EUROPEAN CENTRAL BANK
EUROSYSTEM

BANK OF JAPAN

STELLA - a joint research project of the European Central Bank and the Bank of Japan

# Payment systems: liquidity saving mechanisms in a distributed ledger environment

September 2017

# Contents

EUROPEAN CENTRAL BANK
EUROSYSTEM

STELLA

BANK OF JAPAN

# Project Stella

## Payment systems: liquidity saving mechanisms in a distributed ledger environment

## 1 Background

A Distributed Ledger Technology (DLT) is a set of tools for recording data, such as asset holdings or financial transactions, allowing a network of computers to verify and store updates without a single central management system. In December 2016, the Bank of Japan (BOJ) and the European Central Bank (ECB) announced the launch of a joint research project entitled "Stella" to assess the applicability of DLT solutions in the area of financial market infrastructures. This report is the first outcome of the collaboration.[1]

Project Stella contributes to the ongoing broader debate concerning the usability of DLTs for financial market infrastructures. This joint research builds on the interest of central banks in ensuring that innovations facilitate safer, faster and cheaper financial transactions.

This project is of an exploratory nature within the described limited scope. It serves the sole purpose of assessing whether specific functionalities of existing payment systems could be safely and efficiently run in a DLT application, focusing on hands-on testing only. The areas of cost efficiency, market integration and oversight are left for future study.[2]

The report is structured as follows: section 2 presents the most salient results of the first phase of the project. Section 3 provides background information on the two payment systems, while section 4 describes the set-up of the two test environments. Section 5 assesses whether a DLT-based solution could meet the service levels of existing central bank payment systems in relation to efficiency, with a focus on the implementation of aspects of current liquidity saving mechanisms (LSMs). Section 6

[1] The joint research was conducted by Shuji Kobayakawa (BOJ team leader), Yuji Kawada, Akihiko Watanabe and Akiko Kobayashi from the BOJ, and by Dirk Bullmann (ECB team leader), Frederick Chorley, Cedric Humbert, Thomas Leach and Andrea Pinna from the ECB.

[2] The efficiency and safety of a DLT arrangement are broad concepts which encompass the design, functionality and resource needs of the arrangement (See Committee on Payments and Market Infrastructures (February 2017), "Distributed ledger technology in payment, clearing and settlement - An analytical framework"). This report is, however, a first step in the process of assessing DLTs with a limited focus on some facets of (i) the speed of processing and (ii) operational resilience. Furthermore, it should be taken into account that the analysis contained in this report is based on Hyperledger Fabric version 0.6.1, which is a "developer preview release […] intended to exercise the release logistics and stabilize a set of capabilities for developers to try out" (see release from Hyperledger Fabric dated 16 September 2016).

provides an analogous assessment with respect to safety. Finally, conclusions are drawn in section 7.

## 2 Main findings of the joint analysis

The main findings of the joint analysis, as detailed in this report, can be summarised as follows:

**DLT-based solutions could meet the performance needs of a Real-Time Gross Settlement (RTGS) system:** The analysis found that a DLT application could process volumes of payment requests comparable to those routed to RTGS systems in the euro area and Japan. Taking into account the average traffic of the two centralised payment systems (between ca. 10 and 70 requests per second (RPS)), transactions were processed in less than one second on average. When increasing RPS up to 250, however, the analysis confirmed that the trade-off between traffic and performance was not negligible. More generally, tests proved the feasibility of implementing the processing logic of standard LSMs (queuing and bilateral offsetting) in a DLT environment.

**DLT performance is affected by network size and distance between nodes**: The analysis confirmed the well-known trade-off between network size and performance. Increasing the number of nodes[3] led to an increase in payment execution time. Furthermore, the impact of the distance between nodes on performance was found to depend on the network configuration: provided the minimum number of nodes (quorum) required to achieve consensus was sufficiently close together, the effect of dispersion in the rest of the network on latency was limited. That being said, the nodes on the periphery of the network may produce inconsistencies with the quorum. If the quorum is sufficiently dispersed, the effect on latency will be greater.

**DLT solutions have the potential to strengthen resilience and reliability:** The analysis, while not exhaustive, indicated the potential of a DLT network to withstand issues such as (i) validating node failures and (ii) incorrect data formats. With regard to node failures, it was observed that, as long as the number of nodes required by the consensus algorithm was operational, system availability was not affected. Tests also confirmed that a validating node could recover irrespective of downtime. However, it should also be taken into account that the chosen DLT set-up includes a single certificate authority, which is a single point of failure that could undermine the benefit of distributed validation. Furthermore, tests using incorrect data formats showed the system to be capable of detecting incorrect data formats without affecting overall performance.

[3] Nodes, or "validating nodes", are responsible for gathering and processing transactions to append to the ledger (see annex 2).

## 3 Liquidity saving mechanisms in payment systems

The ECB and the BOJ both have responsibilities in the operation of the RTGS systems, which enable the safe and efficient flow of payments in the respective markets. The performance of these systems is closely interlinked with the fields of monetary policy and financial stability:

- **TARGET2** is the RTGS system owned and operated by the Eurosystem. It settles payments relating to monetary policy operations, interbank and customer payments, and payments relating to the operations of all large-value net settlement systems and other financial market infrastructures (such as securities settlement systems or clearing houses) handling the euro. In 2016, TARGET2 settled an average of €1.8 trillion (approximately 217 trillion yen) in central-bank-money transactions each day, with an average volume of 343,729 transactions. TARGET2 operational sites are located in two different regions which alternate as the primary site for the RTGS system every six months;

- **BOJ-NET Funds Transfer System** is the RTGS system owned and operated by the BOJ. It settles payments stemming from money market transactions, securities transactions, customer payments, monetary policy operations and transactions arising from private-sector net payment systems and other financial market infrastructures. In 2016, BOJ-NET settled an average of 67,326 payments totalling 137 trillion yen (approximately €1.1 trillion) each day. BOJ-NET facilities are located at both its main data centre and its out-of-region backup centre.

RTGS systems settle transactions individually, in real time and on a gross basis. LSMs operate in conjunction with the RTGS system and enable a more efficient use of central bank reserves. TARGET2 and BOJ-NET comprise a host of LSMs including, but not limited to, queuing, bilateral offsetting and multilateral offsetting, albeit with differences in implementation across the two systems (as described in more detail in annex 1).

## 4 Test set-up

**DLT platform:** DLTs allow participants in a network to update their ledgers by means of a consensus mechanism. This means that multiple parties must reach consensus on each transaction, thus enabling enhanced control over the validity and accountability of transactions. DLT platforms are at various stages of development, with differences in access control, consensus algorithms, confidentiality, smart contract programmability and other features. Some DLT platforms, such as Hyperledger Fabric version 0.6.1 (Fabric), on which the analysis was performed, store transactions at each party, which are continuously synchronised by means of an implementation of the Practical Byzantine Fault Tolerance (PBFT) algorithm.

**Code design:** In DLT applications, the business logic for transactions is implemented by means of smart contracts.[4] The ECB and BOJ teams participating in project Stella programmed and ran two types of smart contract: a simple one that processes payments without offering any queuing and offsetting, and another that includes LSMs. LSM smart contracts for the ECB and the BOJ were designed based on queuing and bilateral offsetting mechanisms in TARGET2 and BOJ-NET, respectively.[5]

**Test approach**: The ECB and the BOJ implemented aspects of the logic of LSMs present in their respective systems. To benchmark its efficiency, the code was first run outside a DLT set-up. Then, in order to measure the impact of moving to a DLT without the effects of a distributed network, the smart contract was deployed on a single node without a consensus mechanism. Lastly, the code was run in a distributed environment with a consensus mechanism.

**Test environment**: The ECB conducted its experimental work in a virtualised and restricted in-house test environment,[6] while the BOJ used cloud computing services.[7] The ECB and the BOJ established a series of tests that they carried out in parallel in their respective test environments, and confirmed that the findings were replicable.

**Test data:** The tests were conducted using simulated data. Each fictitious participant in the system was allocated an account and all related information (i.e. account balances, pending transactions) was stored in the ledger. Depending on the specific test performed, input transactions[8] were fed into the DLT application either (i) at a constant rate or (ii) to replicate the pattern of transaction traffic throughout the day,[9] for example peak hour requests, in order to assess the performance of smart contracts in plausible scenarios.

**Measuring performance:** Performance was measured based on the latency of the system. Throughput was set to replicate daily RTGS traffic or up to 250 RPS. To estimate latency, the time taken between (i) a transaction request being sent and (ii) the transaction being executed and written to a block was recorded for each node.[10] For every transaction, the elapsed time across all nodes, or the time at which the quorum of nodes appended the block to their ledger, was calculated.

[4] The term "smart contract" is used throughout the report to identify a collection of program codes deployed and executed on each node to append to the ledger, with no reference to legal aspects.

[5] The algorithms designed for LSMs in the joint research aim to produce similar results to the algorithms in TARGET2 and BOJ-NET, but are not necessarily identical.

[6] Two Red Hat Enterprise Linux 7 (RHEL) machines with 16 virtual cores, 8 GB of RAM and 50 GB of storage hosted the validating nodes and test code.

[7] Each validating node was run on a separate Ubuntu server (16.04.1 LTS 64bit), each with 7.5 GB of RAM and 8 GB of storage. The number of validating nodes and the distance between the nodes were changed in accordance with the test scenarios.

[8] Taking as a reference the number of transactions executed during the peak period of the day, the ECB ran tests consisting of around 200, 700 and 1,000 accounts together with 11,000 transactions, while the BOJ ran tests consisting of around 200 accounts and 38,000 transactions.

[9] Test data generation was based on: Soramäki, K. and Cook, S. (2013), "SinkRank: An Algorithm for Identifying Systemically Important Banks in Payment Systems", *Economics: The Open-Access, Open-Assessment E-Journal*, Vol. 7.

[10] A transaction was recorded in a block when it was settled or placed in a queue.

**Assessing safety:** The joint experiment focused on measuring the impact of three specific scenarios on the functioning of the system[11]: (i) the temporary failure of one or more validating nodes; (ii) the temporary failure of a special node used in Fabric to certify participants, and transaction requests; and (iii) percentages of transactions sent to the system with an incorrect data format. Additional latency brought by such events and the time needed to restore the functionality of the system were the main parameters taken into account in such tests.

# 5 Findings in relation to efficiency

## 5.1 Effect of network size on efficiency

Tests were conducted to verify the impact of an increasing number of validating nodes on performance, in the case of both the simple smart contract (i.e. conducting payment transfers without LSMs) and the LSM smart contract (i.e. conducting payment transfers with LSMs).

### 5.1.1 Results based on the simple smart contract

The simulation of simple payment transfers (without LSMs) confirmed a trade-off between the number of nodes and latency, i.e. the higher the number of nodes, the longer it takes for a payment request to be executed and recorded in a block. While the median latency hovered constantly at around 0.6 seconds in networks of 4-65 nodes, some transactions required longer processing times as the number of nodes increased (see Chart 1). The peak latency reached 1.6 seconds when the number of nodes increased to 65.[12]

### 5.1.2 Results based on the LSM smart contract

Similarly, the tests on the LSM smart contract highlighted a trade-off between the number of nodes and latency (see Chart 2). The latency of LSM transactions was 0.01-0.02 seconds longer than for transactions without LSMs. These tests suggest that the execution of the LSM smart contract in Fabric is not a major factor contributing to latency.

[11] Safety cannot be assessed by means of an all-encompassing test since any payment system is subject to a wide range of sometimes unpredictable threats. Furthermore, a lack of documentation such as detailed explanations of functionality in certain components, test coverage and reliability highlights the immaturity of the DLT set-up at this stage.

[12] The number of validating nodes that could participate in Fabric is limited. Tests were successfully conducted up to 65 nodes.

**Chart 1**

Latency – Simple smart contract

(y-axis: seconds, x-axis: network size)

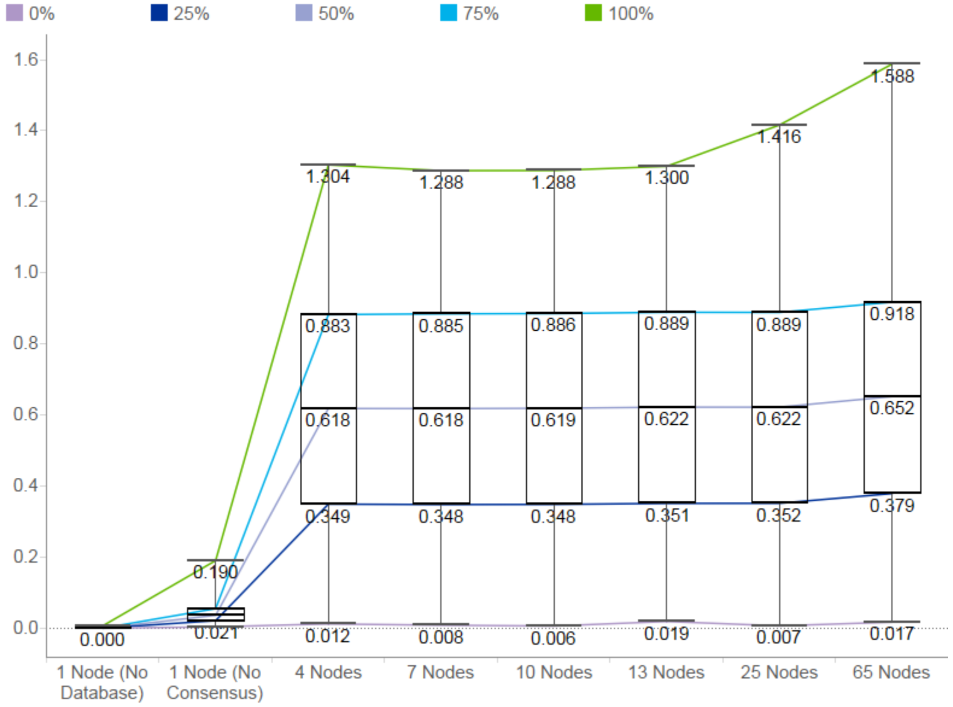
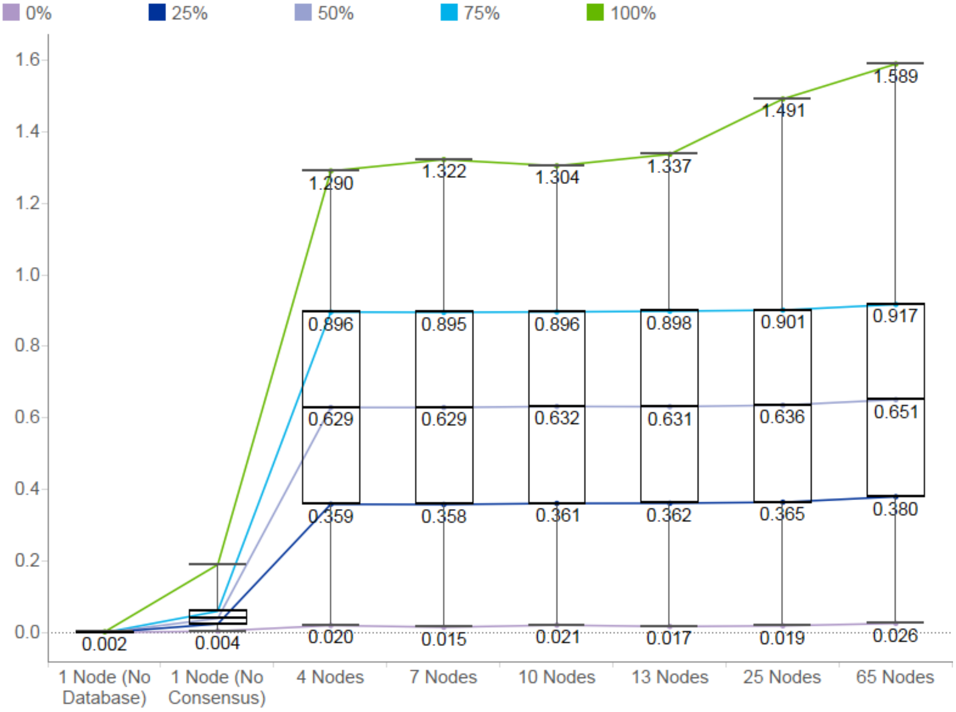


**Chart 2**

Latency – LSM smart contract

(y-axis: seconds, x-axis: network size)



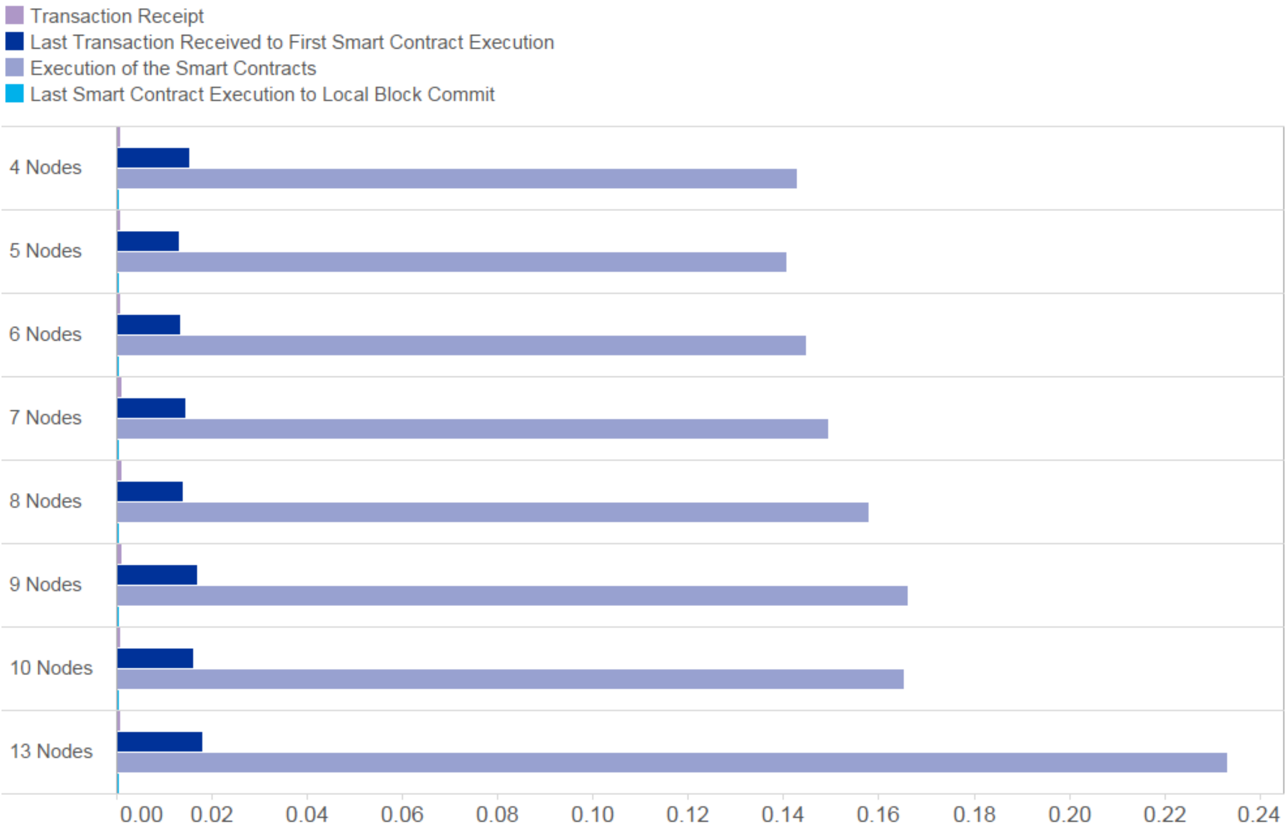
### 5.1.3 Breakdown of latency

To assess why latency increases with the number of nodes, latency was broken down according to the process flow in Fabric. See annex 2 for background information on Fabric and annex 3 for additional information on latency breakdown.

Fabric processes transactions in batches and part of latency is accounted for by the time needed for the transaction requests to fill a batch.[13] Here, it was observed that the average batching time stayed fairly constant at around 0.5 seconds and did not increase with the number of nodes.[14]

**Chart 3**
Breakdown of latency

(y-axis: network size, x-axis: seconds)



Test results after batching time was deducted from the total are depicted in Chart 3. "Execution of the Smart Contracts" accounted for a large portion of latency since

[13] Transactions submitted are grouped together before being executed and placed together in a block (see annex 2). Based on the parameters used for this analysis, transactions were grouped together either once (i) the number of transactions had reached 500 or (ii) one second had passed.

[14] Since RPS was under 500 in the test data, a new block was created every second; on average, it took half a second before processing of the block began.

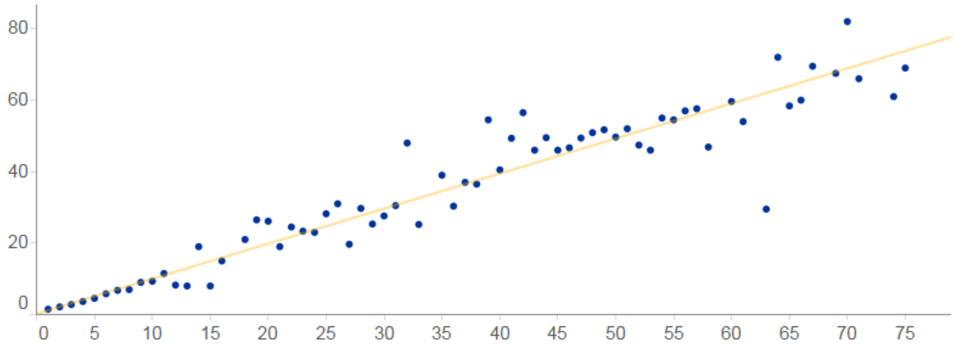
they were executed consecutively.[15] The results also showed that the average latency accounted for by the ordering and communication step (see "Last Transaction Received to First Smart Contract Execution") in the processing of LSMs increased by around 20% when the number of nodes went from 4 to 13. The time it took for each node to commit a block to the chain appears negligible.

Furthermore, there exists a strong correlation between the size of blocks and RPS (see Chart 4).

**Chart 4**

Block sizes and RPS

(y-axis: block size, x-axis: RPS)

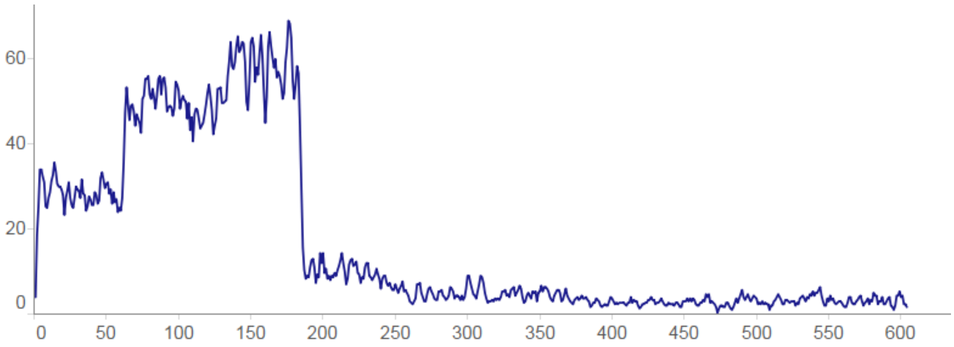


Note: The blue dots are the actual observations. The yellow line represents the trend.

**Chart 5**

Sample RPS during peak hour

(y-axis: RPS, x-axis: time in seconds)

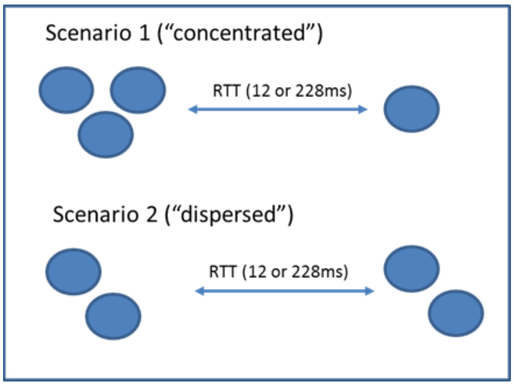


[15] The execution of individual contracts happens in what can be considered constant time, provided sufficient processing power. As a consequence of the serial execution of each individual contract, the more transactions in a block the longer it takes to process the batch.

## 5.2 Effect of distance between nodes on efficiency

Tests were conducted to assess performance in cases where validating nodes were in different locations from one another (i.e. causing communication between them to take longer).

### Scenarios



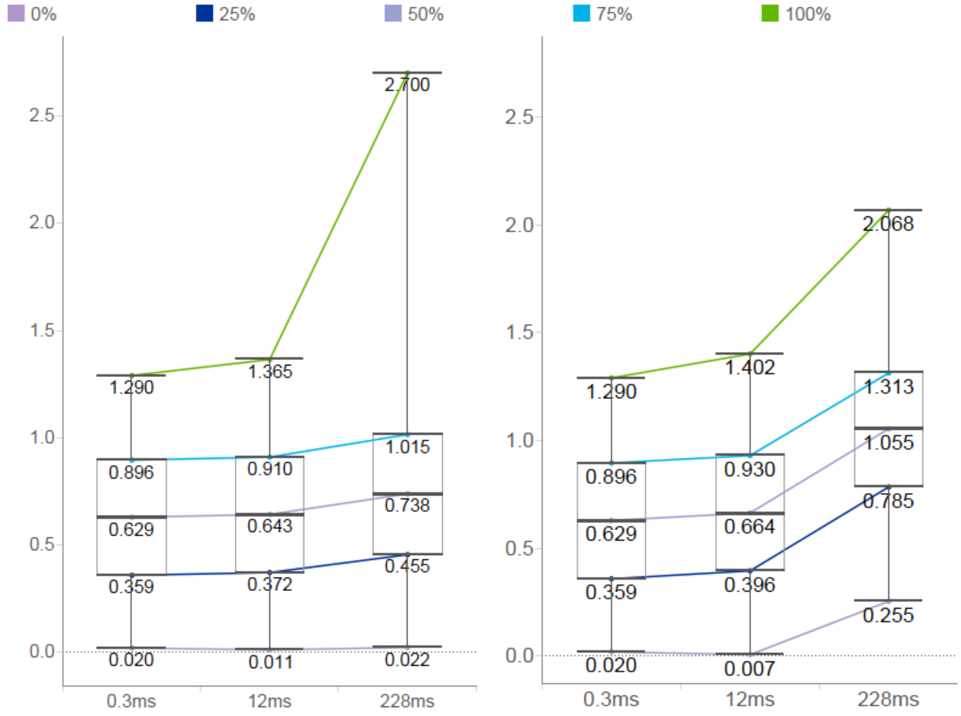
Two scenarios were explored, each based on four nodes. In the first “concentrated” scenario, three nodes were in the same location and the fourth node was separated from the others. The second scenario modelled a “dispersed” network in which the nodes were evenly distributed between two locations (i.e. two nodes in each location). In both scenarios, the distance between the locations was set to have a round trip time (RTT) of (i) 12 milliseconds (i.e. roughly the time needed for a message to cover the distance between Frankfurt and Rome or Tokyo and Osaka), and (ii) 228 milliseconds (i.e. roughly the time taken for a message to travel between Frankfurt and Tokyo).[16]

**Chart 6**

The effect of node location on latency

**The concentrated scenario (left) and dispersed scenario (right)**

(y-axis: seconds, x-axis: RTT)



[16] For the ECB, this was emulated using the “traffic control” command, delaying network traffic between nodes. For the BOJ, nodes were set up in separate cloud computing regions.

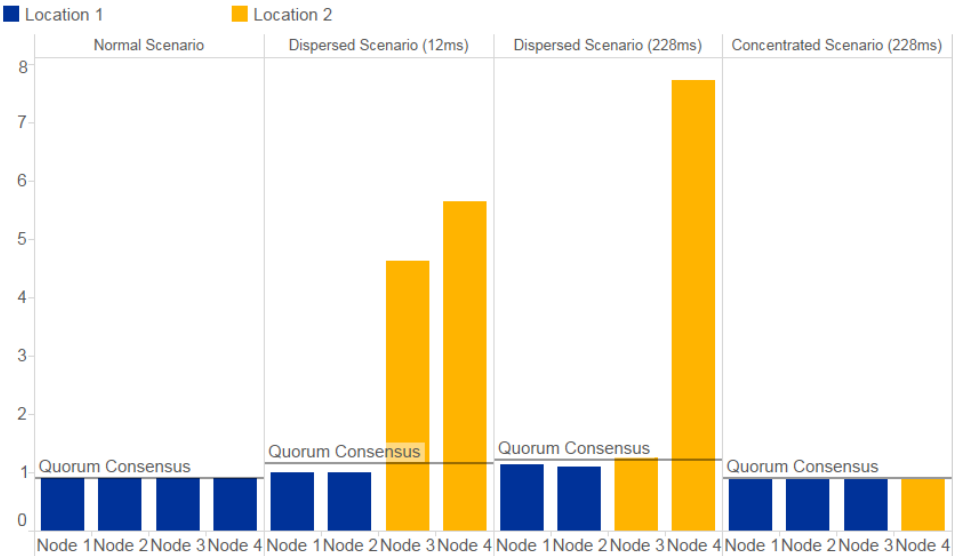
The results obtained in the concentrated scenario (see Charts 6 and 7) showed that performance was less affected owing to the closer proximity of the nodes. The latencies measured across nodes were comparable to a baseline scenario without delay. However, in this scenario, the node separated from the others showed either large latencies (up to 112% higher than in the baseline scenario) or signs of catching up with the other nodes without participating in the consensus process.

The dispersed scenario showed higher latencies as a result of the long distance between the sets of nodes, with an increase in latency of up to 67% in comparison to the baseline scenario.

Results drawn from the two scenarios show that consensus formation is faster, on average, when nodes that need to communicate to achieve consensus (three nodes when using four nodes) are located in close proximity to each other. When the nodes were separated in such a way that required the participation of a distanced node in order to reach consensus, it took more time.

**Chart 7**
The effect of node location on latency

(y-axis: seconds)



Notes - System status is determined by the fact that a quorum has validated a transaction. This latency is taken from the third node appending the transaction to its chain and thus validating this property.
It shall be noted that one node alone can be significantly behind the other nodes in which case its information may be inconsistent with the rest of the ledger results exhibiting this behaviour are omitted.

## 6 Potential impact on safety

### 6.1 Validating node failure

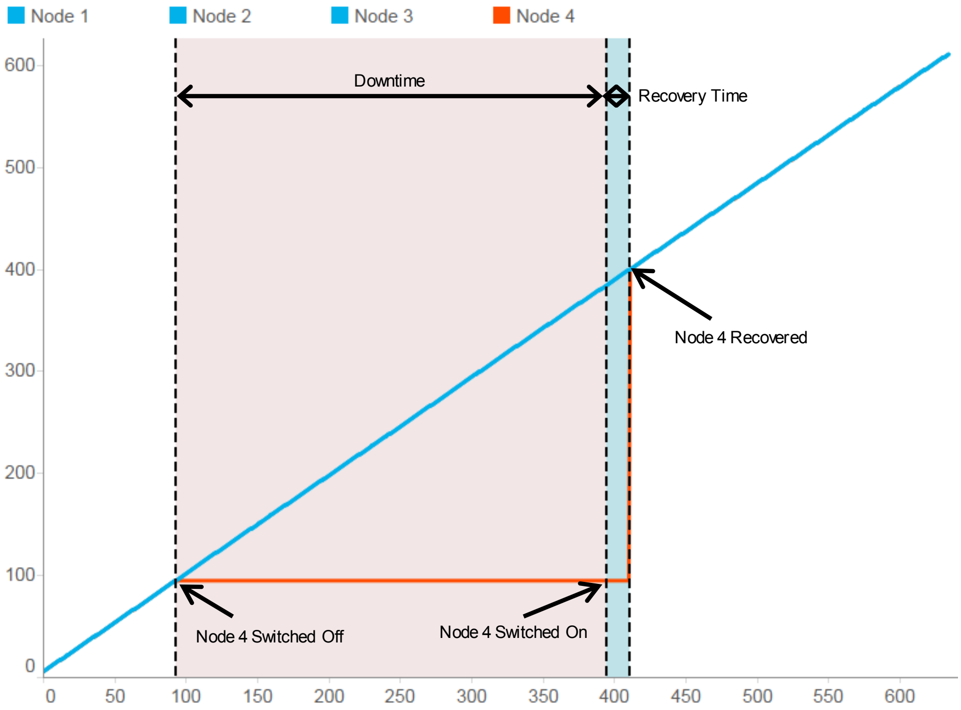
The failure of one or more validating nodes to participate in consensus formation, due to either internal failure or a network disconnection, requires procedures to be put in place to allow the reconnecting node(s) to catch up with the state of the other validating nodes.

Tests were conducted to assess the consequences of a validating node failing. Specifically, one of a total of four nodes was shut down for a given time interval (downtime) and then restarted to measure the time needed for the node to catch up with the other nodes (recovery time, see Chart 8).

**Chart 8**

System availability and recovery with a failing node

(y-axis: block height, x-axis: seconds elapsed)



The tests revealed that, as long as the number of nodes required by the consensus algorithm (three when using four nodes) was operational, the availability of the overall system was not affected by the failure of one node. The cumulative number of blocks (block height) recorded in the blockchain of the three operational nodes gradually increased, while that of the failure node was not updated until it recovered from the failure.

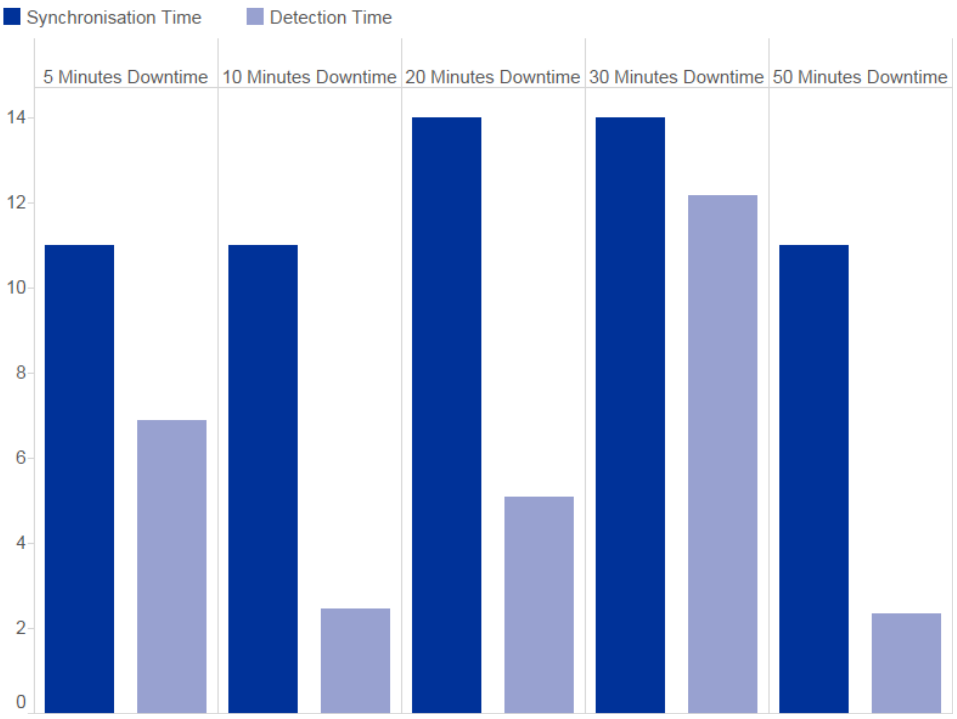
Overall, the test results showed that a validating node recovered in a relatively short period of time (less than 30 seconds) for a range of plausible downtime scenarios (see Chart 9).[17] More specifically, a validating node periodically checked the consistency of its ledger with that of other nodes. Whenever a node detected inconsistencies, its ledger was synchronised with the current state of ledger.

[17] While conducting this test, changes to the default parameters of Fabric were necessary: view changes (change in leader node) needed to be inactivated in order for the failure node to synchronise its ledger following recovery.

Accordingly, the recovery time was broken down into (i) the time required to detect the inconsistency (detection time) and (ii) the time required to synchronise the ledger (synchronisation time). The results showed that, for the payment traffic used for this scenario, the synchronisation time remained fairly constant (11-14 seconds), while the detection time fluctuated (2-13 seconds), reflecting fluctuations in the volume of payment traffic at the time of the restart.

**Chart 9**
Breakdown of recovery time

(y-axis: seconds)



## 6.2 Certificate authority failure

Registering and authenticating participants and transactions is crucial to ensuring the security of the system. Fabric ensures this by means of a certificate authority (CA).

Although transaction validation is distributed by design, Fabric's system architecture provides for a single CA. This introduces a single point of failure to the system. To determine how Fabric deals with the CA becoming unavailable, the CA was stopped and then restarted while the validating nodes kept sending and processing transactions.

The test results showed that, whenever the CA was not available, transactions were rejected, alerting the sending party to Fabric's unavailability. Once the CA became available again, transaction processing began without any other system intervention required.

## 6.3 Resilience to requests with incorrect formats

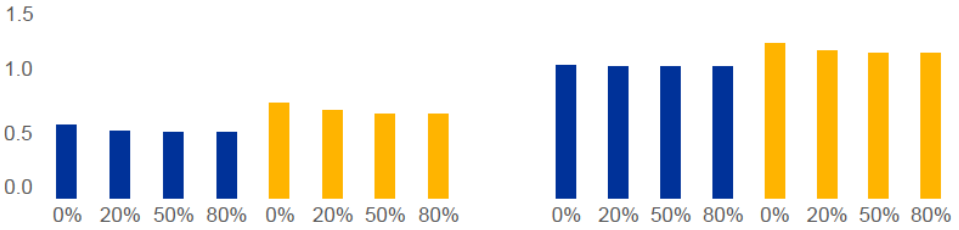
One of the challenges in ensuring the resilience of a DLT system is making sure it could continue to function if a high number of transaction requests with incorrect formats were submitted. This could, for example, be the outcome of unintended behaviour from a participant in the system.[18]

The test was programmed to have 0% to 80% of messages incorrectly formatted. These incorrectly formatted messages triggered an error detection mechanism embedded in the smart contract. The test showed that the system had no difficulties processing transactions with a correct format regardless of the percentage of incorrectly formatted messages. In scenarios in which RPS was 10 and 100, the median and maximum latency remained in the range of 0.5 to 1 second and 1 to 1.3 seconds, respectively (see Chart 10). Rather unsurprisingly, larger flows of transactions required more computational resources to be processed. As the fraction of incorrectly formatted messages was increased, the strain on computational resources was reduced (see Chart 11).

**Chart 10**
Effect of incorrectly formatted transactions on latency

**latency median (left) and latency maximum (right)**

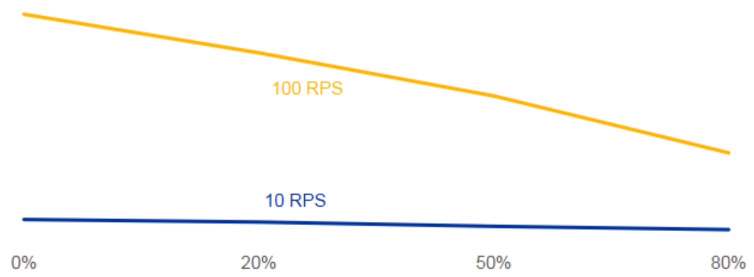
(y-axis: RPS, x-axis: % of incorrectly formatted transactions)



Note: The blue bars represent transactions sent at 10 RPS and the yellow bars represent transactions sent at 100 RPS.

**Chart 11**
Effect of incorrectly formatted transactions on computational resources

(y-axis: CPU Usage, x-axis: % of incorrectly formatted transactions)



[18] For the test, it was assumed that the possible set-up of DLT for market infrastructures operated by the central banks was restricted to members carrying out interbank settlements. Against this background, the risk of flooding packets, dispatched by unknown sources on the internet, appears less pronounced than the risk of one of those members unintentionally disseminating incorrectly formatted messages.

## 7 Summary and conclusion

The ECB and the BOJ, in their role as operators of important market infrastructure services, decided to conduct in-depth experiments to determine whether specific existing functionalities of their respective payment systems could run in a DLT environment.

Findings in relation to efficiency show that, with regard to the specific aspects of RTGS services tested to date, a DLT-based solution could meet the performance needs of current large value payment systems. Given the nature of DLT arrangements, in which the process of validating transactions and reaching consensus is more complex than in a centralised system, this is encouraging evidence. The project also confirmed the well-known trade-off between network size and performance: increasing the number of validating nodes led to an increase in payment execution time. Moreover, the distance between validating nodes has an impact on performance: the time required to process transactions increased with the distance between sets of validating nodes.

The test results also suggest that a range of node configuration and system parameters needs to be taken into account when designing a DLT arrangement. As discussed in this report, the number of nodes, and the distance between these nodes, has a crucial impact on performance. Similarly, system parameters, such as (i) the number of transactions grouped together in a block (linked to the batch size) and (ii) the minimum interval needed to create a new block (timeout), affect overall latency. Node configuration and parameters should be taken into account, depending on the application needs.

In terms of resilience and reliability, the test results provide fresh perspectives underpinned by quantitative results, highlighting DLT's potential to withstand issues such as (i) validating node failures and (ii) incorrect data formats. As for the node failures, the test results confirmed that a validating node could recover in a relatively short period of time irrespective of downtime. The results also showed that transactions were rejected whenever the certificate authority was not available, which could possibly constitute a single point of failure (processing restarted without any other system intervention once the certificate authority became available again). Lastly, the impact of transaction requests with incorrect formats can be managed by smart contracts: here, the system has no difficulties processing transactions in the correct format.

In conclusion, this joint effort has produced a thorough set of results that provide reasons to be optimistic with respect to the capabilities of DLT within payment systems. It is, however, important to bear in mind that this work has been conducted in a test environment; therefore, any assumptions regarding the capacity for DLT to be used in production should not be made from this report.

# Annex 1: Liquidity saving mechanisms in Real-Time Gross Settlement systems

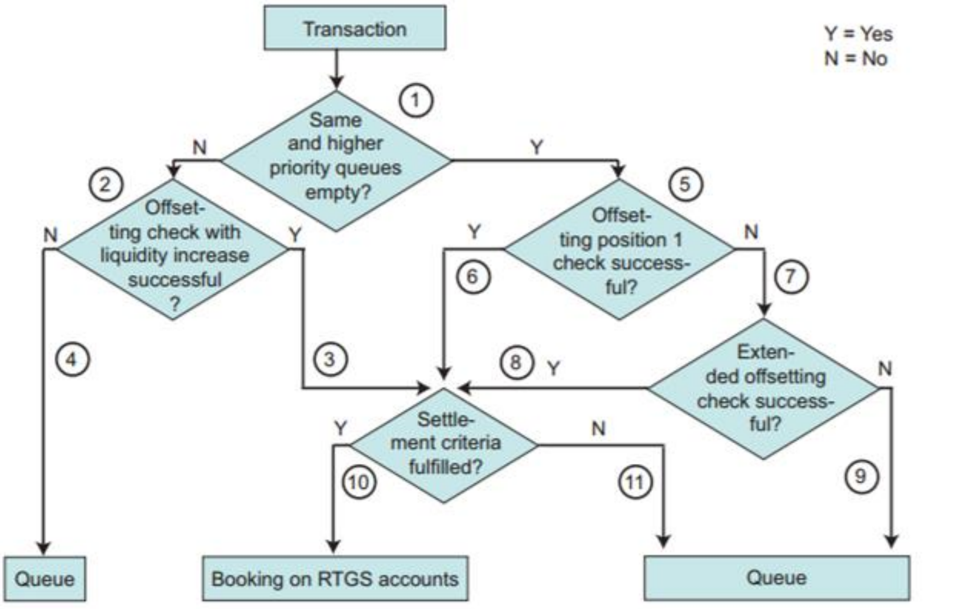
## TARGET2

Entry disposition is the process to which payments are subject following their submission to the system. The basic principles of entry disposition are as follows:

- each payment submitted is assigned a priority: normal, urgent or highly urgent. Payments with no assigned priority will be marked as normal;
- the process attempts to settle payments immediately after they are submitted, with the exception of payments with a later settlement time (i.e. "Earliest Debit Time Indicator");[19]
- payments that cannot be settled immediately are placed into the participant's queue, according to the priority assigned to them.

**Chart A**

Entry disposition algorithm TARGET2



Transaction
Y = Yes
N = No
1
Same and higher priority queues empty?
N
Y
2
Offset-ting check with liquidity increase successful ?
N
Y
5
Offset-ting position 1 check success-ful?
Y
N
6
7
4
3
8
Y
Exten-ded offsetting check success-ful?
N
Settle-ment criteria fulfilled?
Y
N
10
11
9
Queue
Booking on RTGS accounts
Queue

Source: https://www.ecb.europa.eu/paym/t2/shared/pdf/professionals/release_11/T2_UDFS_book_1_v11.0_20170505.pdf

Participants are able to manage the parameters of unsettled payments as follows:

- participants can change the priority of queued transactions;
- participants can reorder queued transactions;

[19] Payments with an "Earliest Debit Time Indicator" will be settled at the time specified wherever possible.

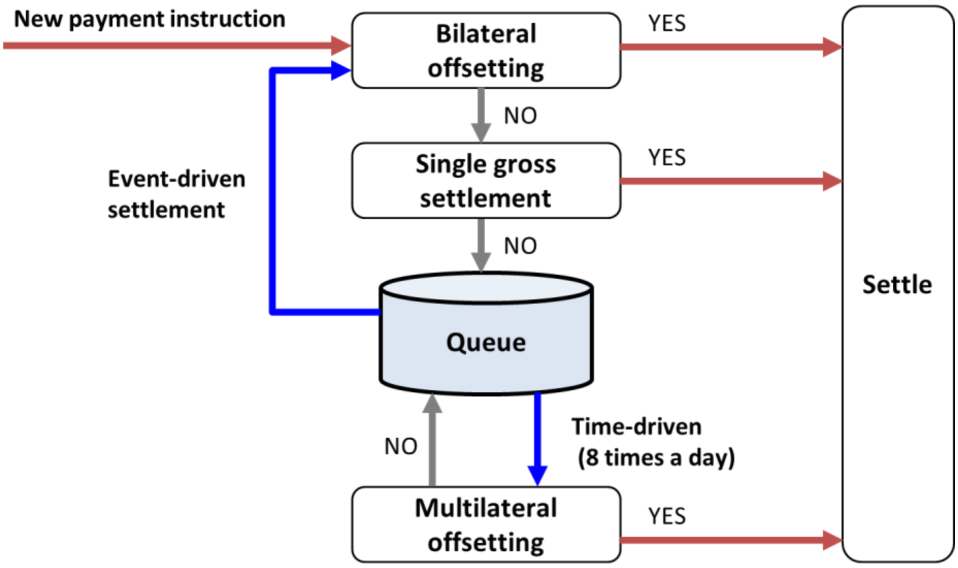
- participants can change the set execution times (if defined prior to submission to TARGET2);
- participants can revoke a queued transaction.

Queue dissolution is handled in two ways. First, certain events in TARGET2 can trigger attempts to settle pending payments in participants' highly urgent and urgent queues. Second, optimisation algorithms run sequentially throughout the day in an attempt to resolve all unsettled transactions. The latter algorithms[20] fall outside the scope of the initial investigation at this stage.

### BOJ-NET

BOJ-NET participants can hold two types of accounts with the BOJ: a standard account for pure gross settlement and a Queuing and Offsetting account (Q/O account) for the use of LSMs.

**Chart B**
Settlement algorithm BOJ-NET



Once a payment has been submitted for settlement in the Q/O account, the bilateral offsetting algorithm first searches for a pair of offsetting payment instructions. For example, when Bank A submits a payment to Bank B, the system searches from the top of Bank B's queue for a payment from Bank B to Bank A that can be settled simultaneously using the balances available. If there is no offsetting payment, the

[20] A full description of these algorithms can be found in the User Detailed Functional Specifications document, pp. 157-170, available at https://www.ecb.europa.eu/paym/t2/shared/pdf/professionals/release_10/UDFS_book_1_v10.0_20160712.pdf?2a6a2ac1bbb113e551b563a6a547188f

system attempts to settle the payment on a gross basis. Payments that cannot be settled immediately are placed in the queue, with “priority” payments placed earlier than “normal” payments.

Bilateral offsetting is also triggered by (i) an increase in balances, and (ii) a change in the payment at the top of the queue. Participants can manage their queues by reordering, revoking or changing the priority of queued transactions.

The multilateral offsetting algorithm runs at set times during the day. It attempts to find the largest set of queued payments that can be settled using the balances available by first determining whether it can settle all queued payments at once, and then removing the largest queued payment from participants with funding shortfalls until a set of payments that causes no funding shortfalls can be found.

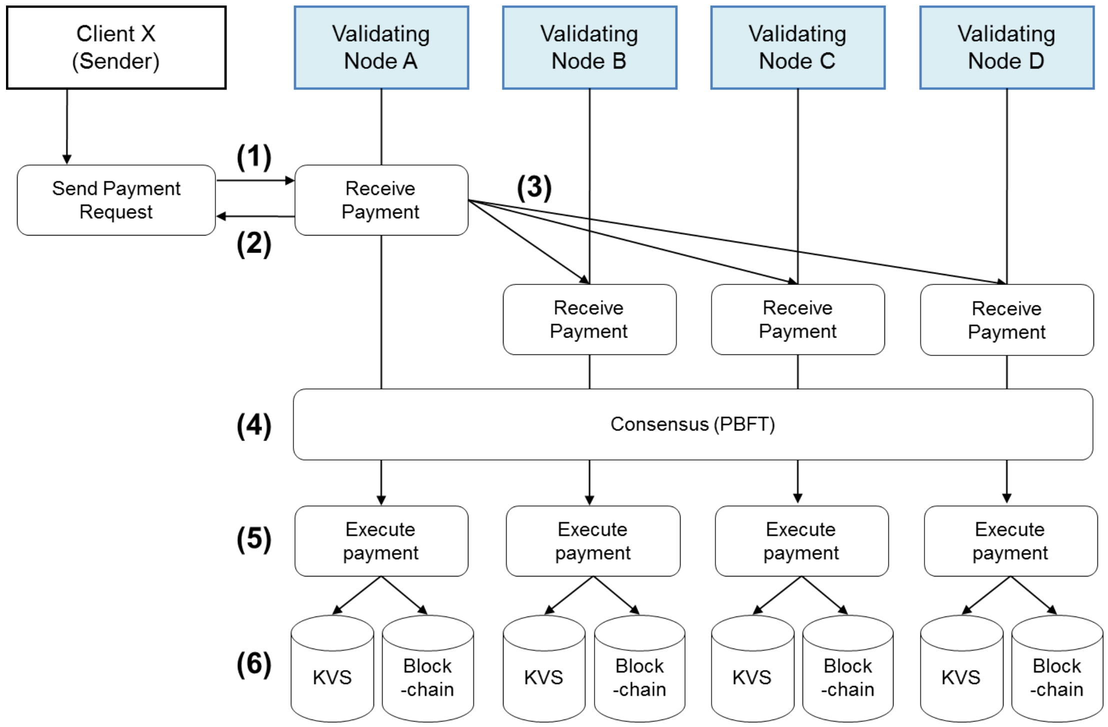
## Annex 2: Background information on Hyperledger Fabric

Fabric v0.6.1 is a DLT platform used to create a restricted blockchain that incorporates byzantine fault tolerant consensus, specifically PBFT.

**Basic architecture:** A Fabric network consists of validating nodes, a certificate authority and client applications. Validating nodes, or simply "nodes", are responsible for endorsing and maintaining the ledger/state by committing transactions. The certificate authority distributes and revokes cryptographic certificates representative of user identities and privileges; use is optional. Client applications send transactions to nodes.

**Chart C**

Process flow Fabric v0.6.1



(1) Client X sends a payment with a digital signature to Validating Node A.
(2) Validating Node A acknowledges receipt of the payment.
(3) Validating Node A broadcasts the ordered payments (pre-prepare message) to the other nodes after its batch is filled or a timeout is triggered.
(4) Validating nodes verify the payments (for example digital signatures, transaction serial numbers) and broadcast two types of messages to the other nodes (prepare/commit messages).
(5) Validating nodes execute the payment using a smart contract after receiving a commit message from the minimum number of nodes required to achieve consensus (in the case of four nodes, three including itself).
(6) Validating nodes record the updated status and create a new block in which to store payment information (for example payment requests, timestamps).

**Process flow**: Transactions can be sent from a client application to any node, but are always forwarded to the leader node. The leader node orders the transactions and broadcasts them to all other nodes for consensus, or agreement, on the proposed order. Once the order of transactions is agreed upon, the transactions are executed and appended to the ledger on each node. A new leader is elected by the other nodes if the current one is suspected to be failing.

Nodes periodically check whether the latest block on the blockchain is consistent with that of the other nodes after processing a predefined number of transactions. If a node detects that its ledger is not up to date, it updates it to synchronise with the others by obtaining any missing information. If the ledgers are inconsistent among the nodes – above the level tolerated by PBFT – the system will fail to process transactions.

*Several technical features of note related to the code design of LSMs are:*

**Data replication**: All nodes share the same copy of the ledger, which can contain data in arbitrary format. This means that current algorithms which rely on some degree of centralised information (such as both sender's and receiver's balances and queues in the case of LSMs) are easier to implement in Fabric than in DLT platforms, in which only a subset of the nodes share information. At the same time, sharing information among all nodes could raise concerns regarding data privacy.

**Deterministic obligations**: The PBFT consensus algorithm implemented in Fabric determines the order of execution for the transactions to be processed. Each node then separately executes a smart contract and amends the ledger accordingly. As a result of this, smart contract processing shall be deterministic in order to produce the same outcome for every node. This implies that there are some limitations or issues to overcome when using non-deterministic values such as timestamps or random values.

**Serial execution**: Fabric executes smart contracts serially, i.e. the same smart contract cannot be executed concurrently. This implies there is little need to use locks to synchronise access to shared resources; however, it also implies the existence of an upper performance limit. For this study, LSM smart contracts for the ECB and the BOJ were designed to be executed serially, unlike the actual algorithms in TARGET2 and BOJ-NET. This type of execution also places limitations on the use of multilateral offsetting mechanisms, as their execution could cause the figures for latency to vary significantly.

*Several technical topics of note related to the measurement of performance are:*

**Parameters that affects performance**: In PBFT “batch” mode (default setting in Fabric), consensus is obtained for a set of transactions to balance throughput versus latency. Latency is therefore largely affected by the time it takes to fill a batch or to wait for the triggering of a timeout prior to the initiation of the PBFT execution. For the purpose of the analysis, default parameter settings in Fabric (batchsize = 500, timeout = 1 second) were used.

**Measuring method**: During querying, PBFT is not executed and results are returned from only one node. This implies that querying cannot easily be used to retrieve the current status of the system. For this study, latency was measured based on the initiation of a transaction by a client and local block commit times.

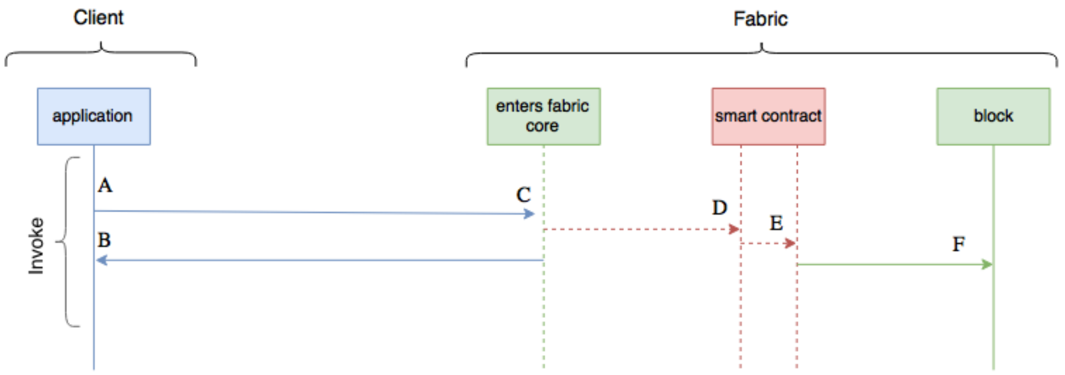
## Annex 3: Breakdown of timestamps

The diagram below summarises the flow of a transaction from its request to its confirmation in a block. During this process six timestamps were collected.

- $A$ – Request sent by the client
- $B$ – The response of the invoke to the client
- $C$ – Transaction received by Fabric core
- $D$ – Smart contract processing begins
- $E$ – Smart contract processing ends
- $F$ – Transaction is committed to a block, as part of a group of transactions

**Chart D**

Breakdown of timestamps



Combining the definitions above with those in Section 5.1.3, we have:

Where $\mathrm{I}$ is the set of all transactions and $\mathrm{K}$ the set of all blocks

- **transaction receipt**: average of $\mathrm{C_i} - \mathrm{A_i}$ for $\mathrm{i} \in \mathrm{I}$
- **last transaction received to first smart contract execution**: average of $\mathrm{D_k} - \mathrm{C_k}$ for $\mathrm{k} \in \mathrm{K}$ where $\mathrm{D_k} = \mathrm{Min}(\mathrm{D_i})$, $\mathrm{C_k} = \mathrm{Max}(\mathrm{C_i})$ for $\mathrm{i}$ in the same block
- **execution of the smart contract**: sum of $\mathrm{E_i} - \mathrm{D_i}$ for i in the same block
- **last smart contract execution to local block commit**: average of $\mathrm{F_k} - \mathrm{E_k}$ for $\mathrm{k} \in \mathrm{K}$ where $\mathrm{E_k} = \mathrm{Max}(\mathrm{D_i})$ for i in the same block and where $\mathrm{F_k}$ is taken arbitrarily since it is the same for all transactions in a block.

Postal address 60640 Frankfurt am Main, Germany
Telephone +49 69 1344 0
Website www.ecb.europa.eu

Postal address 2-1-1 Nihonbashi-Hongokucho, Chuo-ku, Tokyo 103-0021, Japan
Telephone +81 3 3279-1111
Website www.boj.or.jp