corda, a fund transfer with insufficient funds in the sender's balance will result in issuance of an Obligation
- In Hyperledger Fabric, an unsettled payment instruction is added as a new 'queued transaction' state in a bilateral channel and can be viewed by channel participants
- In Quorum, each bank maintains its own Private Queue which is a list of unsettled payment instructions. System uses a Global (public state) Gridlock Queue to track all queued payments

4.2.2 Hyperledger Fabric

A payment instruction is added to the outgoing queue when a fund transfer is performed and one of the following conditions apply:

- Outgoing queue has unsettled payment instruction with equal or higher priority
- No payment instructions in the outgoing queue but there are insufficient funds to settle the payment instruction
- Participants in the bilateral channel are currently participating in an ongoing gridlock resolution cycle

When a new payment instruction is added to the queue, a new 'queued transaction' state, which is a JSON object of all queued payment instructions, is created. Within a channel, participants are able to view the same states.

Both banks in a bilateral channel will have an identical view of the queued items, removing the need for maintaining two separate queues for incoming and outgoing payment instructions.

Queue settlement is a chaincode function of its own that is invoked/orchestrated via the App layer upon the completion of a pledge, cross-channel fund movement, additional incoming payment transaction, cancellation of a payment instruction in the outgoing queue and reprioritisation of a payment instruction in queue. Upon settlement of a payment instruction, the payment instruction is changed from a 'queued transaction' state to a 'completed transaction' state.

4.2.3 Quorum

Each bank maintains its own Private Queue which is a list of payment instructions that are not yet settled. Additionally, the system uses a Global (public state) Gridlock Queue to track all queued payments system-wide and trigger gridlock resolution. Both queues only hold the reference ID of the payment instruction, so that privacy of queues is maintained. Information about the payment instruction such as timestamp, amount, status, priority level and receiver is stored in the payment object itself in the Private Queue. Only the counterparties to the payment have access to this data.

When a fund transfer is performed, the system checks whether the bank has sufficient liquidity to make the transfer. Insufficient liquidity results in the payment instruction being added to both the Private and Global Gridlock queues.

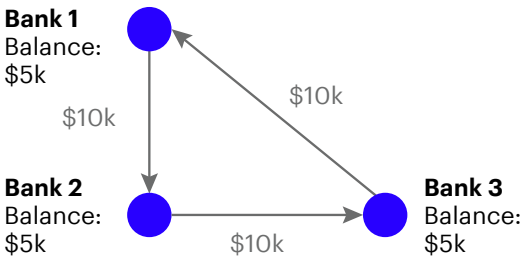
When the bank obtains new funds through either a fund transfer, pledge, or gridlock resolution, there will be an attempt to use the fresh funds to settle the payment instructions in the queue. The settlement order is based on priority level and FIFO as per the general queue mechanism logic established for Ubin Phase 2. Upon settlement, the payment instructions are removed from both Private and Global Gridlock queues.

4.3 GRIDLOCK RESOLUTION

As described in Section 2.1.2, a gridlock is when a group of senders and receivers with queued payment instructions are unable to settle unilaterally in a sequential manner due to insufficient funds, but the net liquidity across the participants in the gridlock is sufficient to settle the transactions simultaneously.

On the other hand, a deadlock arises when the gridlock results in a negative net liquidity across the participants and it is not possible to resolve unless additional liquidity is injected to the system. An example of a deadlock scenario is illustrated below.

Figure 14: Illustration of a deadlock scenario



Similar to fund transfer and unsettled payments in queue, privacy is an important consideration during gridlock resolution.

Table 1: Starting position of ten transactions in gridlock between five banks

5 BANKS

- A Bank A
- B Bank B
- C Bank C
- D Bank D
- E Bank E

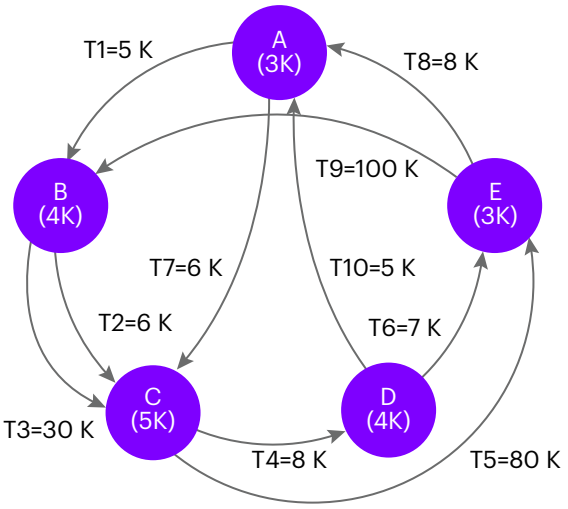
10 TRANSACTIONS

	A (K)	B (K)	C (K)	D (K)	E (K)
Starting balance	3	4	5	4	3
T1	-5	+5			
T2		-6	+6		
T3		-30	+30		
T4			-8	+8	
T5			-80		+80
T6				-7	+7
T7	-6		+6		
T8	+8				-8
T9		+100			-100
T10	+5			-5	

To illustrate the design for gridlock resolution across the 3 platforms, this section will refer to a common gridlock scenario, described below:

- There are 5 participating banks (Bank A, Bank B, Bank C, Bank D and Bank E) with starting balances of $3,000, $4,000, $5,000, $4,000 and $3,000 respectively
- There are a total of 10 payment instructions—T1, T2, T3, T4, T5, T6, T7, T8, T9 and T10. These are detailed in Table 1 where a negative value indicates amount to be paid while a positive value indicates amount to be received
- All payment instructions are of Normal priority
- The banks have insufficient liquidity to settle the first payment instruction in their outgoing queues

Figure 15: An illustration of a gridlock scenario between five banks, referenced in this report section



4.3.1 Corda

The three stages of Corda's gridlock resolution are: Detect, Plan and Execute. The gridlock resolution mechanism will run repeatedly in an attempt to settle the queued payment instructions in a gridlock. Instead of relying on conventional gridlock resolution algorithms such as EAF2, the Corda workstream developed a new cycle-based algorithm called Cycle-solver as described in this section.

Stage 1: Detect

1. Bank A initiates gridlock resolution.

2. The other 4 banks participate by contributing their highest priority and oldest active queued payment instructions for gridlock resolution.

   - When there are more than one queued payment instructions from one sender to the same receiver, the payment instruction to be selected will be based on the following criteria: Highest Priority, First-In First-Out (FIFO).
   - Bank B has two queued payment instructions to the same receiver Bank C i.e. T2 and T3. Both have the same priority but T2 has an older timestamp than T3. Based on FIFO, T2 is selected and T3 is excluded from this gridlock resolution cycle.

3. Bank A will initiate a flow to require all neighbouring nodes to propagate a scan request to discover available queued payment instructions in the network.

4. A recipient of the scan request must propagate the same request to all its neighbours to return a scan response message containing its queued payment instruction for gridlock resolution.

5. Upon receiving a scan request, the recipient must respond with a scan acknowledgement message and generate a random key which will anonymise its identity when responding with a scan request.

6. A recipient of a scan request will respond with a scan response based on either one of two criteria:

   - The receiver of its outgoing queued payment instruction is the sender of the scan request.
   - All of its recipients of the scan request it had sent has returned with the scan response.

7. Having collated all replies, the recipient of the scan request will send the scan responses back to the requester.

8. By responding to the request recursively and via propagation, each of the nodes in the propagation process can observe the transaction amount from each of the queued payment instruction in the scan responses. However, the identities of the sender and receiver participating in the queued payment instruction are anonymised, preserving the privacy of the parties involved. At the same time, the data structure used to facilitate the storage of this information is in volatile memory and will be purged upon gridlock resolution completion.

Table 2: Bank B has two queued payment instructions to the same receiver Bank C. Only transaction T2 is selected based on FIFO criteria

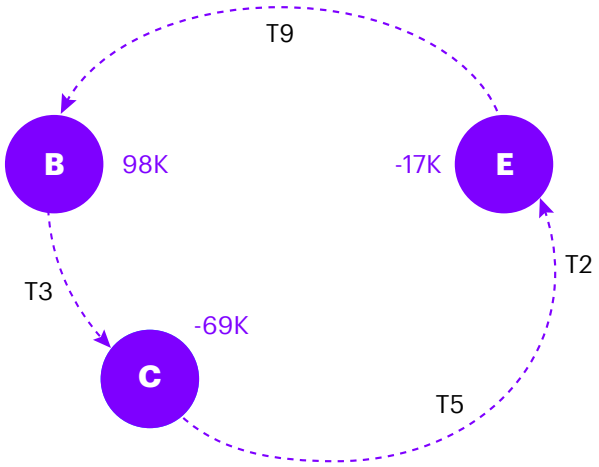
	A (K)	B (K)	C (K)	D (K)	E (K)
Starting balance	3	4	5	4	3
T1	-5	+5			
T2		-6	+6		
T3		-30	+30		
T4			-8	+8	
T5			-80		+80
T6				-7	+7
T7	-6		+6		
T8	+8				-8
T9		+100			-100
T10	+5			-5	

Stage 2: Plan

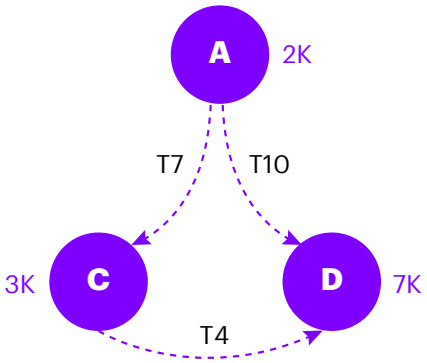
1. Bank A will compute cycles based on the scan responses with its respective sender-receiver of the queued payment instruction
2. Bank A constructs an in-memory graph representation based on the sender-reciever edge
3. Based on the cycles, Bank A will calculate the obligation sum (total value of obligations) in each of the cycles

In this scenario, eight cycles can be constructed. The diagram below illustrates how the obligation sum is calculated.

Figure 16: Illustration of how obligation sum is calculated



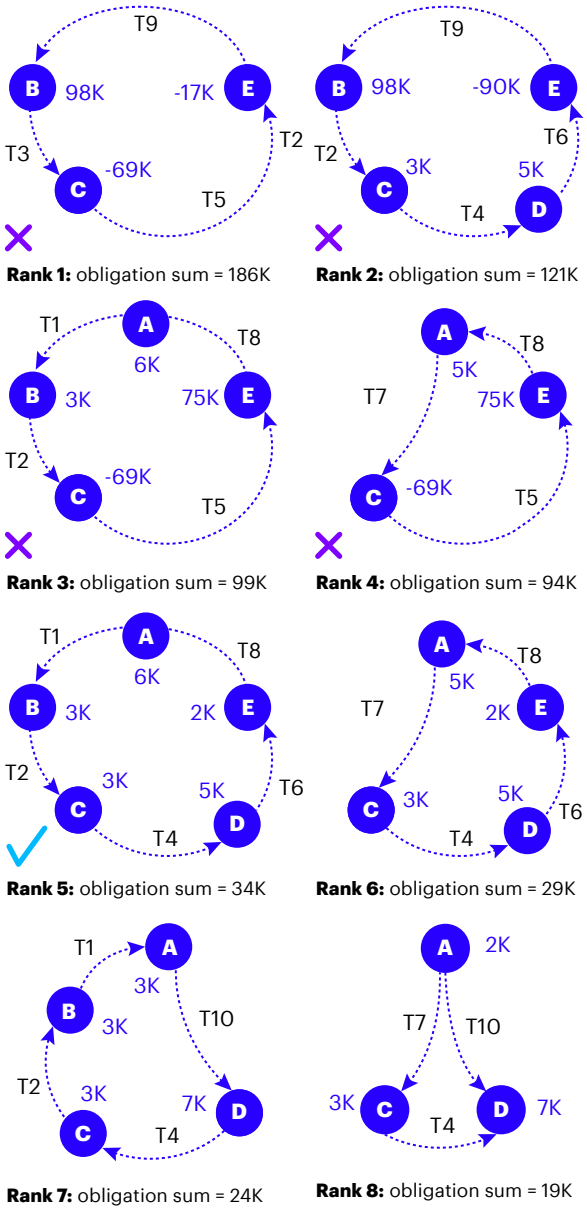
- Obligation sum = 6 + 80 + 100 = 186
- Netted balance for B, C and E are $98,000, -$69,000 and -$17,000 respectively



- Obligation sum = 6 + 8 + 5= 19
- Netted balance for A, C and D are $2,000, $3,000 and $7,000 respectively

4. The eight cycles are ranked based on obligation sum

Figure 17: Illustration of how cycles are ranked and eliminated



The cycles are ranked based on highest obligation sum, i.e. from the largest to the smallest obligation sum. The first four cycles are invalid as they produce deficits in netted balance. Cycle A-B-C-D-E will be chosen for the resolution as it is the next in rank (Rank 5) where netted balances are all positive. The remaining cycles will not be considered, though they do not produce deficit in the netted balance.

Stage 3: Execute

Resolution for Cycle A-B-C-D-E will be executed through a single atomic netting transaction.

Figure 18: Net settlement transactions to resolve the selected gridlock cycle

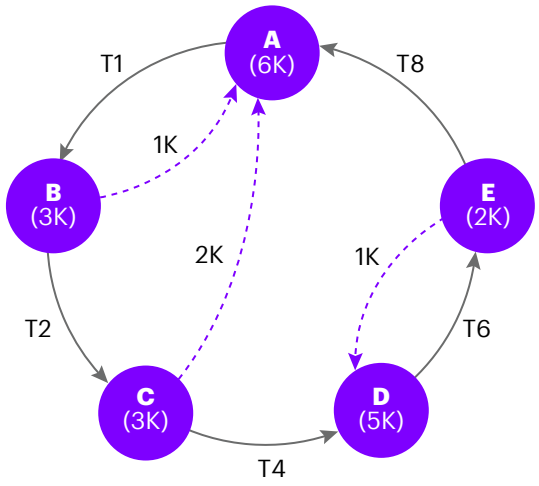


Table 3: Status of ten transactions after first round of gridlock resolution

	A (K)	B (K)	C (K)	D (K)	E (K)
Starting balance	3	4	5	4	3
T1	-5✓	+5✓			
T2		-6✓	+6✓		
T3		-30	+30		
T4			-8✓	+8✓	
T5			-80		+80
T6				-7✓	+7✓
T7	-6		+6		
T8	+8✓				-8✓
T9		+100			-100
T10	+5			-5	
Closing balance	3	3	3	5	2

After this round of gridlock resolution, the remaining obligations are T3, T5, T7, T9 and T10.

Table 4: Remaining gridlock transactions after first round of gridlock resolution

	A (K)	B (K)	C (K)	D (K)	E (K)
Starting balance	6	3	3	5	2
T3		-30	+30		
T5			-80		+80
T7	-6		+6		
T9		+100			-100
T10	+5			-5	

The three stages of Detect, Plan and Execute will be reiterated for the second gridlock resolution cycle with the remaining obligations. However, all the cycles are invalid as they produce deficits in netted balance. Nevertheless, unilateral payments can be performed to settle obligations T7 and T10. Bank A has enough liquidity to settle the $6,000 obligation with Bank C and Bank D can also settle the $5,000 obligation with Bank A.

Corda workstream designed and developed a new cycle-based algorithm called Cycle-solver. Corda's gridlock resolution are executed in three stages: Detect, Plan and Execute. The process starts when a participating bank initiates a flow to request all neighbouring nodes to propagate a scan request to discover available queued payment instructions in the network. The algorithm discovers possible netting cycles, and resolves the cycle with the largest sum of Obligations and which does not result in any participant going into deficit.

Table 5: The proposed list of gridlock transactions after Initiation/Participation stage

	A (K)	B (K)	C (K)	D (K)	E (K)	
Starting balance	3	4	5	4	3	
T1	-5	+5				Matched
T2		-6	+6			Matched
T3		~~-30~~	~~+30~~			
T4			-8	+8		Matched
T5			~~-80~~		~~+80~~	
T6				-7	+7	Matched
T7	-6		+6			Matched
T8	+8				-8	Matched
T9		~~+100~~			~~-100~~	
T10	+5			-5		Matched
Netted balance	5	3	9	0	2	

4.3.2 Hyperledger Fabric

For the Hyperledger Fabric prototype, the EAF2 algorithm is used for gridlock resolution. Each bank proposes their respective set of payment instructions to the netting channel and calculations for determining whether a solution is achieved or not will be done on the netting channel chaincode. In the netting channel, participants can only see the reference ID of the payment instructions proposed and the total net value of these payment instructions.

Gridlock resolution for the Hyperledger Fabric prototype is split into 2 main stages: Initiation/Participation and Settlement.

Stage 1: Initiation/Participation

1. To initiate a new gridlock resolution cycle or participate in a current cycle, a bank retrieves all active payment instructions (both incoming and outgoing) across all of its bilateral channels.
2. These payment instructions will be sorted according to their priorities and time of creation.
3. The payment instructions will be separated into two lists, nettable and non-nettable, based on the criteria that all nettable payment instructions should not cause the netted balance of the bank to be in deficit after gridlock resolution.
    - If the bank is participating in an ongoing gridlock resolution cycle, the current non-nettable list in the cycle should also be taken into consideration (i.e. non-nettable payment instructions already logged in the current gridlock resolution cycle by one bank should also be marked as such when another bank proposes its non-nettable list)
4. The proposed list of all nettable and non-nettable queued payment instructions will then be submitted into the netting channel along with the net value of the nettable payment instruction list

5. The gridlock resolution cycle will be marked as 'Achieved' if the new proposal by the bank results in the following:

   - There is a matching pair for all nettable payment instructions proposed for the cycle
   - The total net value of all payment instructions equals to 0

6. Banks will continually participate until a resolution is found. If there is no solution found within the stipulated time, the gridlock resolution cycle will be marked as 'Expired'

Table 5 on the previous page illustrates a typical gridlock cycle after Initiation/ Participation stage.

1. Assuming Bank A initiates the gridlock resolution cycle, Bank A will propose all of its payment instructions T1, T7, T8 and T10 as nettable, since these payment instructions have a net value of +2. As Bank A has a current balance of 3, this would result in a positive netted balance of 5

2. Similarly, Bank B will also propose T1, T2, T3 and T9 as nettable as they add up to a net value of +69, resulting in a netted balance of 73

3. Bank C, however, will only propose T2, T3, T4 and T7 as nettable and T5 as non-nettable, as inclusion of T5 will cause Bank C to have a netted balance of -41 (deficit)

4. In the same vein, Bank D will propose T4, T6 and T10 as nettable

5. When it comes to Bank E, only T6 and T8 will be proposed as nettable while T5 and T9 will be proposed as non-nettable since T5 is already marked as non-nettable by Bank C and the inclusion of T9 will cause a deficit

6. During the second round of participation and proposal by Bank B, only T1 and T2 will be proposed as nettable while T3 and T9 are marked as non-nettable as T9 is already marked by Bank E as non-nettable and with that as a consideration, T3 will now cause the netted balance for Bank B to be in deficit

7. Similarly, when Bank C re-proposes, since T3 is already marked as non-nettable by Bank B, T3 will be removed from the nettable list, leaving only T2, T4 and T7 in the nettable list

8. With the second proposal by Bank C, there is now a matching pair of for every proposed nettable payment instruction in the netting channel. The netting channel chaincode will now mark the cycle as 'Achieved' and it's now ready for settlement

Stage 2: Settlement

At this stage, the netting chaincode will calculate and settle netted payment instructions. The closing balance of each bank is $5,000, $3,000, $9,000, $0 and $2,000 respectively after gridlock resolution. T3, T5 and T9 remain unsettled/ remain in the queue.

Figure 19: The remaining gridlock transactions

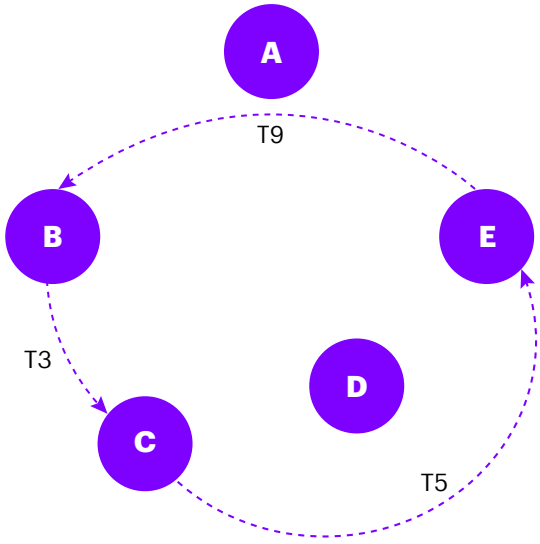


Table 6: Remaining payment instructions after gridlock resolution

	A (K)	B (K)	C (K)	D (K)	E (K)
Closing balance	5	3	9	0	2
T3		-30	+30		
T5			-80		+80
T9		+100			-100

As a facilitator of gridlock resolution settlement, once it is detected that a gridlock resolution cycle is achieved, MAS node will calculate the resulting net balance of each participating bank to ensure that no bank ends up in deficit at the end of settlement.

After the check, MAS will then deduct and add funds accordingly to each bank's balances. Thereafter, MAS will mark all the appropriate payment instructions in their respective bilateral channels as settled.

An alternative decentralised settlement approach could be designed which will involve additional orchestration between channels.

For the Hyperledger Fabric prototype, the EAF2 algorithm is used for gridlock resolution. Gridlock resolution for the Hyperledger Fabric prototype is split into 2 main stages: Initiation/Participation and Settlement. The resolution starts when their respective set of payment instructions to the netting channel and computation using EAF2 algorithm for gridlock resolution is done on the netting channel chaincode.

4.3.3 Quorum

For the Quorum prototype, the EAF2 algorithm is used for gridlock resolution whilst maintaining balance privacy and meeting the goal of decentralised processing. Quorum's implementation of decentralised gridlock resolution is driven by a cycle of 4 systemic states; Normal, Line Up, Resolving, and Settling. The states are maintained in a smart contract that is synchronised to all nodes. Node behaviour is determined by the current system state. The state dictates how the nodes should handle payment instructions and function during the gridlock resolution process. An event is emitted by the Quorum node each time the state changes. The DApp listens for state change events and orchestrates smart contract execution.

State 1 – Normal

During the Normal state, which is the default state, banks may transfer funds to each other. Fund transfers are stored as payment instructions in the system, and are either settled immediately (if the sending bank has sufficient liquidity) or queued. The Quorum node remains in Normal state until the number of payment instructions in the Global Gridlock Queue reaches a predefined threshold. This threshold, which is configurable, serves as an automatic trigger to move the node into the next state - Line Up.

State 2 – Line Up

The Line Up state kicks off the gridlock resolution process. Banks that have outgoing or incoming queued payment instructions line up by executing a smart contract function. The order in which they execute this function results in the sequence for the gridlock resolution algorithm to execute.

Whilst this order is dictated by when each bank executes this smart contract function, and is therefore arbitrary, i.e. determined by hardware and network latency, it can also be configured to be pre-determined. The next state Resolving begins when all banks have lined up or when a timeout is reached.

State 3 – Resolving

In this state, Resolving, each bank initiates gridlock resolution according to the EAF2 algorithm. In the resolve state, each bank evaluates its queued active payment instructions by inactivating its latest payments instructions until it results in a positive balance position. If a payment instruction is inactivated, the receiver's position will be adversely affected, and it will need to evaluate its netting set again.

- Bank A will evaluate payment instructions T1, T7, T8 and T10. Since the net of all these 4 transaction results in positive balance for Bank A, Bank A responds to the network that it can resolve all 4 payment instructions.
- Bank B, which is next in resolve sequence, evaluates payment instructions T1, T2, T3 and T9 and responds to the network that it can resolve all payment instructions
- However, when Bank C evaluates payment instructions T2, T3, T4, T5 and T7, it will notice that the net balance is a deficit of $41,000. As such Bank C will inactivate the latest payment instruction T5 and will respond to network that it can only resolve T2, T3, T4 and T7.

Table 7: Starting position of 10 transactions in gridlock between five banks

First bank → Last bank

Oldest transaction ↓ Latest transaction

	A (K)	B (K)	C (K)	D (K)	E (K)
Starting balance	3	4	5	4	3
T1	-5	+5			
T2		-6	+6		
T3		-30	+30		
T4			-8	+8	
T5			-80		+80
T6				-7	+7
T7	-6		+6		
T8	+8				-8
T9		+100			-100
T10	+5			-5	

Table 8: Inactivating T5 as it caused Bank C and Bank E to have a negative netted balance

First bank → Last bank

Oldest transaction

	A (K)	B (K)	C (K) Deficit	D (K)	E (K) Deficit
Starting balance	3	4	5	4	3
T1	-5	+5			
T2		-6	+6		
T3		-30	+30		
T4			-8	+8	
T5			1 -80		+80
T6				-7	+7
T7	-6		+6		
T8	+8				-8
T9		+100			-100
T10	+5			-5	
Netted balance	5	73	-41	0	-18

Latest transaction

Table 9: Inactivating T9 as it caused Bank B and Bank E to have a negative netted balance

First bank → Last bank

Oldest transaction

	A (K)	B (K)	C (K)	D (K)	E (K) Deficit
Starting balance	3	4	5	4	3
T1	-5	+5			
T2		-6	+6		
T3		-30	+30		
T4			-8	+8	
T5			1 -80		+80
T6				-7	+7
T7	-6		+6		
T8	+8				-8
T9		2 +100			-100
T10	+5			-5	
Netted balance	5	-73	39	0	-98

Latest transaction

Table 10: Inactivating T3 as it caused Bank B and Bank C to have a negative netted balance

First bank → Last bank

Oldest transaction ↓ Latest transaction

Deficit

	A (K)	B (K)	C (K)	D (K)	E (K)
Starting balance	3	4	5	4	3
T1	-5	+5			
T2		-6	+6		
T3		3 -30	+30		
T4			-8	+8	
T5			1 -80		+80
T6				-7	+7
T7	-6		+6		
T8	+8				-8
T9		2 +100			-100
T10	+5			-5	
Netted balance	5	-27	-39	0	2

Table 11: Final netted balance for each bank after inactivating transaction T3, T5 and T9

First bank → Last bank

Oldest transaction ↓ Latest transaction

	A (K)	B (K)	C (K)	D (K)	E (K)
Starting balance	3	4	5	4	3
T1	-5	+5			
T2		-6	+6		
T3		3 -30	+30		
T4			-8	+8	
T5			1 -80		+80
T6				-7	+7
T7	-6		+6		
T8	+8				-8
T9		2 +100			-100
T10	+5			-5	
Netted balance	5	3	9	0	2

- Bank D, next in resolve cycle, responds to the network that it can resolve all payment instructions T4, T6 and T10
- Bank E will see that T5 has already been inactivated. It evaluates only the remaining payment instructions T6, T8 and T9. Bank E notices that the net balance is deficit of $98,000. Bank E in this case will inactivate the latest payment instruction T9 to bring it to a positive net balance of $2,000.
- The inactivation of T9 will result in Bank B having deficit balance position, hence the resolve cycle will move back to Bank B. Bank B will inactivate the next latest payment instruction T3 as illustrated in table 10
- Since the inactivation of T3 impacts the balance position of Bank C, the resolve cycle will move to Bank C. However Bank C will notice that the inactivation of T3 does not take adversely impact its balance position. The netted balance position for each of the bank after Resolve Cycle is as shown in table 11.
- Since the entire network has an agreed set of resolvable payment instructions, the resolve cycle completes and this initiates the next state.

State 4 – Settling

In this phase, banks with remaining active payment instructions generate zero knowledge proofs for their respective incoming and outgoing payment instructions and submit the same for validation by the entire network. These proofs form a chain, which is to be validated atomically. Once validation is successful, the shielded salted balances of the participating banks are updated, and the shielded payments that are netted during this round will be marked as processed. This completes the gridlock resolution phase and the system returns to Normal state.

For Quorum prototype, the EAF2 algorithm is used for gridlock resolution. The Quorum workstream's gridlock resolution is driven by a cycle of 4 systemic states: Normal, Line Up, Resolving, and Settling. The process starts with banks having queued payment instructions to line up by executing a smart contract function. Gridlock resolution is initiated according to the EAF2 algorithm. Once resolution is found, banks will generating ZKP for their transactions and submit for validation. Upon validation, the shielded balances are updated in the public contract and all the netted transactions are marked as processed.

KEY OBSERVATIONS AND FINDINGS 05

5.1 CORDA

Privacy

The Corda network distributes the ledger based on a need-to-know basis instead of a global broadcast method, so only parties involved in a particular transaction have visibility of transaction details. This model inherently addresses privacy concerns.

The Corda prototype strengthens privacy in its design with an additional layer of Confidential Identities added to each transaction, whereby only parties involved in a transaction can identify the participants. These participants exchange the fresh key and certificate using Corda's 'Swap Identities Flow' and the new keys are used for the output, command and signature of the transaction. While both the sender and receiver details are anonymised, the transaction amount is not. The transaction amount is required to enable the 'planning' phase of gridlock resolution algorithm to compute the best gridlock resolution opportunities based on the amount of the queued payment instructions.

In the current prototype of a small network, exposing transaction amount in the network may lead to privacy concerns where it may be possible for a network member to attempt to graph the network and deduce the sender and receiver of each of the queued payment instructions. However, such a mapping would be more complex for larger payment networks than MAS MEPS+ that already has 63 participating banks and average of 6,000 transactions in a single gridlock resolution cycle.

Scalability and performance

In the design of the Corda workstream, adding a new participating node involves only the installation of the new node itself. Minimum change is required to existing nodes or existing network setup in the process of adding new nodes.

In Corda, transactions are only sent on a need-to-know basis and each peer only sees a subset of facts on the entire ledger. This alleviates the scalability and performance issue commonly faced by traditional DLT platforms, which store and update the entire ledger on every peer. At the same time, in industries where high transaction throughput is appreciated, multiple notary services allow load balancing and an increase in transaction throughput.

The distinct UTXO model in Corda requires input states to link one or more inbound transactions by their hash, resulting in an immutable chain of asset lineage. This lineage chain could be 'long and heavy' especially after a long duration and many cycles of transactions. In every transaction, each of the Corda nodes will 'walk-the-chain' to verify each input was generated in a sequence of valid transactions, validating the authenticity of the chain. This can be

easily overcome with implementing an expiry to the digital assets. In other words the pledged funds will need to be recycled after a period of time.

Additional exceptional scenarios were considered during the project:

EXCEPTIONAL SCENARIO	OBSERVATIONS
Impact of injecting transaction(s) to the network during gridlock resolution	New payment instructions can be processed while gridlock resolution is running. In addition, any obligation that is used during a gridlock resolution can be cancelled, or reprioritised without disruption. In the case where the incoming transaction uses the same states involved in gridlock resolution process, the first transaction to be notarised would succeed and the second would fail, because Notary would reject the transaction due to double spending.

Resiliency

If any of the participating bank nodes are unreachable, for example machine failure, network issues, etc., the network can still operate for all transactions that do not require the involvement of the failed nodes. At the same time, nodes can be shut down and restarted at will due to Corda's 'Flow Checkpointing mechanism', ensuring data is never lost and flow progression is protected. In addition, fund transfer transaction can still proceed among participating nodes whom are not involved in the gridlock resolution cycle.

For this prototype, a single Simple Notary service was used, introducing a potential single point of failure. If the single notary failed, transactions cannot be completed. As an observation, further enhancement can be done to implement the notary service as a cluster potentially being operated by multiple parties. Such a solution can be designed to support load balancing to increase transaction throughput, multi-threading of incoming transactions and minimising latency for geographically diverse transacting parties.

EXCEPTIONAL SCENARIO	OBSERVATIONS
Impact of removing 1 participating bank during gridlock resolution	The current prototype does assume that the production version ensures High Availability (HA), therefore it does not support removal of any participant node during gridlock resolution cycle.
Impact of removing MAS during gridlock resolution	There is no impact to the network (other participating nodes) as MAS is not required to orchestrate the gridlock resolution as well as any transactions that MAS is not a participant of.

Finality

In Corda, the notary service provides the point of finality in the transaction where the presence of a notary signature indicates transaction finality. By obtaining a notary signature, participants of the transaction can be equally sure that the input states are unconsumed (unspent) by prior transactions.

Due to the notary model, a transaction's proposed changes are either all accepted or none are. Any queued payment instruction that is involved in gridlock resolution can be modified freely (reprioritised or cancelled), and new fund transfers can be initiated and settled in real time. The netting solution can guarantee atomicity when it fails either because the graph is no longer valid due to modified payment instruction or a decrease in balance.

5.2 HYPERLEDGER FABRIC

Privacy

Hyperledger Fabric is a permissioned network with the ability to set up private channels between participants where each channel maintains an independent ledger. Channel enables information to be shared between parties on a need-to-know basis. A channel is a data partitioning mechanism to limit transaction visibility only to stakeholders. Other members on the network are not allowed to access the channel and will not see transactions on the channel. Ledgers exist in the scope of a channel. This enables the setup of ledgers which can be shared across an entire network of peers (e.g. netting channel) and ledgers which include only a specific set of participants (e.g. bilateral channels). By design and to emphasise privacy, given that fund transfer and queuing mechanisms occur within bilateral channels, the transaction details are only visible to the pair of banks in the bilateral channel and MAS (as a regulator). Both banks in this design can view the balances of both party's channel-level accounts. However, given that a bank can only view the balance of one channel-level account per counterparty, it is not possible for any bank to deduce the total balance or liquidity of a counterparty within the DLT.

The funding channel is intended to facilitate legitimate movement of funds across channels. In the current prototype, this channel is only used to ensure that funds being moved are tracked in the funding channel to allow for traceability. It is possible for network participants to identify the channels involved in each fund movement transaction. For future considerations, it is advised that the data is secured at the infrastructure level, such as deploying additional cryptographic functions to ensure confidentiality. With this, it is also possible to remove the funding channel altogether to allow higher efficiency.

In the netting channel, the identities of the banks participating in a gridlock resolution cycle, payment instruction IDs and net value of nettable payment instructions are included per bank. There is no individual transaction amount exposed. In a gridlock resolution cycle with few payment instructions, it is possible to deduce the amount for the payment instructions. However, given that netting is likely to involve multiple payment instructions per cycle, the amount for each particular payment instruction cannot be deduced.

Scalability and performance

In this design, it is required to set up [N x (N-1) / 2] + M channels to achieve the intended objectives, where:

N = number of participating nodes

M = number of multilateral channels, i.e. 2 for Ubin Phase 2 with a funding channel and a netting channel

e.g. for 10 banks, the design requires 47 channels i.e. [(10 x (10-1) /2] + 2

The number of channels will increase with every new participant, which in turn increases the complexity in terms of network and channel management.

Bilateral channels allow bilateral netting to happen within a channel, without involvement of the rest of the network participants. This introduces the possibility that queued payment instructions in a bilateral gridlock can be resolved within the bilateral channel without having to depend fully on gridlock resolution with multiple parties in the network.

The design of an individual bilateral channel between pairs of transacting banks also introduces the need for bank operators to maintain the funds in each channel. This means that in addition to the total pledged fund from MAS to the participating bank in the DLT, these banks need to define and move specific amounts of funds between its bilateral channels where needed. The movement of funds across bilateral channels is driven by the bank users based on business operations demands. There is a possibility to automate fund movement to conform to the business operation rules. A possible solution for this is to design a fund movement algorithm that can be automated in the system.

An orderer broadcasts the transactions to all peers in a channel for validation before committing the transaction. For the purpose of prototyping, Ubin Phase 2's Hyperledger Fabric setup consists of one orderer which sends transactions to all peers in a channel for validation. A multiple node ordering service (e.g. Kafka) can be implemented for high availability of the ordering service to ensure it does not become a single point of failure.

The overhead caused by the setup of multiple channels in this design can be reduced by means of sharing hashed data among all participants while keeping private data with a limited set. This new component is underway and expected in future releases of Hyperledger Fabric.

Note: Kafka is an open-source stream processing platform that allows high-throughput and low-latency processing for real-time data feeds.

Additional exceptional scenarios were considered during the project:

EXCEPTIONAL SCENARIO	OBSERVATIONS
Impact of injecting transaction(s) to the network during gridlock resolution	Any functionality which reduces the liquidity or alters the queue positions of a gridlock resolution participant will be unavailable. New payment instructions will be queued until the current gridlock cycle has completed or timed out.

Resiliency

The design of Ubin Phase 2's Hyperledger Fabric prototype consists of one orderer to send transactions to all peers in a channel for validation. This presents a single point of failure to the network; when the orderer fails, no transaction can be ordered into blocks and committed to the chain. Fortunately, this issue can be resolved with the implementation of multiple node ordering service (e.g. Kafka) for high availability.

One key trade-off in the Hyperledger Fabric workstream design is the number of channels required to guarantee high levels of privacy, while achieving gridlock resolution. Hence cross-channel communication (e.g. movement of funds from one bilateral channel to another) becomes a vital part of the design. In the developed prototype, the orchestration logic to manage communication between two channels are programmed in a custom application using Node.js. At this point, cross-chain interaction is still not supported and is being planned for future platform release. Another alternative is to ensure there is a rollback mechanism for an application-orchestrated function that involves multiple chaincode executions.

In terms of node setup, higher resiliency be achieved by setting up reasonable and cost-effective redundant nodes in the system. This applies for bank nodes and MAS node.

EXCEPTIONAL SCENARIO	OBSERVATIONS
Impact of removing 1 participating bank node during gridlock resolution	Only gridlock resolution will be affected as it requires all nodes to endorse any new netting proposal. If a node fails mid-cycle, no new proposals can be added until the failed node recovers or the cycle times out. In order to address such resiliency risks, the endorsement policy could be revised to require a set minimum number of participating nodes.
Impact of removing MAS during gridlock resolution	In this design, settlement for gridlock resolution is dependent on MAS to update balances and settle queues across all bilateral channels. An alternative decentralised settlement approach could be designed which will involve additional orchestration between channels.

Finality

For each transaction, the peers defined in the endorsement policy of the chaincode will endorse the transaction. The endorsed transaction is then sent to the orderer for it to order it into a block and broadcast it to the channel participants to validate and commit the block to their ledgers. This mechanism assures the finality of transactions within a channel.

However, transactions involving communication between channels remain subjected to orchestration by conventional technologies, as such orchestration logic is developed using a custom application on Node.js instead of within the chaincode.

Another key observation is that endorsement for cross-channel fund movement only involves the sending and receiving banks who can collude to create more funds in the DLT. It is possible for MAS to trace such an occurrence but this would increase reliance on MAS as a governing entity.

5.3 QUORUM

Privacy

Quorum supports both public and private transactions within the permissioned network. The public transactions are broadcast to all the nodes within the network and are processed like regular Ethereum transactions. The private transactions are sent directly to the specified recipients by Quorum's privacy service Constellation as encrypted blobs. It does this by sending the transaction payload only to the involved participants and the rest of the network can only see a hash of the encrypted payload. The key benefit of propagating these hashes to all participants is one of security and resiliency: should a party to a private transaction require validation of the existence of that transaction at some point in the future, they can confirm this with the rest of the network by comparing it to the hashes that the network holds, thereby not needing to trust the information held at the counterparty. Private transactions allow banks to execute payment instructions to the specified receiving bank only.

In addition to the above, the current design also incorporates zero knowledge proofs (ZKP) for managing the shielded salted balance of each participating bank. The true balance position of any participant bank is only visible to itself. Any change in balance movement as part of transaction execution can only happen via submission of ZKP by the sender and receiver, followed by verification of these proofs by the entire network. This allows balance validation by the network without knowledge of the true balance and thus avoids double spending in a truly decentralised way.

The prototype also incorporates the following components for privacy:

- Payment transfers on the public contract identify the sender and receiver but shield transfer amounts by storing hash values
- Account balances are kept in private contracts and are only accessible from the account owner's node
- Dynamic salt is used when hashing transaction amounts, starting balance and ending balance, uniquely for each transaction

Scalability and performance

Quorum Network Manager (QNM) – an open source tool for creating and managing Quorum networks – was used in Ubin Phase 2 to setup a Raft based Quorum network. The tool automates basic network setup tasks and configuration. The current version of Quorum supports dynamic addition of new nodes, however this could not be tested as a lower version of Quorum was used.

It was observed that the current ZKP generation process takes approximately 4 seconds to generate with a total transaction processing time of 5 seconds for a fund transfer. There is currently research and development work underway to improve the performance of ZKP algorithms. These include plans to increase proof generation and validation speed and lower the memory requirements.

Additional exceptional scenarios were considered during the project:

EXCEPTIONAL SCENARIO	FINDINGS
Impact of injecting transaction(s) to the network during the gridlock resolution cycle	After a gridlock resolution cycle has begun and is ongoing, new payment instructions submitted will be inserted into the Gridlock queue as 'Inactive' and will not be processed till the current resolution cycle is complete.

Resiliency

Quorum inherits Ethereum's block propagation mechanism. If any Quorum node goes down and is disconnected from the network for any reason, the rest of the network can still function as normal. However in such a scenario, no transactions with the unavailable node will be allowed. The intrinsic resiliency of Ethereum ensures that the transaction history is automatically synchronised when the disconnected node comes back online.

This demonstrates that high availability is built into the core platform itself. There are other observations in the current Quorum prototype that can be further enhanced. For example, when the Raft leader is chosen at the time of network set up, the Raft leader could be randomly elected for each transaction for a more resilient design. Future Quorum release is expected to include Byzantine fault tolerant consensus mechanism that rotates the leader before every block creation. Also given that the DApp orchestrates payment flow, manages the proof generation and submission and acts as a link between the public and private contracts, it is another area which requires design focus to ensure overall network resiliency.

EXCEPTIONAL SCENARIO	FINDINGS
Impact of removing 1 participating bank node during gridlock resolution	Gridlock resolution in Quorum is designed to be less dependent on the participants. In a case where one of the nodes goes down during gridlock resolution process, remaining available nodes are able to proceed and complete ongoing gridlock resolution.
Impact of removing MAS during gridlock resolution	Gridlock resolution is not dependent on the participation of the MAS node. The other participants can complete gridlock resolution successfully.

Finality

This prototype leverages the Raft consensus model where the elected Raft leader commits new blocks to the chain after verifying the block's transactions and all followers update to the latest block in lock-step. Once a block is committed to the chain it cannot be reversed, thereby providing transaction finality. Currently, the Raft leader is elected during network creation. However, when the nodes in the network detect that the Raft leader is down, the network will elect a new Raft leader, allowing new transactions to be processed.

During the execution phase of the gridlock resolution cycle, the netting process is orchestrated by the DApp where each settlement transaction is processed individually. Atomicity of netting is ensured where the shielded balances are only updated once all the relevant proofs are validated. If one proof fails the entire netting round fails.

The execution time taken for ZKP generation may pose a concern on transaction finality, as participating nodes may drop off during the gridlock resolution cycle (Settling state). As per the current design, if a node drops off during Settling state, the related proofs for the netting will not be received by the network and as such the entire settlement will be invalidated. However, with the expected improvement in ZKP algorithm in the near future, this may no longer be a concern.

5.4 MICROSOFT AZURE

In Ubin Phase 2, the three prototypes were hosted in Microsoft Azure across a total of 41 virtual machines. The Azure cloud solution provided quick turnaround time for resource provisioning and marketplace templates for DLT environment and network setup. This quick provisioning and reliability of the Azure virtual machines complemented agile delivery methods to enable Ubin Phase 2 development to proceed without infrastructure constraints.

FUTURE CONSIDERATIONS 06

Ubin Phase 2 has successfully demonstrated that with three different designs on three different platforms, a traditionally centralised RTGS process could be executed in a decentralised manner without compromising privacy.

Apart from the key findings and observations, there were additional discussions and learnings brought up during the demonstration sessions led by Accenture with the consortium of bank representatives. Although these discussions were not intended topics for Ubin Phase 2, they are worth addressing in view of operationalising a fully-functional DLT-based RTGS system. Each of these considerations are likely to require an in-depth assessment and design which could impact the current operating model, policies and procedures as well as the technicalities of the system. Although not an exhaustive list, the discussions are categorised into the following six topics:-

6.1 RESILIENCY AND CLOUD READINESS

A key benefit of DLT is the distribution of processing and data storage across all nodes in the network, mitigating the risk of single point of failure. In theory, it also allows the system to recover quickly and continue operating in the event of system failure or other disruptions, as data and processing are distributed across nodes. Although Section 5 showed that resiliency across the network can be realised with the various design considerations, more work should be done to understand the resiliency of a distributed system, particularly, the recovery point and high-availability backup. In addition, in a conventional centralised system, Disaster Recovery and High Availability strategies focus on a single organisation. In a decentralised system, the approach should consider the entire network and cross-organisation.

Similarly, all three prototypes of Ubin Phase 2 were successfully developed and deployed to the Microsoft cloud infrastructure. This underscores the feasibility of operating DLT on a cloud infrastructure.

In order to operationalise a DLT-based RTGS system on cloud, the current prototype has to be enhanced to integrate various cloud infrastructure functionalities such as environment administration, monitoring

and recovery. Furthermore, interoperability of a DLT network on different cloud solutions can be further explored to cater to participants who have different preferred choice of cloud provider.

6.2 24X7 OPERATIONS

Deploying a DLT-based RTGS system opens up the possibility of operating 24x7, providing new opportunities for Singapore to be a global financial centre. This is especially beneficial for cross-border transactions across countries with no overlapping office hours, e.g. Singapore and Canada. The DLT-based RTGS system also allows interbank fund transfer to be completed without requiring all participants to be active. However, there are several operational considerations that need to be addressed before a truly 24x7 decentralised interbank payment system can be deployed. These include:

- Transacting value-date handling for transactions executed after operating hours and on public holidays
- Transaction/Handling fees and incentives during and after business hours
- Differing cut-off time of different banks
- Varying SLA requirements for bank operations and customer services
- Determination of foreign currency exchange rates, particularly for transactions involving non-SGD after trading hours, or dependent on parties outside the DLT network (e.g. securities or foreign markets)

6.3 LIQUIDITY SAVING MECHANISM (LSM)

With the current design, the RTGS system triggers the Liquidity Saving Mechanism (LSM) algorithms centrally through a predefined interval. Ubin Phase 2 prototypes demonstrate that there are different methods to initiate gridlock resolution in a decentralised system (and also detect and avoid simultaneous gridlock resolution). In other words, there is flexibility to select how and which node initiates gridlock resolution, be it scheduled, user-triggered or based on a predefined state or event.

However, the mechanism of initiating gridlock resolution may lead to unintended consequences and inequality among participants in the network. For example, a participating bank may instigate that gridlock resolution be triggered to its preference while other participants may not be in the right liquidity position at that time. Therefore, to ensure a degree of fairness to all participating banks, a more detailed analysis of the transaction patterns across the entire processing day is required in order to arrive at an optimal design for the mechanism to initiate gridlock resolution. Such mechanisms should also take operational factors into account, such as operating hours and liquidity threshold in the system.

In addition, LSM processing may take a longer time in a large network which may result in delay to the settlement. Hence, the desired design is to decouple LSM from fund transfer. One consideration is to improve this prototype to segregate bank balances by reserving a portion of liquidity for fund transfer during LSM processing.

6.4 DEGREE OF DECENTRALISATION

Another observation that came out of Ubin Phase 2 is that while decentralisation is technically feasible, there are multiple facets to consider to achieve a fully decentralised model where every node is equivalent. This is because, in practice, not all banks (nodes) are equal. Participating banks may differ in the transaction volume and the incentive to participate in an equally distributed network may vary. This may lead "smaller" participants to be unable and/or unwilling to bear the cost of ownership for infrastructure equally. Drawing inspirations from other existing RTGS systems, it is observed that there might be a potential to implement a hierarchical system whereby participation of the network can be split between direct and indirect participation. Such a model may work well especially when there is a potential of including participants outside of the financial services industry. For such a 'semi' decentralised network to be feasible, strong governance and policies are still needed to govern the relationship and service level agreement between participants, such that the model does not create an economic advantage to larger banks.

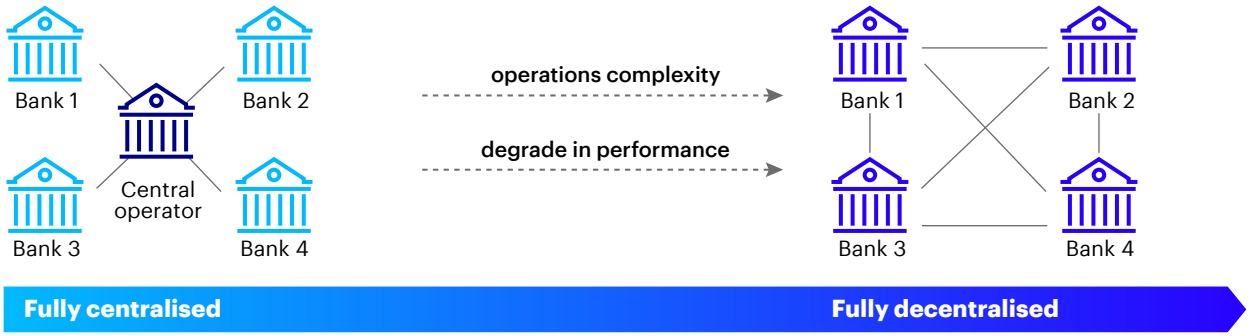
In a broader context, extending beyond Ubin Phase 2 which focuses on domestic interbank payment, multiple ledgers across different DLT platforms maybe required to facilitate an end-to-end Delivery vs Payment (DvP) process. Such a process will still need to ensure atomicity across the transaction lifecycle. Future development is needed to explore the feasibility of cross-chain feasibility which may leverage on the work already done on the Interledger Protocol (ILP). The business and operations considerations should also be evaluated to enable a fully decentralised DvP process.

6.5 MANAGING MULTI NODES

In a decentralised RTGS system, each participant is required to manage and operate its own node. This includes maintenance and support of both hardware and compatible software. For a DLT-based RTGS system to work effectively, it is important to ensure that all banks (nodes) across the entire network are consistent. This includes both functional and non-functional aspects such as:

- **Functional:** Consistency of algorithm and system parameters to ensure the operability of all nodes in a decentralised network. It is necessary to ensure that all participating banks (nodes) are effectively able to execute a distributed queue prioritisation, triggering of gridlock resolution and handle exceptions for

Figure 20: Illustration of transitioning from a centralised RTGS system to a fully decentralised RTGS system



each node such as time-out and retry. Also, when the network is opened up to other banks from different jurisdictions with different liquidity and infrastructure, operational level agreements and service level agreements are needed to manage the serviceability of the node.

- **Non-functional** considerations include operating systems and software patches, as well as infrastructure configurations as this has a bearing on security and vulnerability of the entire DLT network.

Another key aspect of managing multiple nodes is to consider the process and governance to add and remove participant nodes. A central system operator may no longer be required in a decentralised RTGS system, but a central governance body is still required to govern the consistency of the entire network. The governance parameters and process should be defined in detail to ensure the consistency and governance of a multi-node decentralised RTGS system.

Moreover, network latency can be a challenge in a very large multi nodes network i.e. with more than a thousand nodes. In the case of RTGS, it may impact the efficiency of the LSM algorithms. One option is to decentralise the LSM execution by distributing the LSM processing to smaller clusters of nodes, perhaps in a pyramid or tiered system where the left-overs are aggregated and netted at the next level. Distributed netting may alleviate efficiency and scalability concerns.

6.6 ROLE OF CENTRAL OPERATOR

With the potential of operating a DLT-based RTGS system, the conventional role of a central bank or payment system operator as the centralised infrastructure operator in the ecosystem will be obsolete. This would also mean that a central financial market infrastructure operator would not be necessary, as the processes and data are distributed across the participants in the DLT network. A DLT-based RTGS system reduces the costs and resources for the day-to-day operations and eliminates the risk of the central bank being the single-point-of-failure of the entire financial ecosystem.

However, Ubin Phase 2's findings suggest that the mandate to oversee the safety and efficiency of the payment and settlement system remains unchanged in a new decentralised model. Some of the roles that are observed include:

- **Overall liquidity manager:** Monitoring overall network liquidity (i.e. identify gaps and volatility) as well as intervening where necessary
- **System governance:** Ensuring compliance, consistency of the system operations such as software patches and parameters, and hardware configurations, as well as creating rules and guidelines for all players to operate and further monitoring, adding or removing participant nodes
- **Service Level Agreement (SLA) governor:** Defining the SLA in the decentralised system and, where possible, eliminating the need of multiple bilateral SLAs between banks
- **System auditor and mediator:** Resolving disputes

The above list is non-exhaustive and serves as a starting point for proper definitions and operations design to reconsider the role of a central operator in a decentralised RTGS model.

CONCLUSION

07

With the collaborative effort of MAS, ABS, 11 financial institutions, four technology providers, and Accenture, Ubin Phase 2 successfully concluded with its intended goals achieved.

The findings from Ubin Phase 2 demonstrate that all three workstreams can perform fund transfers, queue reprioritisation and gridlock resolution in a decentralised manner, without compromising the privacy of the transactions. Each workstream has its own merits and design considerations to meet the requirements. Findings and observations from the three workstreams in Ubin Phase 2 have also contributed to topics such as scalability, performance and resiliency of a DLT-based RTGS system. The project also identified areas where the prototypes can be further improved before becoming fully operationalised.

Ubin Phase 2 has also extended the conversation beyond the technology. The project highlighted six key future considerations which include a point of view on the role of a central bank and regulator in a decentralised payment system. To achieve round-the-clock operations and managing multiple nodes in a cloud environment, rigorous governance, policies and operating models need to be put in place. Although a centralised operator is no longer required, a central bank or regulator still plays a vital role in this critical payment network infrastructure which needs to be redefined.

Building on the success of Phase 1 and this Phase 2, MAS and its partners will continue to journey towards the goal of making Singapore a Smart Financial Centre. Future phases of Project Ubin could focus on a decentralised bonds payments system, which could be supported by MAS and the participant banks with execution driven by Singapore Exchange. This could deliver a more efficient fixed income securities trading and settlement cycle through DLT. MAS will also focus on new methods to conduct cross-border payments, leveraging on the findings from Phase 1 and Phase 2. These are in tandem with the eventual aims of contributing to the community and to develop more efficient alternatives to current financial systems based on DLT.

ACKNOWLEDGEMENT 08

8.1 Project Steering Committee

NO.	NAME	ROLE	ORGANISATION
1	Ong-Ang Ai Boon, Mrs	Co-Chairperson	The Association of Banks in Singapore
2	Sopnendu Mohanty	Co-Chairperson	Monetary Authority of Singapore
3	Toh Wee Kee	Project Director	Monetary Authority of Singapore
4	Stanley Yong	Lead SME Consultant	Monetary Authority of Singapore
5	Nicholas See	Project Director	The Association of Banks in Singapore
6	Daniel Gunawan	Project Director	Accenture
7	Adam Burden	Member	Accenture
8	Zhang Xiao Min	Member	Bank of America Merrill Lynch
9	Ruth Wandhofer	Member	Citi
10	Bhushan Kowshik	Member	Credit Suisse
11	Ng Peng Khim	Member	DBS Bank
12	Jennifer Doherty	Member	Hongkong And Shanghai Banking Corporation Limited
13	Naveen Mallela	Member	J.P. Morgan
14	Michael Truter	Member	Mitsubishi UFJ Financial Group
15	Altona Widjaja	Member	OCBC Bank
16	Kelvin Tan	Member	Singapore Exchange
17	Jamshed Cooper	Member	Standard Chartered Bank
18	Graeme Greenaway	Member	United Overseas Bank

8.2 Project Management Team

NO.	NAME	ROLE	ORGANISATION
1	Toh Wee Kee	Project Director	Monetary Authority of Singapore
2	Nicholas See	Project Director	The Association of Banks in Singapore
3	James Gan Hwa Jaan	Project Lead	Accenture
4	Judy Ng	Project Manager	Accenture
5	Bhushan Kowshik	Product Owner (Corda workstream)	Credit Suisse
6	Tan Hsien Chuang	Product Owner (Hyperledger Fabric workstream)	Mitsubishi UFJ Financial Group
7	Sai Valiveti	Product Owner (Quorum workstream)	J.P. Morgan

8.3 Accenture Delivery Team

NO.	NAME	ROLE
1	Daniel Gunawan	Project Director
2	James Gan Hwa Jaan	Project Lead
3	Judy Ng	Project Manager
4	David Treat	Subject Matter Advisor
5	Luca Schiatti	Subject Matter Advisor
6	Munni Ellwood	Project Facilitator
7	Laks Aravamudhan	Senior Architect
8	Ang Jia Yean	Architect
9	Liew Yen Teen	Business Analyst
10	Reuben Ho Yong Ern	Business Analyst
11	Zhang Wei Jun	Test Architect
12	Adrian Soon	Developer
13	Jake Ryoo	Developer
14	Koh Wen Yao	Developer
15	Lee Han-Rick	Developer
16	Rajiv Raveendran	Developer
17	Triyanto	Developer
18	Lim Hong Yi	Chief Editor
19	Yvonne Liang	Chief Editor
20	Jonathan J Seah	Technical writer, Tester
21	Kenji Sato	Technical writer, Tester
22	Samuel Tang	Technical writer, Tester

8.4 Participating members from Financial Institutions

NO.	NAME	ORGANISATION
1	Amol Pant	Bank of America Merrill Lynch
2	Teo Kian Hui	Bank of America Merrill Lynch
3	Vaibhav Pancholi	Bank of America Merrill Lynch
4	Erhan Saygi	DBS Bank
5	Joan Kim Choo Tay	DBS Bank
6	Prateek Dayal	Hongkong And Shanghai Banking Corporation Limited
7	Shi Zekun	J.P. Morgan
8	Im Junbin	Mitsubishi UFJ Financial Group
9	Reggie Felias	Mitsubishi UFJ Financial Group
10	Andrew Koay	OCBC Bank
11	Tan May Ling	OCBC Bank
12	Aaron Tan	Singapore Exchange
13	Stuart McConnell	Standard Chartered Bank
14	Ong Kit Boone	United Overseas Bank
15	Teo Chze Ping	United Overseas Bank

8.5 Participating members from non Financial Institutions

NO.	NAME	ORGANISATION
1	Coenie Beyers	ConsenSys
2	Peter Munnings	ConsenSys
3	Vinay Mohan	ConsenSys
4	Alan Lim	IBM
5	Shantanu Godbole	IBM
6	Zhao Wei	IBM
7	Connie Leung	Microsoft
8	Vito Chin	Microsoft
9	Antony Lewis	R3
10	Dave Hudson	R3
11	James Carlyle	R3
12	Roger Willis	R3
13	Siddharth Singhal	R3

We would also like to express our appreciation for the following organisations who have supported and contributed to the project: – Accenture Labs at Sophia Antipolis France – Accenture Innovation Center for IBM Technologies (AICIT) – BCS Information Systems Private Limited (BCSIS) – Hitachi Ltd – Microsoft – Singapore Management University (SMU) – Web Synergies (S) Pte Ltd

GLOSSARY

09

ABS	The Association of Banks in Singapore
BLOB	Binary Large Object
DApp	Decentralised Application
DLT	Distributed Ledger Technology
EAF2	Euro Access Frankfurt, Germany payment system, a liquidity saving mechanism
Epic	Large body of work, or group of user stories in Agile methodology
Etcd	An open-source distributed key-value store
FIFO	First-In, First-Out
ILP	Interledger Protocol, an open protocol suit to enable interoperability between different ledgers for payments
JSON	JavaScript Object Notation
LSM	Liquidity Saving Mechanism
MAS	Monetary Authority of Singapore
MEPS+	MAS Electronic Payment System
RTGS	Real Time Gross Settlement
Salt	Random data that is used in cryptography to "hash" a data
SGD	Singapore Dollar
UTXO	Unspent Transaction Output
ZCash	A cryptocurrency from the Zerocoin project
ZKP	Zero knowledge proofs
zk-SNARK	Zero-knowledge Succinct Non-interactive ARgument of Knowledge
ZSL	Zero-knowledge security layer

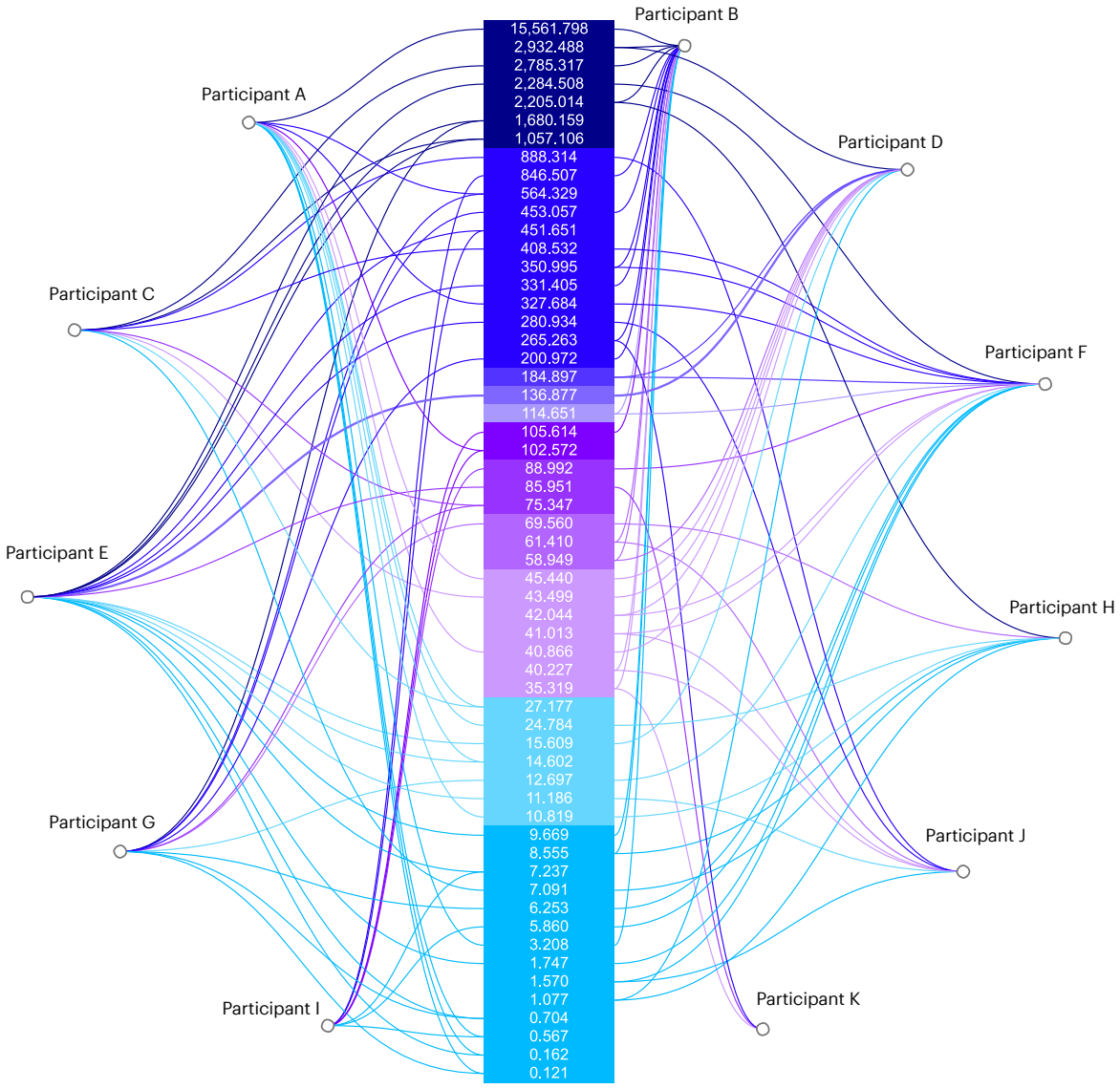
APPENDIX

10

ENABLING DYNAMIC AND INTERACTIVE DATA VISUALISATION

Technology innovation is fundamental to financial services industry transformation. And the Internet has come a long way since 1994. Today, the World Wide Web Consortium (W3C) has advanced Internet standards including: HTML5 (Hypertext Markup Language 5); CSS3 (Cascading Style Sheets 3); SVG (Scalable Vector Graphics) now implemented by all major browsers. By using newer browser capabilities, the Ubin participant has developed the following dynamic and interactive data visualisation to illustrate payment settlements and gridlocks.

Figure 21: 'SuperTree' of Settlement Obligations – contributed by Andrew Koay



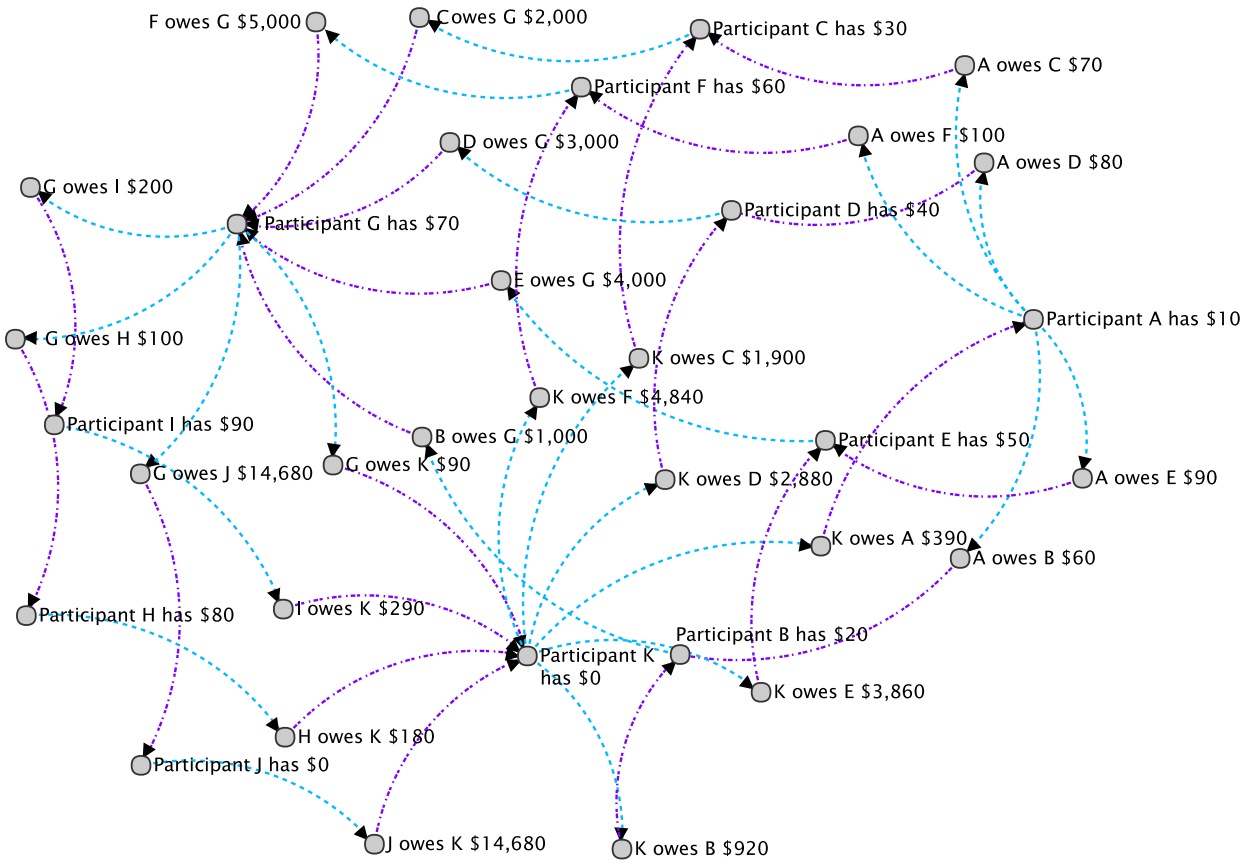
‘SUPERTREE’ OF SETTLEMENT OBLIGATIONS

Figure 21, inspired by Garden By The Bay, illustrates the strength of bilateral relationships, ordered by the value of their settlement obligations. The ‘SuperTree’ looks different when arranged by other sorting criteria.

Graph of Settlement Gridlocks

Figure 22, illustrates 11 participants (combined liquidity of $450K) involved in a settlement gridlock of 23 obligations (value exceeding $60mil). These interactive visualisations were used to enable participants to investigate bilateral and multi-lateral situations and learn how participants are able to reduce liquidity costs (hence, the term Liquidity Saving Mechanism) by minimising the amount of regulatory capital required to cover for settlement obligations.

Figure 22: Settlement Gridlock of 11 Participants in 23 Obligations – contributed by Andrew Koay



MAKING BLOCKCHAIN REAL

Accenture is committed to unlocking blockchain's potential and making it real for our clients.

We apply innovation to achieve real transformation with the ultimate goal of unlocking trapped value in our client's businesses. Through a comprehensive suite of blockchain and distributed ledger technology services and a global team of experts, we help clients move from education and experimentation to production and value—quickly and effectively.

Blockchain will drive profound, positive change. We are working closely with leaders from across industries, regulatory and compliance agencies, the academic community and our key technology alliance partners to move blockchain technology forward so that, ultimately, it can help to improve the way the world lives and works.

EXPERTISE

- Deep industry knowledge focused on redesigning business ecosystems and defining the value
- Active work with Global 2000 clients on strategic uses cases, rapid prototyping, and production system implementation
- A global cross-industry network of relationships and alliances that helps clients engage across their business ecosystems
- Over 300+ dedicated blockchain experts across industries and a much larger bench of blockchain delivery resources
- 10 core hubs in New York, Chicago, Houston, London, Frankfurt, Paris, Dublin, Singapore, Tokyo and Sydney

INNOVATION

- Dedicated research leads focused on developing cutting edge points of view and actionable blockchain research
- 13 patents/patents pending and more in development, including secure 3D model sharing using DLT, antivirus signature distribution, distributed healthcare records management, redaction capability, medical distributed diagnostics, blockchain and hardware security module integration.
- 30+ Accenture Studios for rapid development and prototyping
- Blockchain innovation accelerator service to partner with clients to explore high priority use cases
- Reference architectures across multiple blockchain use cases / typologies and key capability add-ons
- Microsoft & Avanade alliance: preferred access to 35+ blockchain startups in a sandbox environment via Microsoft Blockchain as a Service on the Azure cloud platform
- Accenture Innovation Center for IBM Technologies: 11 hubs and over 45,000 dedicated Accenture practitioners with deep IBM technology skills and extensive executive relations at IBM

ECOSYSTEM

- Founding member of Hyperledger Foundation and the Ethereum Enterprise Alliance, collaborative efforts to create advanced blockchain technology
- Working with over 15 startup companies in our tech labs
- Significant alliances and relationships across all leading cloud providers, Azure, AWS, and IBM
- Strategic alliances with leading blockchain startups such as Ripple and Digital Asset plus several more in progress
- Open innovation process to partner with leading startups

TECHNOLOGY

- Internal blockchain innovation lab focused on R&D activities
- Leading start-up solutions on client-priority use cases
- 7 delivery centres in India, Manila, Shanghai, Bratislava, Milan, Madrid and Recife

Prototypes include: distributed identity, cross-border payments, atomic securities transactions, supply chain track & trace, trade finance, procurement, compliance reporting, hardware security module interface, and much more.

OUR UNIQUE STRENGTHS

Breadth, depth and reach

We are structured in a way that allows us to build deep, personal client relationships where our size helps us to deliver at any scale.

Globally connected

We have an expansive global network, perfectly placed to support our global clients.

Technical experts

We are the world's largest independent technology services provider, with a proven track record of end-to-end transformational projects.

Industry knowledge

Our experienced professionals have deep, hands-on industry knowledge who are ready to advise and navigate our clients through significant technology transformation.

Trusted to deliver

Put simply, we inspire confidence. Our culture of innovation, collaboration and excellence means we always deliver on our promises.

LEGAL NOTICE

ABOUT ACCENTURE

Accenture is a leading global professional services company, providing a broad range of services and solutions in strategy, consulting, digital, technology and operations. Combining unmatched experience and specialized skills across more than 40 industries and all business functions – underpinned by the world's largest delivery network – Accenture works at the intersection of business and technology to help clients improve their performance and create sustainable value for their stakeholders. With approximately 425,000 people serving clients in more than 120 countries, Accenture drives innovation to improve the way the world works and lives. Visit us at www.accenture.com.

173620