ETSI GR PDL 010 V1.1.1 (2021-08)

PDL Operations in Offline Mode

Reference

DGR/PDL-0010_Offline_Mode

Keywords

accountability, ledger, PDL, smart contract

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - APE 7112B
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° w061004871

Contents

Intellectual Property Rights 5

Foreword 5

Modal verbs terminology 5

1 Scope 6

2 References 6
2.1 Normative references 6
2.2 Informative references 6

3 Definition of terms, symbols and abbreviations 6
3.1 Terms 6
3.2 Symbols 7
3.3 Abbreviations 7

4 Introduction To PDL Offline Mode 8
4.1 Introduction 8
4.2 PDL Overview 8
4.3 PDL Application Example 9
4.4 Reasons For PDL Going Offline 10
4.5 Offline Challenges 10

5 PDL Offline Scenarios 11
5.1 Introduction 11
5.2 High-Level Node Reference Architecture 11
5.3 Type of Nodes 12
5.4 Offline Scenarios 13
5.5 Operational Characteristics 13
5.6 Temporal Characteristics 15

6 Technical Issues Arising From Offline Mode 15
6.1 Introduction 15
6.2 Offline Client node(s) 15
6.2.1 General Considerations 15
6.2.2 Securing The Offline Data 16
6.2.3 Enable Access To Ledger Data 16
6.2.4 Enable Smart Contract Operations 16
6.2.5 Enable Data Reconciliation 17
6.3 Offline Validator Node(s) 17
6.3.1 General Considerations 17
6.3.2 Control-Plane Data 17
6.3.3 Proxy Validator Nodes 18
6.3.4 Consensus Violation 18
6.4 Offline Ledger Node(s) 19
6.5 Offline Smart Contract 19
6.6 End-to-End Cybersecurity & Privacy 19
6.7 Monitoring & Orchestration Capabilities 19

7 Proposed Technical Approach 20
7.1 Introduction 20
7.2 Trusted Execution Environments 21
7.2.1 Introduction 21
7.2.2 Hardware TEE 21
7.2.3 Virtualized TEE 22
7.2.4 Local Side-Chain PDL 22
7.2.5 Trusted Third-Party Side-Chain 23
7.3 Offline Operations 23
7.3.1 Introduction 23
7.3.2 Offline Client Node Operations 23

7.3.3 Offline Validator Node Operations 25
7.3.4 Offline Smart Contract Execution 26
7.4 Proxy Mechanisms 26
7.4.1 Introduction 26
7.4.2 What Is A Proxy Node 26
7.4.3 When To Elect A Proxy Node 27
7.4.4 How To Elect Proxy Nodes 27
7.4.5 Operations of Proxy Node 27
7.5 PDL Reconciliation 28
7.5.1 Introduction 28
7.5.2 Ledger Reconciliation 28
7.5.3 Smart Contract Reconciliation 29
7.6 Monitoring and Orchestration 29
7.6.1 Introduction 29
7.6.2 End-to-End Security 30
7.6.3 Privacy 30
7.6.4 Monitoring 30
7.6.5 Orchestration 31

8 Offline PDL Architecture and Procedures 31
8.1 Introduction 31
8.2 PDL Architecture Embodiments 31
8.2.1 Logical Architecture Elements 31
8.2.2 High-Level Architecture 33
8.2.3 3GPP-Aligned Architecture 33
8.3 MANO-Aligned Orchestration Framework 34
8.4 Deployment and Operational Procedures 35
8.4.1 Introduction 35
8.4.2 Offline Client Node Preparations 35
8.4.3 Offline Client Node Operations 36
8.4.4 Offline Validation Node Operations 39

9 Conclusions and Next Steps 39
9.1 Concluding Remarks 39
9.2 Next Steps 40

History 41

Intellectual Property Rights

Foreword

This Group Report (GR) has been produced by ETSI Industry Specification Group (ISG) Permissioned Distributed Ledger (PDL).

Modal verbs terminology

In the present document "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

1 Scope

The present document describes the current challenges related to data storage and ledger operations when a single PDL node or several PDL nodes are offline. This may happen because the nodes are duty cycled, or go offline because of an operational failure or a cyber attack. Operational problems are identified, and possible solutions discussed. Procedures and architecture designs are also presented.

2 References

2.1 Normative references

Normative references are not applicable in the present document.

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] ETSI GR PDL 002 (V1.1.1): "Permissioned Distributed Ledger (PDL); Applicability and Compliance to Data Processing Requirements".

[i.2] Capability Hardware Enhanced RISC Instructions (CHERI); University of Cambridge.

NOTE: Available at https://www.cl.cam.ac.uk/research/security/ctsrd/cheri.

[i.3] ETSI GS NFV-MAN 001: "Network Functions Virtualisation (NFV); Management and Orchestration".

3 Definition of terms, symbols and abbreviations

3.1 Terms

For the purposes of the present document, the following terms apply:

client node: physical node belonging to a client of the PDL, which may or may not be consortium member of the PDL

ledger node: physical node belonging to the PDL which stores data and execution logic in form of Smart Contracts in a distributed and secure manner

offline mode: PDL with at least one element losing connectivity to one, several or all connected elements

online mode: PDL with all of its underlying nodes and protocols being operational and reachable at all times

orchestration: logical capability to provide operational and governance support to the PDL

Permissioned Distributed Ledger (PDL): distributed ledger which is not public, i.e. access permissions are restricted and governed by prior established principles

validator node: physical node belonging to the PDL which is responsible for validating the content provided by the Client nodes before passing it on to the ledge nodes

3.2 Symbols

For the purposes of the present document, the following symbols apply:

N	validated block of the main PDL at time moment N
N'	validated block of the side-chain sPDL at time moment N

3.3 Abbreviations

For the purposes of the present document, the following abbreviations apply:

3GPP	3rd Generation Partnership Project
AI	Artificial Intelligence
CHERI	Capability Hardware Enhanced RISC Instructions
CP	Control Plane
CU	Central Unit
dApp	distributed Application
DDoS	Distributed Denial of Service
DLT	Distributed Ledger Technology
ETSI	European Telecommunications Standards Institute
GDPR	General Data Protection Regulation
HTTP	HyperText Transfer Protocol
IoT	Internet of Things
IP	Internet Protocol
IT	Information Technology
JSON	JavaScript Object Notation
MANO	Management and Orchestration
ML	Machine Learning
OFF	Offline mode
ON	Online mode
OOB	Out Of Band
OS	Operating System
pBFT	practical Byzantine Fault Tolerance
PDL	Permissioned Distributed Ledger
PDLF	PDL Function
PoC	Proof of Concept
PoET	Proof of Elapsed Time
PoS	Proof of Stake
PoW	Proof of Work
RAN	Radio Access Network
RISC	Reduced Instruction Set Computer
RTT	Round Trip Time
SBI	Service-Based Interface
SDK	Software Development Kit
SHA	Secure Hash Algorithm
sPDL	sidechain PDL
SSH	Secure Shell (Protocol)
TCP	Transmission Control Protocol
TEE	Trusted Execution Environment
UK	United Kingdom
VM	Virtual Machine
VPN	Virtual Private Network

4 Introduction To PDL Offline Mode

4.1 Introduction

This clause gives a high-level introduction to Permissioned Distributed Ledgers (PDL) in Offline mode (OFF). To this end, the operational mechanisms of PDL are reviewed first. Then, the present document illustrates a PDL usage scenario and discusses reasons for a PDL going into OFF. Finally, the resulting implications are discussed which lay the foundations for the work carried out in the present document.

4.2 PDL Overview

In the most general case, a ledger is a database. A distributed ledger is thus a distributed database that is consensually shared and synchronized across multiple sites or nodes. The database is typically used through three main interactions:

1) **submit** (content from a client/user onto the ledger);

2) **validate** (and write onto the ledger the submitted content through a consensus protocol); and

3) **read** (the stored content from the ledger).

The content itself has evolved over past years. Initially, only ledger entries could be stored, such as financial transactions, ownership association, etc. However, with the introduction of Ethereum in 2015, execution logic in form of programming code could be stored too. This has led to the emergence of **Smart Contracts** and, as of late, distributed Applications (dApps). The role of distributed ledgers is thus evolving from distributed databases that only store data, to distributed contracts which can take programmatic action on stored or submitted data, to distributed applications that can interface with the clients/users when submitting/reading data.

The **writing of content** onto the ledger typically happens in blocks which are cryptographically linked and, once validated, distributed to all nodes across the entire ledger. The cryptographic linkage and spatial distribution, along with a properly designed validation protocol, make the data written onto the ledger immutable. The distributed nature of the ledger ensures that no central authority can alter content, thus making this technology useful in the context of non-trusted parties interacting with each other.

At the core of each Distributed Ledger Technology (DLT) is the **consensus** protocol, which is being carried out by the validators. It has many roles but mainly ensures that a specific ledger entry cannot appear more than once (thus e.g. preventing the double-spend problem). Different consensus protocols have emerged over past years, such as Proof of Work (PoW); Proof of Stake (PoS), Proof of Elapsed Time (PoET) or practical Byzantine Fault Tolerance (pBFT). They differ in energy efficiency, scale, speed of transactions, among other factors.

Another important aspect is the notion of **public vs. private** ledger. It commonly refers to the degree of anonymity of the validators but also typically extrapolates to the access rights in general, i.e. who can write to and read from the DLT. In the case of a public ledger, validation and access can be done anonymously and by anybody wishing to participate in the ledger. In the case of a private ledger, validation and/or access is restricted to a closed user group such as a consortium (e.g. 10 companies).

Another point to consider is the difference between **permissionless vs permissioned** DLTs. It defines the degree of trust in the validators which execute the consensus protocol. In a permissionless ledger, anyone can participate in the consensus mechanism; whereas, in a permissioned ledger, only those fulfilling certain requirements can take part in the consensus mechanism. Not all consensus protocols are suitable to all scenarios; e.g. permissionless ledgers (such as Bitcoin) would use protocols such as PoW while permissioned ledgers (such as HyperFabric) may use more protocols such as pBFT.

In the present document, solely Permissioned Distributed Ledgers (PDLs) are considered.

4.3 PDL Application Example

Figure 4-1 illustrates an example of a PDL reference use-case scenario, which had been introduced in ETSI GR PDL 002 [i.1]. The scenario pertains to an agricultural application, which is explained in more detail below.

Consider a farmer of a large set of disaggregated land claiming to only be using natural and organic substances, without any chemical and/or genetically modified substances. To prove these credentials, and thus boosting sales, the farmer decides to join a Bio Certification Alliance. The alliance offers bio certification using a PDL, so as to increase transparency to its alliance governance players, to its farmers and to the end consumer wishing to validate the truthfulness of the bio certificate.

At the farmer's side, this is enabled through a set of Internet of Things (IoT) sensors measuring chemical and other pollution throughout the growth and production process. These sensors have their trusted certification and unique digital identity. They constitute the **Client nodes** which transmit information into the PDL for validation. Said validation is done by means of **Validator nodes**. Once validated, the information is immutably written onto the ledger and stored by means of the **Ledger nodes**.

The accuracy of the measurements, and thus the credibility of the bio credentials, depends on the quality and accuracy of the IoT devices and the sampling process. This is referred to as "the last mile" problem. It is beyond the scope of the present document to discuss or solve the last mile problem, but embedding the trusted certification of such IoT devices in the PDL may increase users' trust in the data these devices collect and store in the PDL.

In the context of a PDL, the Validator and Ledger nodes typically belong to a consortium where each member may own a prior agreed set of these nodes. Furthermore, each member or a sub-set of members may offer a set of applications. For instance, a part of the alliance members jointly offers the bio certificate, as long as the sensors in the field support the bio credentials. Another alliance member may offer a smart irrigation service which controls the irrigation system in each of the disaggregated land areas.

Above is enabled through **Smart Contracts** residing in the PDL. Notably, the logic of the Smart Contract will issue a positive certification flag only when all sensors from each of the fields report adherence to bio credentials. The logic can be programmed to perform that check at regular intervals, or be updated when new data from the sensors in the field arrives. Equally, the irrigation Smart Contract will trigger the water valves to be opened when moisture falls below a certain level across a prior agreed set of nodes. Other interesting conditions can be baked in, such as only switching on irrigation when the water price is below a certain threshold (unless irrigation is critical to the survival of the crop). Note that such conditions may be specific for bio credentials (e.g. detection of chemicals) while others may be general (e.g. water cost optimization).

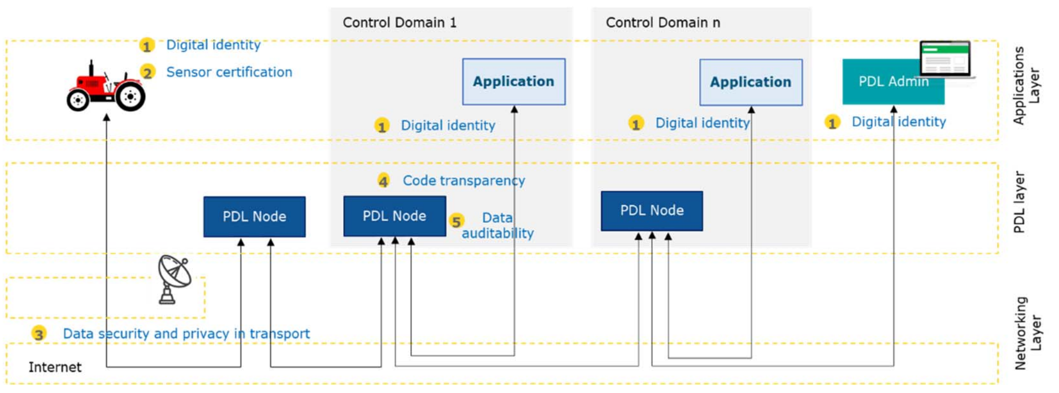

Figure 4-1: Example of a PDL reference use-case scenario from ETSI GR PDL 002 [i.1]

4.4 Reasons For PDL Going Offline

From above example, it is clear that underpinning technology elements of a distributed ledger may occasionally go offline. The reasons are discussed here:

- **By Design:**
  - **Maintenance:** Part of the ledger may require maintenance, such as hardware, networking or software maintenance. Whilst maintenance occurs, the normal operations of the ledger might be impacted which leads to vulnerabilities that need to be dealt with.
  - **Duty Cycle:** To minimize energy consumption of the entire ledger, some of its elements might be duty cycled as per a given and prior agreed schedule. Whilst nodes are offline, possible operational issues as well as vulnerabilities may occur which ought to be dealt with.
- **By Circumstance:**
  - **Temporary Unavailability:** Elements of the ledger may unexpectedly become temporarily unavailable. This could be, for instance, due to an end point using wireless communications moving away from a base station and temporarily losing access to the ledger. Another example occurs in very remote locations where the ledger might be connected via intermittent fixed-line or satellite network.
  - **Congested Network:** In the case of bandwidth-limited or highly congested networks, intra and inter ledger data may not be delivered in time (or delivered at all) and thus compromise the integrity of the ledger operations. That is, all elements are in principle connected but data is not distributed properly. The communication or PDL protocols may have mechanisms in place to resend missing data but that may not happen within the timeframe requirements by the PDL.
- **By Incident:**
  - **Disaster and natural phenomena:** Certain elements of the ledger might be powered by power sources which go out of operations, e.g. due to unforeseeable natural phenomena, such as a lightning strike. The loss of power can cause serious issues to the proper operations of the ledger.
  - **Attack:** In the case of malicious attacks, elements of the ledger can be compromised and rendered intact or offline; in the worst case, the majority quorum capabilities are compromised. This poses serious threats to the integrity and operations of the ledger.

4.5 Offline Challenges

In the case that a PDL or some nodes of a PDL or certain functionality of a PDL are rendered offline, a series of challenges arise which are not normally encountered in fully operational PDLs. These are summarized below and form the rational for the technical approach outlined in the later part of the present document.

From a high level technical and operational point of view, the offline challenges can be categorized as follows:

1) **Security:** In terms of security of the PDL, the following issues arise:

   a) **Consensus Capabilities**, i.e. the consensus approach underpinning the very essence of the PDL may get compromised.

   b) **Weakening Security**, i.e. cryptographic primitives may become unavailable and thus certain processes become unverifiable.

2) **Availability:** In the most general terms, availability is impacted as follows:

   a) **Reconciliation Time**, i.e. when nodes come back online after being offline, it may take some time for them to catch up with the PDL. In the case of nodes suffering intermittent connectivity, it may be possible that by the time they reconcile, the service is interrupted again and they go offline again.

   b) **Side-Chains & Chain Merging**, i.e. the offline mode may trigger the emergence of side chains which has an impact on integrity and availability; and certainly will influence the approach to chain merging. The latter is typically a rare event but may happen very frequently in the discussed scenarios of offline operations.

c) **Stale Transactions**, i.e. if chain merging is not successful, some transactions may become stale which ought to be dealt with by the offline ledger design.

3) **Integrity:** Important issues arise in the context of data integrity:

a) **Control Data**, i.e. data a previously offline node needs to have in order to be accepted back into the PDL network. This typically pertains to authentication data, cryptographical keys, connectivity data, etc. The data related with the content of the ledger itself is out of scope here.

b) **Software Code**, i.e. before being accepted back into the PDL network, checks will need to be performed to ensure that the software/program running on it have not been compromised.

5 PDL Offline Scenarios

5.1 Introduction

This clause discusses possible scenarios resulting from ledgers going offline, as discussed in clause 4. To aid technical understanding, a high-level technical reference architecture is introduced. Thereupon, the role of the different types of nodes is discussed. It is then possible to construct and discuss the different PDL offline scenarios. Last, temporal and spatial characteristics emerging from the different offline scenarios which is unique to a permissioned ledger having offline constituents are discussed.

5.2 High-Level Node Reference Architecture

The application example given in clause 4.3 can be abstracted to a high-level reference Ledger node architecture as illustrated in figure 5-1.

Notably, a set of Client nodes collects data which needs to be written onto the PDL. The data from the Client nodes is transmitted via a fixed or wireless network to the Validator nodes.

The Validator nodes prepare the data for the specific ledger, i.e. sort the data, cast it into a specific format, check for initial consistency, etc. They then perform the PDL-specific consensus protocol to validate the content of the information provided by the Client nodes.

Once validated, the content is written onto the Ledger nodes which store the content for perpetuity. Ledger and Validator nodes belong to several parties.

The end-to-end ledger is configured and maintained by means of PDL-orchestration. Said orchestrator may be implemented on one or several Ledger nodes or on special-purpose orchestration nodes.

The links between the client and Validator nodes and between validator and Ledger nodes might be volatile due to intermittent connectivity or congested networks.

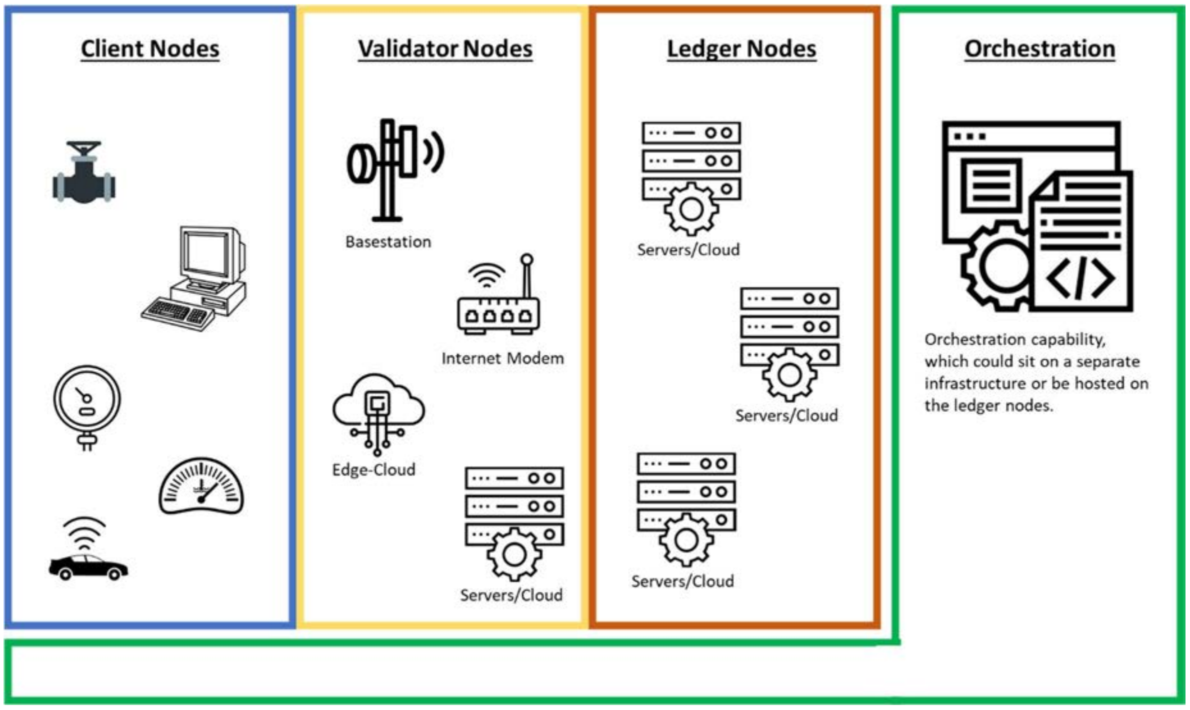

Figure 5-1: High-level node reference architecture comprised of client, validator and Ledger nodes; the system is overseen by orchestration capabilities

5.3 Type of Nodes

As discussed in clause 4.4, there can be several reasons for a node to go offline, such as an engineered duty cycle or an unexpected outage of the wireless network. In this clause, the role of each node type is explained and the likelihood of them going offline estimated.

- **Client nodes:** These are nodes which belong to a client collecting, storing and/or transmitting data. From the ledger's perspective, these are temporary and transient nodes: temporary because a specific Client node may lose connection with the ledger (e.g. a vehicle is going through a patch of poor connectivity) and transient because clients may sign onto the specific PDL service and after a few months/years sign off. Since their presence cannot be guaranteed, they do not take part in the consensus process nor do they form part of the storage ledger. They typically only update their state by sending transactions to Validator nodes.

- **Validator nodes:** These nodes accept transactions from the Client nodes, check the validity and send them to Ledger nodes. Whilst Validator nodes are typically hosted in operationally reliable locations, they can go offline when e.g. there is a network congestion or poor mid/backhaul. For that reason, a strong governance model is required which ensures the viability of the consensus approach when some of the Validator nodes are OFF. For example, a requirement could be that consensus can only be reached if two-thirds of the Validator nodes are online.

- **Ledger nodes:** Ledger nodes are permanent nodes of the ledger. This means that all of the nodes are generally available and online, unless of course in the case of force majeure or a cybersecurity attack. Depending on the design requirements these nodes can or cannot be Validator node. Some architecture models allow Ledger nodes to also serve as Validator nodes and vice versa. The state of the Ledger nodes is being updated by the Validator nodes.

- **PDL-Governance/Orchestration:** Governance is important as it ensures the proper monitoring and execution of the PDL. It is particularly important in the offline scenario as the orchestration capabilities will have to maintain viability of the consensus protocol and ledger operations, among others. It may include roles like the verification of certificates or the revocation of access rights.

Each of these nodes requires trusted digital identities and trusted certificates. Operationally, they can be hosted on bare-metal servers, Virtual Machines (VMs) or containers. Depending on policy and regulation they may be required to be hosted on-premises of the node operator or may be hosted on a public or private cloud.

5.4 Offline Scenarios

The possible scenarios arising due a subset of certain node types becoming unavailable are summarized in table 5-1. Note that when nodes are offline, they may or may not be functional. If they are functional, then they can continue executing tasks locally. Note further that OFF refers to at least one of the node types not being reachable, and not necessarily all of them. The table is sorted from the most likely scenario to the least likely one:

- **Scenario #1:** Is the reference scenario where all nodes are functional and reachable. This should be the modus operandi of the PDL.
- **Scenario #2:** Occasionally, one, several or a cluster of the Client nodes may go offline due to reasons discussed in previous clauses. When this occurs, in most cases the nodes will remain functional, but unable to communicate with other PDL nodes. Situations where the nodes have been rendered dysfunctional or where an entire cluster of nodes has gone offline should also be catered for. It is important to note that when a node goes OFF the reason and functionality of such node are unknown to the other nodes.
- **Scenarios #3 & #4:** Rarely, one or more Validator nodes goes offline for reasons discussed in previous clauses. Again, one needs to cater for situations where the node has been rendered dysfunctional, and not just temporarily gone offline.
- **Scenarios #5-#8:** Extremely unlikely but plausibly, one or more Ledger nodes goes offline for reasons discussed in previous clauses.

The likelihood of any of the scenarios to occur depends on the spatial distribution of the nodes as well as on redundancy/diversity of power and communication utilities, which also impacts the temporal behaviour of the system. Unlike always-on PDLs, Offline PDLs thus suffer from breaks in chain causality. This, in turn, jeopardizes the very essence of distributed ledgers and thus warrants appropriate design attention.

Table 5-1: Possible operating scenarios due to different node types being reachable or offline

	Client nodes	**Validation Nodes**	**Ledger nodes**	***Likelihood***
Scenario #1	ON	ON	ON	*very likely*
Scenario #2	OFF	ON	ON	*occasional*
Scenario #3	ON	OFF	ON	*rare*
Scenario #4	OFF	OFF	ON	*rare*
Scenario #5	ON	ON	OFF	*unlikely*
Scenario #6	OFF	ON	OFF	*unlikely*
Scenario #7	ON	OFF	OFF	*unlikely*
Scenario #8	OFF	OFF	OFF	*unlikely*
NOTE: Nodes that are offline may or may not be functional. Furthermore, OFF refers to at least one of the node types not being reachable. The table is sorted from most likely to least likely.				

5.5 Operational Characteristics

The likelihood of a node or set of nodes going offline depends on spatial and operational characteristics of the system, i.e. the node location and operational provisioning.

The spatial composition of a typical PDL is likely as follows:

- **Client nodes:** These are typically lightweight nodes with an embedded Operating System (OS) and limited processing power, onboard memory and battery power. Depending on the PDL design, they could form part of the PDL or not. In any case, Client nodes write data onto the ledger and may receive instructions from the ledger via Smart Contracts. They are located at the very edge of the network, are typically mobile and often untethered.
- **Validator nodes:** These are typically placed physically close to the Client nodes but within the networking infrastructure. For instance, in a 5G system, Validator nodes could sit in the Central Unit (CU) of the Radio Access Network (RAN), or any other edge-cloud location. They would typically be virtualized via VMs or containers, and have sufficient processing power. Whilst spatially placed in a managed environment, the edge is often connected via unreliable backhaul which makes the operational viability of these nodes volatile.

- **Ledger nodes:** These are typically placed in public or private data centres. They are well powered and well networked. They have sufficient processing power and storage capacity. They are also typically virtualized via Virtual Machines (VMs) or containers.

To minimize operational outage, all nodes in the system should be provisioned as follows:

- **Physical Protection:** Ideally, each node ought to be physically tamper resistant, whether IoT Client node or Validating/Ledger nodes. This will ensure that the certificates stored on the nodes cannot be compromised. Nodes that are exposed to the elements (e.g. a sensor in an agricultural field) should be tested and calibrated at specific intervals to ensure integrity of the information they produce.

- **Power Provisioning:** Each node ought to have one primary and ideally a secondary backup power source so that no power outages occur during the operations of the PDL.

- **Cyber Security:** Each node should be properly protected against cyber security attacks, Distributed Denial of Service (DDoS) attacks and other means of malicious unauthorized access to the nodes.

- **Networking Connection:** Each node ought to be properly networked, using reliable and uncongested networks.

- **Critical Network Path:** The PDL ought to be designed such that there is no single point of failure which is critical to a plurality of nodes. In other words, enough network diversity ought to be provided so that when one network path fails, another can be used to connect the node.

- **Monitoring Capabilities:** The PDL ought to have network, storage and compute monitoring capabilities so that the orchestrator obtains reliable real-time information on the state of the PDL.

Above is an ideal operational scenario which largely holds true for Ledger nodes; however, Validator nodes which reside at the edge of the network will likely have the following differences in a real-world deployment:

- **Networking Connection:** Whilst each node is connected, an edge may experience a poor backhaul network and thus be not reachable over a given period of time.

- **Critical Network Path:** An edge may be connected via a single network path which creates a single point of failure (e.g. a single fibre link or single satellite link) thus not providing sufficient diversity in case that link goes down.

- **Monitoring Capabilities:** Whilst monitoring capabilities might be in place, above-discussed networking problems may render the monitoring, and thus orchestration, capability ineffective. Certain implementations may use out-of-band communications for management and monitoring so that such functions are implemented using different network infrastructure that may survive failure of the main network. Such out-of-band solutions are typically costly and the self-healing features of TCP/IP networks are such that diversified connectivity will offer sufficiently high survivability for the requirements of most PDL scenarios.

Client nodes (e.g. IoT sensors or actuators) will be even less equipped to operate in ideal operating conditions:

- **Physically Protection:** Client nodes in the field are often exposed and without supervision, thus making it difficult to guarantee physical protection and tamper resistance consistently. Periodical testing and calibration may be required to ensure credibility.

- **Power Provisioning:** Client nodes in the field may experience power supply issues such as batteries running low, solar panels not producing enough power, etc.

- **Cyber Security:** Client nodes in the field are also exposed to cyber attacks, mainly because embedded systems are not security patched at the same level as traditional IT equipment.

- **Networking Connection:** For nodes in the field, networking may be volatile over a short period due to wireless network congestion or fading/shadowing blockage. The outage may also be seasonal due to vegetation growth or rain in satellite connections obstructing an already weak wireless link.

- **Critical Network Path:** Client nodes rarely have networking diversity, i.e. they are typically connected via one networking interface only. Therefore, if that link breaks, the node is not accessible.

- **Monitoring Capabilities:** Due to the embedded nature of the devices, monitoring capabilities will be limited. Furthermore, the duty cycling of the devices and the intermittent networking conditions may disrupt real-time monitoring periodically.

5.6 Temporal Characteristics

An important element of Offline PDLs are the temporal characteristics of certain nodes, differentiated between the following levels of planned and unplanned intermittence:

- **Schedule:** When a node goes on scheduled OFF on purpose by the owner(s). This could be done in the context of duty cycling to save energy or for planned maintenance. Whilst the schedule may have a limited impact onto the node itself, it may have a strong impact on any dependent Smart Contract. For instance, a Smart Contract may only work when a certain number of nodes provides information within a given time window which will require the schedules of the nodes to be synchronized. Clearly, there is a strong governance issue when it comes to schedules, which in the context of PDLs, goes beyond simple energy savings.

- **Graceful Disappearance:** Client nodes or even validating nodes can disappear gracefully in that the connection becomes weaker and weaker over time. In the case of a validating node, this could be because the network has peaks of congestion which increase response times until the outage occurs. Equally, Client nodes may suffer from network congestion or simply a weakening wireless connection as the Client node moves away from the base station/access point. In principle, the PDL system could be programmed to recognize such events and initiate contingency protocols before the connection is cut off. For instance, in the case of the validating node, another or a prior-established Proxy node could take over the role of a validating node. In the case of a Client node, this may give an opportunity to establish a trusted local environment which is able to operate offline for a given period of time.

- **Sudden Disappearance:** Nodes may of course disappear suddenly, without any prior warnings or indications. The reasons could be manifold and range from non-malicious due to power cut, equipment failure, or sudden network blockage; to malicious due to a DDoS attack or other cybersecurity breach.

In the context of temporal characteristics, it is important to pay attention to the **causality of the events**. For instance, the order in which some Client nodes or validating nodes become unavailable is important to the proper functioning of the PDL and Smart Contracts. It will impact recovery/auto-recovery protocols, the way majority quorum is determined, the way governance is being executed, among others.

6 Technical Issues Arising From Offline Mode

6.1 Introduction

This clause discusses the technical issues which arise from elements of the PDL going offline. Discussed here are technical issues arising from the points of view of a Client, Validator and Ledger node. Furthermore, specific issues arising in the context of Smart Contracts are discussed, as well as governance and cybersecurity approaches.

6.2 Offline Client node(s)

6.2.1 General Considerations

In the context of Client nodes, technical issues arise in the following situations:

- **Client Data:** A major concern is securing and ensuring a trusted environment for a Client node when OFF.

- **Ledger Data:** Client nodes may need access to data residing on the PDL which is not possible when OFF. Such clients may only have access to data stored locally (if any).

- **Smart Contracts:** Smart Contracts need to remain operational in the event that Client nodes are OFF.

- **Chain Reconciliation:** A major challenge is to enable the reconciliation with the main chain, once the OFF node gets connected again.

These issues are now discussed in greater detail in subsequent clauses.

6.2.2 Securing The Offline Data

Securing the content and fidelity of inferred data in a Client node, e.g. a remote IoT node, is a challenge whether the node is online or offline. However, the online mode allows for regular monitoring and reporting which is not possible in the offline mode. The data in the Client node can be compromised through the following (non-exhaustive list of) mechanisms:

- **Challenge #1 - Deteriorating Calibration:** The data collected and processed in Client nodes is typically calibrated, or requires Artificial Intelligence (AI)/Machine Learning (ML) inference models. These are rarely static and more often than not require constant updating. Given the technical constraints of the Client nodes, the algorithmic updates of the calibration and inference algorithms is done centrally or close to the edge. In either case, it remains inaccessible to the Client node. As a result, the fidelity and accuracy of the inferred data deteriorates and thus becomes of little use to the overall PDL operations. Therefore, an important technical challenge is to find mechanisms which prevent or minimize the impact of such deteriorating calibration.

- **Challenge #2 - Cybersecurity Breach:** Being out of reach of the PDL, regular monitoring is not feasible anymore. For instance, the validity of certificates installed on the Client node cannot be verified through advanced hashing (e.g. SHA-256) and a secure communications tunnel (e.g. VPN). This, in turn, lowers the defences and allows malicious hackers to gain access to the Client nodes by gaining access to the last mile network and/or be in proximity to exploit the wireless network to which the node is connected. Therefore, an important technical challenge is to ensure that access to the Client node is sufficiently secure to prevent malicious network attacks.

- **Challenge #3 - Physical Tampering:** In offline mode, physical access to nodes can go unnoticed. A malicious adversary may physically access the node by opening the enclosure, reading data from the node's memory, changing or reprogramming the logic on the chips, etc. When the node then re-joins the PDL, it may gain access to the overall PDL with potentially serious implications. Therefore, an important technical challenge is to create sufficiently secure trusted environments which eliminate or minimize such risk.

6.2.3 Enable Access To Ledger Data

Client nodes may need access to data which resides on the PDL or which is managed through the PDL. The following technical challenges thus arise:

- **Challenge #4 - Access to Ledger Data:** For instance, a Client node may need historical data which resides on the PDL. When the node is OFF (disconnected from the PDL) such data is not accessible. A technical challenge is therefore to cater for mechanisms which avoid or minimize the impact of such a scenario.

- **Challenge #5 - Access to 3rd Party Database Data:** Idem to challenge #4 but related to accessing data from a 3rd party database with credentials administered through the PDL. There is a need to cater for mechanisms which avoid or minimize the impact of such a scenario.

6.2.4 Enable Smart Contract Operations

Smart Contracts may require data from Client nodes when writing data to the PDL or may need to instruct Client nodes based upon information read from the PDL. In the context of offline operations, the following technical challenges thus arise:

- **Challenge #6 - Read Data from Client nodes (upstream):** A Smart Contract may require data from nodes for a certain logic or conditions to be triggered. That data ingest may happen regularly or when needed. In offline mode, an important technical challenge is to address the impact absence or late arrival of data from Client nodes has on the viability of the Smart Contract.

- **Challenge #7 - Instruct Client nodes (downstream):** Likewise, the Smart Contract may trigger instructions to a Client node. That trigger may happen regularly or be triggered through Smart Contract logic. In offline mode, an important technical challenge is to address the impact the absence or late arrival of the control data to the Client nodes.

6.2.5 Enable Data Reconciliation

Once the Client nodes return to normal online operations, the data between the Client nodes and the PDL needs to be reconciled. To this end, the following challenges arise:

- **Challenge #8 - Data Validation:** Any data inferred or generated in the Client node whilst offline needs to be validated by the PDL system before being added to the main PDL. A technical challenge is thus to enable data validation so that only trust-worthy data is added.
- **Challenge #9 - Data Reconciliation:** Once validated, said data needs to be added to the PDL. A technical challenge is thus to enable a post-event reconciliation of data generated earlier than the current PDL time stamp.

6.3 Offline Validator Node(s)

6.3.1 General Considerations

As per clause 5.4, it is unlikely that Validator nodes go offline but it may still occasionally happen. PDL needs to be prepared for such an occasion. To this end, the technical issues arising can be summarized as follows:

- **Control-Plane Data:** Whilst offline, a major concern is how to ensure that the PDL orchestrator is able to deliver/update information important to the operational part of the Validator nodes, i.e. new validation approach, security certificates, access instructions to 3rd party databases, etc.
- **Proxy Validator nodes:** The controlled nature of the Validator nodes allows the establishment of Proxy Validator nodes which become operational once a given Validator node goes offline.
- **Consensus Violation:** A major challenge is to ensure that consensus majority rules are not violated with a given set of Validator nodes going offline.

These issues are now discussed in greater detail in subsequent clauses.

6.3.2 Control-Plane Data

The operational health of the Validator nodes is crucial to the operations of the PDL. Validator nodes typically receive control data from the PDL and/or its orchestration framework. Not having access to this control-plane data because the Validator node has gone offline poses a series of technical challenges which are summarized below:

- **Challenge #10 - PDL Control Plane:** The PDL Validator nodes receive important control data from the PDL or orchestration frameworks. Examples of this are security certificates of the Validating PDL nodes, security certificates of specific Client nodes; specific algorithmic validation frameworks; timing and synchronization data; security credentials to 3rd party ledgers or data and networking infrastructure, etc. An important technical challenge is therefore to ensure that the overall operations of the PDL is not jeopardized when said control-plane data cannot be delivered to some of the Validator nodes. This has two aspects:

  a) Ensuring that an OFF node, if still functional (meaning it is powered on and can communicate with some Client nodes but is isolated from the rest of the PDL and specifically the orchestration functionality) is pre-programmed to limit its functionality to specific pre-defined autonomous actions (e.g. in an agricultural plant it can still collect data from Client nodes but it cannot send new irrigation instructions; or, in the case of autonomous vehicles, it can bring the vehicle to a safe halt but it cannot restart the autonomous drive afterwards until communications has resumed).

b) Ensuring the rest of the PDL can continue operation in absence of the isolated/offline nodes and ensuring the PDL does not resume operations if certain criteria are not met (e.g. allow continued operation as long as enough Validator nodes are still on-line, but halt operations if an insufficient number of Validator nodes is on-line or other functionalities, such as orchestration, are below certain performance criteria).

- **Challenge #11 - 3rd Party Data Access:** Similarly, to challenge #10, some Validator nodes may require access to 3rd party databases which may not be accessible when the Validator node is OFF. There are two scenarios to consider:

  a) The Validator node does not have connectivity to the 3rd party database, in which case it cannot access such data.

  b) The Validator node has connectivity to the 3rd party database but is isolated from the PDL and cannot obtain credentials or certificates in order to access that database.

  In both cases, the Validator node will not have access to the 3rd party database and should be pre-programmed (if OFF) or instructed by the orchestrator (if on-line) to take appropriate actions. Such actions need to be defined when designing the PDL.

6.3.3 Proxy Validator Nodes

Depending on PDL type, the number of Validator nodes may be limited either by design or by technical/performance limitations. A Validator node going offline may thus pose a serious operational risk. A Proxy Validator node can take over the validation functionality of an OFF Validator node. The technical challenges related to this can be summarized as follows:

- **Challenge #12 - Proxy node:** The role of the Proxy node is to take over the validation work once the main Validation node goes offline. The controlled nature of the PDL environment allows for the establishment of Proxy Validator nodes. These become operational once the main Validator node(s) disappear or suffer from a graceful but measurable performance/connection degradation. An important technical challenge is to ensure the control data (certificates, validation protocols, etc.) is synchronized between Validators and their respective Proxy nodes. Furthermore, the Proxy node election process needs to be provided, along with the monitoring and decision frameworks (e.g. pre-assignment of Proxy to each Validator node, pre-assignment of a pool of Proxies and the selection process of a specific Proxy from within that pool which could be random or round-robin or topology-based, etc.) Also, a framework needs to be provided which establishes the process once the original Validator node comes back online (e.g. restore its status as a Validator node, or put it in the pool of Proxies in standby).

- **Challenge #13 - Disrupting Ongoing Validation Session:** When a Proxy Validator node becomes active, the validation process that was performed by the now OFF primary Validation node might have been interrupted. A technical challenge is thus to ensure that an ongoing validation operation can either be seamlessly handed over to a Proxy node or restarted by the Proxy node.

6.3.4 Consensus Violation

As discussed in clause 6.3.3, the number of Validator nodes in PDL may be limited. A simple outage or cyber attack might reduce the number of operational Validator nodes thus hamper the validation consensus mechanisms in a manner that exposes the PDL to attacks. The technical challenge related to this can be summarized as follows:

- **Challenge #14 - Consensus Violation:** An important technical challenge is to ensure that the majority consensus validation rules remain intact and are not prone to possible cyberattacks on the PDL. The mechanisms should cater for Proxy nodes too. A practical example may be that a 2/3 majority is required for a specific consensus mechanism. If the number of validators drops down to three, and one of the three is hacked the consensus process is flawed. Thus, a possible solution may be to declare a minimum number of active validators (the exact number may depend on security levels and other risk mitigation factors) and define a situation where the number of validators drops below that minima as "Consensus Violation". The actions that the PDL takes when Consensus Violation occurs need to be properly designed (e.g. halt operations of the PDL until the number of validators meets the minimum requirements and notify the users/administrators).

6.4 Offline Ledger Node(s)

As per clause 5.4, it is extremely unlikely that PDL nodes go offline. Furthermore, the PDL protocol should be able to cope with the outage of any number of Ledger nodes as they are generally only used to enable the distributed nature of the PDL.

Therefore, no further technical challenges are discussed here.

6.5 Offline Smart Contract

A Smart Contract would typically reside on the PDL. This clause summarizes specific technical challenges arising from Smart Contracts going OFF:

- **Challenge #15 - Synchronize Data Causality:** Challenges #6 and #7 arise when there is a linear relationship between Client nodes and Smart Contracts. However, the situation can be significantly more complex: For instance, the Smart Contract may require input from several Client nodes and send instructions back to several (possibly other) Client nodes. These Client nodes may go offline in a non-predictable pattern, thus interfering the causal logic of the Smart Contract. To support complex internal Smart Contract logic or a virtualized peer-to-peer network between nodes (via the Smart Contract of the PDL), an important technical challenge is thus to minimize the impact of breaking the data causality due to Client nodes going offline and coming back online in an unpredictable manner.

- **Challenge #16 - Offline Smart Contract Operations:** A possible solution to mitigating issues arising under challenge #15 is to permit the duplication of Smart Contracts into an offline environment. This, however, triggers an avalanche of technical challenges, such as the provisioning of a trusted execution environment for the duplicated Smart Contract; the ability to reconcile the generated data with the main ledger, access rights management, among others.

6.6 End-to-End Cybersecurity & Privacy

Given the non-causal nature of a PDL with offline nodes, end-to-end security needs to be managed more carefully. Following are some important challenges:

- **Challenge #17 - Consistent End-to-End Security Provisioning:** A major technical challenge is to ensure that the security of the ledger and its constituents is not jeopardized under any possible operating condition. This pertains to the validity and provenance of security certificates, encryption keys and protocols, hash functions, majority consensus algorithms, among others.

- **Challenge #18 - Ensure PDL Member Privacy:** Whilst the privacy of the data collected, inferred and processed by the Client nodes is out of scope, the privacy requirements of the members being part of the PDL may be subject to regulations or laws (e.g. GDPR) or company policy. For instance, some consortium members may not want other members to know about specific operating conditions or decision logic. An important technical challenge is thus to guarantee (to the extent possible) the privacy of the consortium members by implementing best practice algorithmic frameworks which guarantee the privacy whilst not jeopardizing the operations of the ledger.

6.7 Monitoring & Orchestration Capabilities

Last but not least, the ledger orchestration and monitoring capabilities are discussed in this clause. The technical challenges can be summarized as follows:

- **Challenge #19 - Monitoring Capabilities:** The ability to observe the sudden or graceful deterioration of networking links or processing capabilities of Client and Validator nodes very much depends on the monitoring capabilities of the PDL. A technical challenge is thus to enable such monitoring capabilities across compute, storage and networking resources; whilst not creating a large operational overhead. The monitoring and probing should extend to virtualized environments, such as VMs or containers. Furthermore, alerting capabilities should be set up and triggered when monitoring identifies certain conditions.

- **Challenge #20 - Orchestration Capabilities:** An important technical challenge is to enable PDL-wide orchestration capabilities which ensure a proper deployment, operations and decommission of the PDL. E.g. the orchestrator, among other, should be able to synchronize duty cycle schedules of Client nodes such that Smart Contract operations are not jeopardized.

7 Proposed Technical Approach

7.1 Introduction

This clause introduces a set of (non-exhaustive) technical approaches which address the technical challenges outlined in clause 6. Table 7-1 summarizes how the technical solutions of this clause address the challenges. The technical solutions are discussed in subsequent clauses.

Table 7-1: Summary of technical challenges as discussed in clause 6, and how they can be addressed as discussed in this clause 7

	Trusted Environments	Offline Operations	Proxy Mechanisms	Ledger Reconciliation	Monitoring & Orchestration
Challenge #1 - Deteriorating Calibration		🟩			
Challenge #2 - Cybersecurity Breach	🟩	🟩			
Challenge #3 - Physical Tampering	🟩	🟩			
Challenge #4 - Access To Ledger Data	🟩	🟩			
Challenge #5 - Access To 3rd Party Database Data	🟩	🟩			
Challenge #6 - Read Data From Client nodes (upstream)	🟩				
Challenge #7 - Instruct Client nodes (downstream)	🟩	🟩			
Challenge #8 - Data Validation	🟩	🟩			
Challenge #9 - Data Reconciliation	🟩	🟩			
Challenge #10 - PDL Control Plane	🟩		🟩		
Challenge #11 - 3rd Party Data Access	🟩		🟩		
Challenge #12 - Proxy node	🟩		🟩		🟩
Challenge #13 - Disrupting Ongoing Validation Session	🟩		🟩		
Challenge #14 - Consensus Violation	🟩		🟩		
Challenge #15 - Synchronize Data Causality	🟩			🟩	
Challenge #16 - Offline Smart Contract Operations	🟩			🟩	
Challenge #17 - Consistent End-to-End Security Provisioning	🟩				🟩
Challenge #18 - Ensure PDL Member Privacy	🟩				🟩
Challenge #19 - Monitoring Capabilities					🟩
Challenge #20 - Orchestration Capabilities					🟩

7.2 Trusted Execution Environments

7.2.1 Introduction

A prerequisite for viable operations of a PDL with offline nodes is to create Trusted Execution Environments (TEEs) which can hold the certificates, data, execution instructions and algorithmic frameworks. The TEEs ensure that data and algorithms cannot be tampered with, and thus ensure the required integrity and confidentiality. There are many ways to providing TEEs, with a non-exhaustive list provided below:

- **Hardware TEE:** The original approach to TEE is to establish a secure execution area in the main underlying processor. That ensures that code and data are protected through an isolated execution environment. Important features, such as isolated execution, integrity of applications executing with the TEE, confidentiality of data assets, and more, are therefore provided.
- **Virtualized TEE:** To aid developers and scale secure code production, an emerging trend is to virtualize the TEEs through specialized SDKs. Said approach is similar to VMs/containers with hardware pinning, and provides the best of both hardware and software worlds.
- **Local Side-Chain PDL:** Another approach proposed here is to establish a local PDL, either hardware or virtualized, and thus create a TEE through the same mechanisms as the overarching governing PDL.
- **Trusted Third Party Side-Chain:** The immutability of public blockchains or mutually independent PDLs, may yield the required level of trust.

These approaches underpin challenges #2 - #18 and are now discussed in greater detail in subsequent clauses.

7.2.2 Hardware TEE

The traditional approach in providing TEEs is in hardware. As illustrated in figure 7-1, the TEE model follows the hardware-middleware-software layering of traditional compute environments but focuses on ensuring integrity of each component and layer.

Notably, the hardware layer of the device will have to have a secured area which ringfences hardware resources, such as hardware security keys, hardware storage, hardware secure elements, etc.

The middleware, i.e. the operating system, is exclusively built from trusted components, such as a trusted core framework, trusted drivers and a trusted agent which is able to communicate with the less secure operating system.

The software layer hosts the trusted apps. They need to be security-vetted and the trusted operating system ensures that the integrity of the apps is maintained. Examples of such trusted apps are payment apps, trusted corporate applications or, indeed, PDL applications.

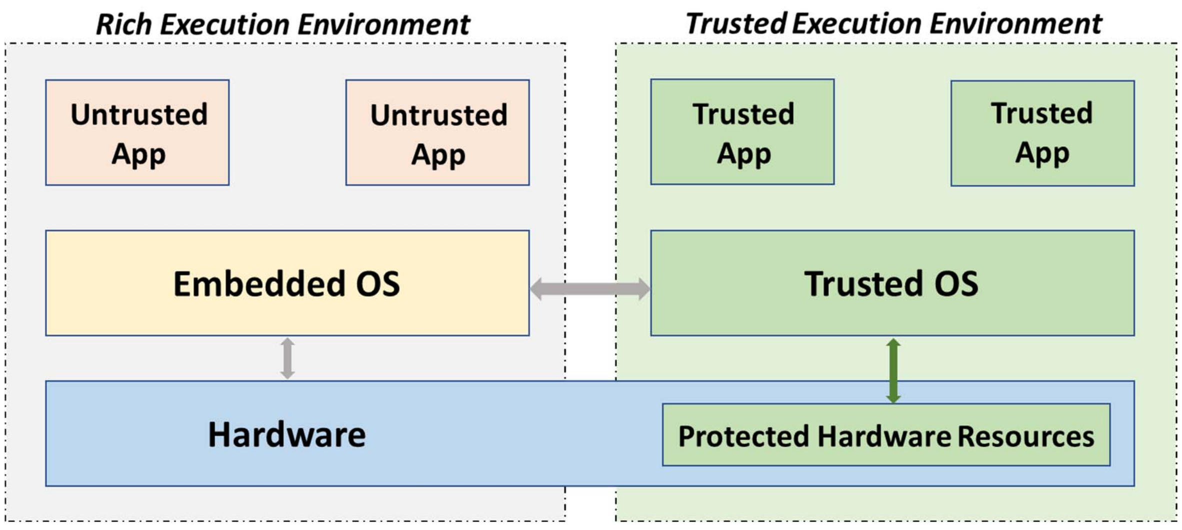
Rich Execution Environment
Trusted Execution Environment
Untrusted App
Untrusted App
Trusted App
Trusted App
Embedded OS
Trusted OS
Hardware
Protected Hardware Resources

Figure 7-1: Traditional rich execution environment versus the provisioning of a trusted execution environment by means of protected hardware resources which host e.g. root key hashes

The trusted operating system with trusted apps can be built from a trusted and protected hardware area as follows: The hardware area stores the root certificates, root hashes and root keys. These are then used to sign and cryptographically link all software elements of the operating system and applications. In fact, the cryptographic linkage is very similar to the one of blockchains.

The described hardware TEE is able to host PDL data, applications and decision logic in a secure manner which cannot be tampered with. This ensures the required level of integrity in case a node goes offline and then reappears online again.

7.2.3 Virtualized TEE

A virtualized TEE still requires a secure area within the operating environment. Typically, this is done through hardware just as with above hardware TEEs. However, the operating system and apps are implemented through a virtualization approach which is typically enabled through Software Development Kits (SDKs).

The first approach is to put a VM environment on the operating system or replace the operating system with a VM environment, with each VM having its own operating system (on "bare metal"). Then, several secure VMs can host one or several securely compartmentalized apps. This further increases the security and integrity of the system.

Another approach is containerizing the system and host secure containers with their orchestration framework on the secure operating system or the secure bare metal. It caters for lighter deployments with smaller memory and compute footprints.

An emerging hybrid hardware-software TEE worth mentioning is being developed in Cambridge, UK. The TEE is referred to as CHERI, i.e. Capability Hardware Enhanced RISC Instructions. It enables memory protection, scalable compartmentalization, fine-grained decomposition of the operating system and apps. As per [i.2], "*CHERI is a hybrid capability architecture in that it is able to blend architectural capabilities with conventional MMU-based architectures and microarchitectures, and with conventional software stacks based on virtual memory and C/C++*".

7.2.4 Local Side-Chain PDL

Leveraging on the hardware, virtualized and hybrid TEEs, another approach of providing a trusted environment is to locally deploy a side-chain of the PDL. This could be the same as the overarching system PDL or a lightweight version specifically adapted to the embedded nature of the device or set of devices it is deployed on.

In essence, the local PDL would merge the design principles of the overarching PDL; that is, the cryptographic principles of the PDL are used to verify the system components and thus provide the required level of integrity.

The specific deployment embodiment could in principle be on a single node or a distributed but connected set of nodes:

- **Single Node Deployment:** If enough independent TEEs can be provided through virtualization or containerization, the PDL could be deployed within the same node with some of the containers taking the role of Validator nodes and the remaining nodes the role of Ledger nodes. This of course requires the Client node to have sufficient compute and storage capacity.
- **Multi-Node Deployment:** Assume the scenario of an island of Client nodes going offline from the main PDL; however, the Client nodes are still connected to each other locally and each Client node is able to cater for a TEE. In that case, a local PDL can be deployed using the set of available nodes. That network, if properly deployed, provides a sufficiently trusted environment.

7.2.5 Trusted Third-Party Side-Chain

Using a third-party external ledger could improve the security of any PDL by providing additional immutable storage and execution capabilities. Two alternatives can be envisaged:

- **Trusted Third-Party PDL:** The approach here would be to provide an addition trusted storage and execution environment by using independent PDLs to which the nodes of the original PDL have access.
- **Public Blockchain:** Idem, public chains could be used which yield a high degree of persistence, in addition to immutability.

In both scenarios, security could be reinforced by providing a point of recovery for the original PDL. The basic procedure could be based on taking snapshots of the blockchain to be protected and storing it in the external blockchain, i.e. storing hashes of the snapshot. The external blockchain could also be used as a hash repository of devices/nodes that can connect to a specific PDL that can be updated by the vendor independently. This limits the breaches of poor or untrusted data sources.

In essence, reinforcing the original PDL with a combined protection of other PDLs or incorporating the strength of public blockchains is a safeguard not only against external attacks but also safeguards against internal security breaches.

7.3 Offline Operations

7.3.1 Introduction

With the TEEs in place, certificates, client data and algorithmic frameworks are stored in a trusted environment providing the required local integrity and confidentiality. That enables a viable offline operation which is discussed in this clause. Notably, technical approaches to the offline operations of Client nodes, Validator nodes and Smart Contracts are discussed below.

7.3.2 Offline Client Node Operations

To maintain the operations of a Client node which is about to or has gone offline, a set of previously discussed challenges need to be overcome:

- **Challenge #1 - Deteriorating Calibration:** As discussed above, the sensing and/or inference algorithms may need to be recalibrated. There are several ways of solving this problem:
  - *Local Re-Calibration:* If possible, enable a local environment which is able to locally recalibrate or locally learn new coefficients for the AI/ML inference engine. That should be done in the TEE, and enough memory and processing power ought to be made available locally.
  - *Timer:* If local recalibration is not possible, a timer which clocks the recalibration intervals should be used. Once the timer has expired, said inference/sensing framework should not be used anymore.
  - *Data Confidence:* Several timer epochs or other forms of more sophisticated models to estimate the fidelity of the gathered data and record that fidelity along the actual data can be used. When the node comes online again, any overarching data frameworks can take a more informed estimate about the validity of the provided data.

- **Challenge #2 - Cybersecurity Breach:** Such breach may occur when an adversary obtains access to the last mile network. Several solutions are available:
  - *"Alive-Pings":* A cyber breach is easy to detect if a keep-alive message is implemented between the Client node and, e.g. a Smart Contract on the main PDL. In absence of a request or response, each party becomes aware of the Client node being offline. If a zero-trust policy is implemented in the Client node, no network traffic is allowed except through trusted mechanisms such that the Client node can be reconnected to the main PDL once networks become available again.
  - *Network Monitoring:* A useful mechanism is to implement local network monitoring of data traffic, such as IP addresses, sessions, etc. The data could be discarded after a pre-defined time; e.g. the network data is only stored over the last 24 h. That may prove useful in the case of in situ or remote network forensics and diagnostics.
  - *Trusted Networks:* Network connections should be secure by default. Ideally, only the connection between the Client node and the PDL should be authorized. If needed, a secure network between 3rd party hosts (e.g. 3rd party database) can also be established. The usage of VPNs and infrequently used ports is recommended. The usage of the Secure Shell protocol (SSH) on common port 22 for user data or Out-Of-Band (OOB) control data should be avoided so as to evade port scanning software.
- **Challenge #3 - Physical Tampering:** Client nodes which are easily accessible in the field risk being physically tampered with. The following approaches are possible:
  - *Physical Protection:* If a prior risk assessment deems physical tampering possible, then the Client node ought to be moved or protect such that the risk of physical tampering is mitigated to acceptable levels (which may vary case by case).
  - *Physical Alert:* Embedded, and potentially hidden, sensors could be used to provide an extra layer of physical security. Notably, an embedded vibration sensor or light sensor can be added to the Client node so that when unscheduled movement has been detected (because somebody took the node off its location) or light has been detected (because somebody opened the enclosure), certain emergency protocols can be triggered, such as deletion of critical/all data and frameworks from the TEE.
  - *Downstream/Upstream Separation:* Different risk categories for upstream and downstream data can be introduced which allows different security protocols to be implemented in case of physical tampering. The former is typically data sensed/inferred by the Client node and thus anyway accessible to any adversary in the physical vicinity. The latter is control data received from the PDL which requires extra protection as it may compromise critical instructions, such as opening a critical pressure valve.
- **Challenge #4 - Access To Ledger Data:** A node may require data which is stored on the PDL and thus becomes inaccessible in the case of going offline. The following approaches are possible:
  - *Predictive Content Caching:* Advanced mechanisms can be used to predict what data will be needed by a Client node over a given autonomous operating horizon. That data is then cached into the memory of the Client node. The predictive algorithm needs to balance availability of memory, energy consumption, network usage and minimum required window of autonomous operations. For instance, if a Client node needs to be operational for 24 h, then sufficient data needs to be cached on the Client node to enable 24 h operations.
  - *Use of Edge-Cloud:* The same predictive mechanism can be used to cache data in an edge-cloud which might be far from any of the PDL nodes but close to the Client node. This will not work if the Client node itself goes offline but will be useful if the edge-cloud is isolated from the PDL due to backhaul faults. Another approach would be to replicate the entire PDL in the edge-cloud making it a de-facto Ledger node.
- **Challenge #5 - Access To 3rd Party Database Data:** Similarly, to the PDL data access, a node may need data from a 3rd party database. The following approaches are possible here:
  - *Predictive Content Caching:* The same mechanisms as for above challenge #4 can be used here where data from the 3rd party database is cached intelligently.
  - *Use of Edge-Cloud:* The same mechanisms as for above challenge #4 can be used here where data from the 3rd party database is stored in a trusted environment in an accessible edge-cloud.

  - *Direct Database Access:* Access to a 3rd party database is normally managed via the PDL. If the Client node still has networking access, then a direct and trusted peer-to-peer link can be established.

- **Challenge #6 - Read Data From Client nodes (upstream):** Smart Contract and other operations are impacted here which are discussed in clause 7.3.4 in more detail.

- **Challenge #7 - Instruct Client nodes (downstream):** The offline mode may become critical for the downstream data as it may contain control data. There are several methods to deal with that issue:

  - *Schedule or Predictive Analytics:* If control instructions can be scheduled or have a predictable pattern, then such schedule can be implemented at the Client side and operate autonomously even when OFF.

  - *Available 3rd Party Proxy:* A Client node may lose connection to the PDL but may still have networking connection which would enable the use of a Proxy node. That Proxy node is different from the PDL-proxies introduced earlier in that it may not belong to the PDL. Specific security measures thus need to be put in place to ensure end-to-end security.

- **Challenge #8 - Data Validation:** Upon rejoining the PDL, the Client node will need to transfer data to the PDL. Before doing so, data validation is required. To this end, the PDL needs to verify that a TEE was established and operational, and is trusted; furthermore, it needs to verify that all protocols and contingency plans have been followed to ensure integrity of data.

- **Challenge #9 - Data Reconciliation:** The methods proposed to ensure a proper reconciliation with the main ledger are discussed in clause 7.5.

7.3.3 Offline Validator Node Operations

To maintain the operations of a Validator node, a set of previously discussed challenges need to be overcome:

- **Challenge #10 - PDL Control Plane:** The Validator node is receiving regular control data updates pertaining, e.g. to updated security certificates or updated validation protocols. With the Validator node going offline, the following methods can be implemented to ensure continuing operability:

  - *Freeze Control Plane Updates:* Put control plane updates on a regular schedule which is known to both ledger and Validator nodes. That can be achieved with a timer and regular coarse synchronization of clocks. In that case, when an update notification is not received by the Validator node, it will stop the validation process.

  - *Pre-Emptive Update Delivery:* Another method is to send updates a certain time window before they are due to become effective. For instance, the ledger can update control information 24 h before they are becoming effective which prevents validator outages with short offline periods.

  - *Proxy node(s):* Coupled into above methods, a Proxy node can be activated when the PDL does not receive a response from the Validator node. The proxy approach is discussed in greater detail in clause 7.4.

- **Challenge #11 - 3rd Party Data Access:** Similarly to the Client node data access, a Validator node may need data from a 3rd party database which become inaccessible once the node goes offline. The following approaches are possible here:

  - *Predictive Content Caching:* As already discussed in the context of Client nodes, data from a 3rd party database can be cached intelligently in the TEE of the Validator node.

  - *Direct Database Access:* Access to a 3rd party database is normally managed via the PDL. If the Validator node still has networking access, then a direct and trusted peer-to-peer link can be established between the Validator node and said 3rd party database.

- **Challenge #12 - Proxy node:** The proxy mechanisms for the Validator node are discussed in clause 7.4.

- **Challenge #13 - Disrupting Ongoing Validation Session:** The integrity and trustworthiness of the data put onto the ledger should be of highest priority in any PDL deployment. Therefore, any validation session which is disrupted ought to be discarded and proper mechanism at the PDL governance level ought to be implemented to handle such a situation.

- **Challenge #14 - Consensus Violation:** Similarly, consensus violations ought to be handled at the governance level through proper monitoring and orchestration. An individual Validator node cannot efficiently handle such a situation.

7.3.4 Offline Smart Contract Execution

The Smart Contract resides in the PDL and is unlikely to go offline. However, client and Validator nodes may go offline and thus jeopardize the operations of the Smart Contract. The following challenges need to be addressed:

- **Challenge #15 - Synchronize Data Causality:** With client and Validator nodes going offline, the operations of Smart Contracts could jeopardize. Whilst the complexity of this problem ought to be dealt with at orchestration level, some approaches are possible to aid the Smart Contracts operations:
    - *Client node Diversity:* The most likely cause of a Smart Contract failure is due to one or more Client nodes going offline, the data of which is required for the Smart Contract logic. Mission critical contracts may thus rely on secondary and tertiary data from diverse Client nodes. For instance, rather than just deploying one IoT node in the field, two or three are deployed and connected via independent networks.
    - *Predictive Capabilities:* If predictive algorithms work within the error margins of Client node data acquisition, then this could also be used to feed client data into the Smart Contract logic. For instance, assume that an IoT node has a 20 % measurement error due to imperfect sensors; then, any predictive algorithms which are able to predict that IoT data flow within a 20 % error margin could be deployed.
- **Challenge #16 - Offline Smart Contract Operations:** The following approaches are possible here:
    - *Create Side-Chain PDL:* As already discussed in clause 7.2.4, a suitable approach here is to create an operational offline copy of the Smart Contract through a local PDL, which in essence is a side-chain and should be referred to as sPDL. That can only be done if the Smart Contract relies exclusively on data which is generated and consumed locally.
    - *Predictive Capabilities:* As already discussed above, predictive algorithms can be used to provide for the missing data within the measurement errors.

7.4 Proxy Mechanisms

7.4.1 Introduction

The availability of Validator nodes is utmost important to the well-being of the PDL. However, as discussed throughout the present document, Validator nodes may go offline. To ensure the operability of the ledger with Validator nodes going offline, the notion of Proxy nodes had been introduced. This clause describes the election and operational approach to Proxy nodes in greater detail.

7.4.2 What Is A Proxy Node

A Proxy node is in essence a dormant or idle stand-by Validator node. It is not operational, i.e. does not perform any validation, until triggered to do so. The Proxy node becomes operational when another Validator node experiences operational or networking difficulties.

A Proxy node is typically attached to one or several operational Validator nodes so that when they fail, the Proxy node can continue the required validation process.

A Proxy node should be part of the PDL, i.e. belong to the trusted infrastructure which constitutes the PDL. It would typically reside in trusted VMs belonging to the PDL consortium.

It should be noted that, whilst proxy nodes improve the viability of the PDL ledger, introducing them yields an operational overhead in terms of communication bandwidth, compute and storage overheads.

7.4.3 When To Elect A Proxy Node

A proper election of the Proxy node addresses challenge #12, and should determine the trigger upon which a Proxy node is being elected. Similar to routing protocols, the election of Proxy nodes can be reactive or proactive:

- **Reactive Establishment of a Proxy node:** A Proxy node is established once a given Validator node goes offline. The establishment of the Proxy node takes time and therefore a reactive approach yields some operational delays. The advantage however is that no operational resources are wasted in the case the set of Validator nodes works reliably.
- **Proactive Establishment of a Proxy node:** The governance framework may associate a Proxy node to one or several operational validating nodes prior to any incidence. That allows for a quicker handover in case of failure of validating nodes; however, it creates a management overhead compared to the reactive approach.
- **Hybrid Approach:** Yet another approach might be to proactively establish a pool of trusted Proxy nodes in zones of high volatility, whilst in stable zones a reactive approach is taken. These zones could be updated dynamically to ensure the most efficient trade-off between operability and overheads.

7.4.4 How To Elect Proxy Nodes

Addressing challenges #12 and #14, the actual election process would typically be administered by the governance layer. Notably, the following is important to note:

- **Ensure Validity of Consensus:** The governance methods should ensure that Proxy nodes are elected such that, once they become operational, consensus fairness cannot be jeopardized. If not done properly, a malicious member of the PDL consortium could attack network segments and operational Validator nodes such that the thus-triggered Proxy nodes give that member consensus majority.
- **1-2-1 vs 1-2-Many:** Another important decision point is if to associate a Proxy node to one operational Validator node or to a set of Validator nodes. Whilst a 1-2-Many association is more efficient in terms of control overheads, a new proxy selection may need to be triggered once a given Proxy node from the pool has become operational so as to ensure that the remaining validating nodes still have a Proxy node backup.

7.4.5 Operations of Proxy Node

Addressing challenges #12, #13 and #14, the operations of a Proxy node should address the following:

- **Cold vs Idle vs Fully-Operational Proxy node:** Depending on the operational state of the Proxy node, it could be cold, idle or fully-operational. A cold Proxy node needs to be activated and configured; an idle Proxy node needs only to be activated as it has been configured already; and a fully-operational Proxy node is both configured and ready to take over at any instance. The three modes trade managerial overhead and energy consumption with the latency experienced during handover from the main node.
- **Inform Client nodes:** Once a Proxy node or a set of Proxy nodes has been established by the PDL, the associated Client nodes should be notified, and possibly reconfigured, so they perform further communications with the Proxy node rather than attempt communicating with the defunct Validator node.
- **Synch with PDL:** The PDL will push regular control data updates to the Validator nodes. To minimize delays during validation hand-offs, the PDL should push the same control data updates to the Proxy nodes. That requires both Proxy nodes and the PDL to stay synchronized and is only applicable for the Proactive and Hybrid approaches where the Proxy node is already operational and is part of the PDL and does not require powering up and configuring.
- **Synch with Operational Validator node(s):** Idem, it is recommended that the Proxy node synchronizes with the attached operational validation nodes. That allows exchange of important data, such as the set of trusted Client nodes, timers, etc.

7.5 PDL Reconciliation

7.5.1 Introduction

In clause 7.2.4, the possibility of the establishment of an offline side-chain (sPDL) is introduced. It is, in essence, an operational copy of the main PDL and proactively/reactively helps in the event that Client nodes go offline. One of the biggest challenges is to reconcile the main PDL and the side-chain sPDL, which is the focus of this clause and is illustrated in figure 7-2.

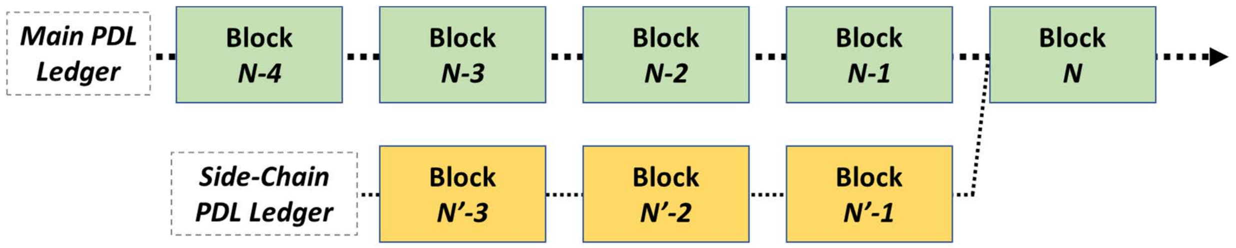

Figure 7-2: Illustration of the main PDL (green blocks) and the side-chain sPDL (yellow blocks), causally ordered in blocks

7.5.2 Ledger Reconciliation

Addressing challenges #8 and #9, the following is important to note in the context of PDL reconciliation:

- **Hindsight Consensus:** This approach would require a re-run of the consensus mechanisms in the main PDL as the side-chain sPDL comes online again. Assuming that correct timing information is recorded, the hindsight consensus mechanism should be able to order the data correctly and therefore run any cryptographic consensus mechanisms again.

  For occasional outages, that approach would work but for frequent outages this can cause an avalanche of parallel hindsight consensus runs making universal consensus impossible. This is because it would create many forks in the ledger and ultimately disrupt the deterministic execution of transactions. Hindsight consensus would also increase the computational resources (and the associated energy and time) consumed during the consensus process.

  Shown in figure 7-3, an approach to accomplish this is to have the reconciled blocks recorded as new blocks in-between the last block when both ledgers were separate and the new block where all nodes are back online.

- **Zipper Consensus:** To avoid hindsight consensus, another approach is to cryptographically link the PDL and sPDL blocks through a separate validation method. It would e.g. create a linking hash between blocks at the same or approximately same (within a pre-defined margin) timestamp. As illustrated in figure 7-4, this allows a very light yet secure validation process combining two or more chains.

There are two mechanisms which help enormously with above approaches:

- **Low-Bandwidth OOB Control:** To aid the reconciliation, an out of band control channel can be very helpful. It can transmit control data from either validation process which could be embedded into the PDL and sPDL validation process, and thus provide an extra layer of trust once both ledgers are reconciled.

- **Placeholder Blocks on Main PDL:** It may also prove useful that the main PDL reserves placeholder blocks once it detects that certain nodes have gone offline and thus are likely to run a sidechain sPDL. Such placeholders could simplify the cryptographic procedures needed for the reconciliation process.

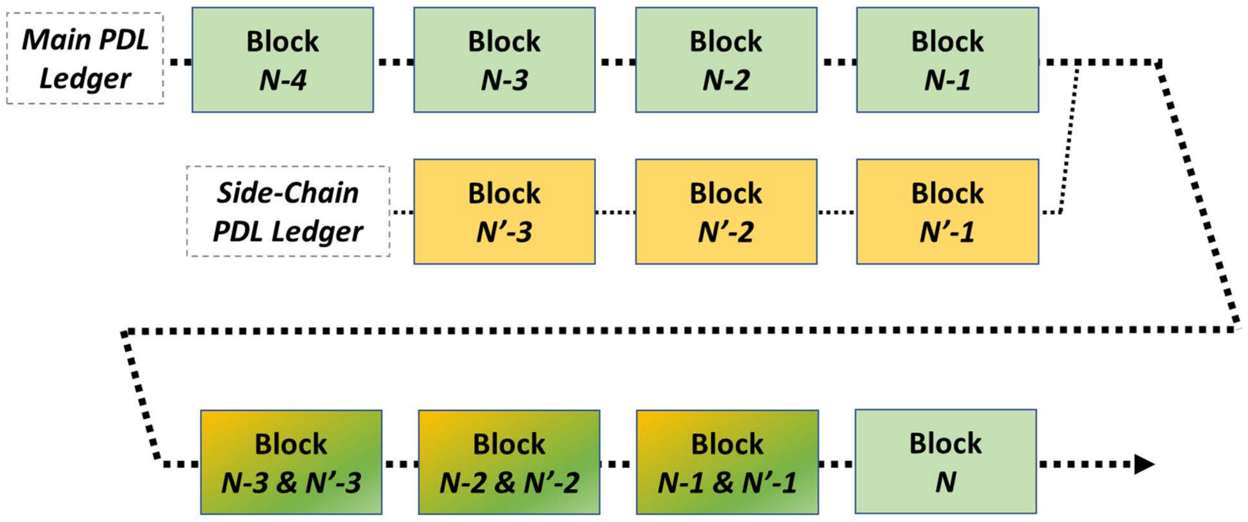

Figure 7-3: Hindsight ledger reconciliation approach where the reconciled blocks are recorded as new blocks in-between the last block when both ledgers were separate and the new block where all nodes are back online

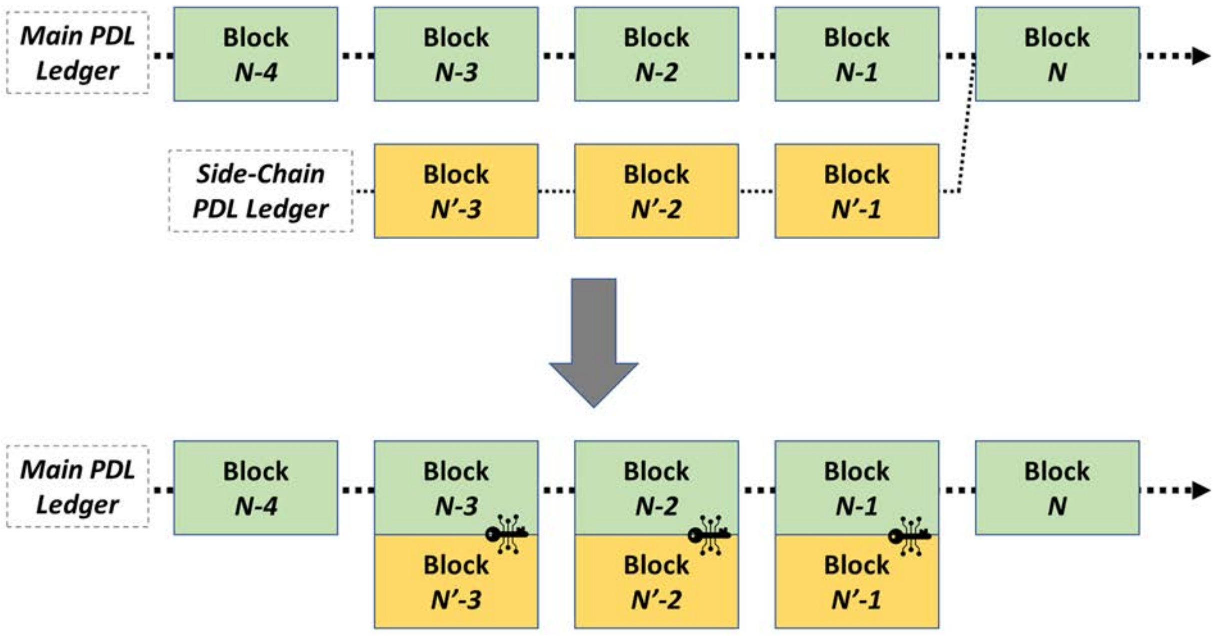

Figure 7-4: The zipper consensus approach where the content of the blocks is first time-aligned and then cryptographically linked through a separate validation procedure

7.5.3 Smart Contract Reconciliation

Once committed to the main PDL, Smart Contracts (in the form of execution logic) cannot be modified. The creation of new Smart Contracts should only be permissible on the main PDL. Therefore, even if a local copy of a Smart Contract is created as per clause 7.3.4, it cannot be modified and therefore no Smart Contract reconciliation is required.

7.6 Monitoring and Orchestration

7.6.1 Introduction

The end-to-end well-being of PDLs is enabled through proper security and privacy provisioning, which in turn is enabled through monitoring and orchestration. These issues are discussed in the subsequent clauses.

7.6.2 End-to-End Security

Securing the data and contracts, as well as links, instances and nodes of the PDL is enabled through certificates and cryptographic means.

Addressing challenge #17, a system-wide end-to-end security view is required to ensure that the PDL is truly secure over its lifetime. To this end, the following is recommended:

- **Risk Assessment:** As with any cyberphysical infrastructure, it is recommended to perform risk assessment which pertains to the infrastructure (Client/Validator/Ledger nodes), protocols and certificates, ledger and Smart Contract, and participating consortium members. Such a risk assessment allows weaknesses to be identified and mitigated.

- **Consolidating Physical and Cyber:** Whilst focus in PDL is on the protocol and Smart Contract side, one should not forget that PDLs are a vulnerable mix of physical and digital entities. A consolidated security approach should be taken when designing the PDL. For instance, it is futile to implement strong physical protection, if the security certificates are outdated or no TEE provided.

- **Zero-Trust Policy:** By default, a zero-trust policy ought to be enforced which prohibits any form of PDL operation for a node with weak, expired or absent security credentials.

- **Security Orchestration:** Since security is an important yet always-evolving element of the PDL design, a proper orchestration framework ought to be implemented which enables security with an evolving cyberphysical security landscape. The orchestration framework should ideally be placed on the ledger through Smart Contracts, but could be run from an all-trusted node in the network. It has to be understood however that such a node becomes a liability in the end-to-end security design.

7.6.3 Privacy

Between PDL consortium members, privacy of data and execution logic should be guaranteed considering that the prime reason for introducing a PDL into a consortium is the *a priori* non-trusted relationship between participating members.

Addressing challenge #18, ensuring privacy preserving mechanisms is thus important and the following mechanisms could be applied:

- **Data Encryption:** Data is encrypted at rest and in transit; and only decrypted when needed for algorithmic reasons. These algorithms should be executed in a TEE without external access. Smart Contracts could be encrypted in their entirety or parts thereof, to hide decision/execution logic.

- **Anonymization Techniques:** Advanced Anonymization techniques can be applied to the data processing, such as K-anonymity, L-diversity, T-closeness or Randomization.

- **Homomorphic Approaches:** Homomorphic data processing techniques could be applied which allow executing an algorithmic framework without decrypting the data at all.

7.6.4 Monitoring

Addressing challenge #19, a reliable monitoring of the PDL is vital. Said monitoring can be implemented in different ways, as discussed below. Furthermore, the monitored PDL node status (e.g. online or offline) can be maintained at the orchestration level and can be used to determine a Proxy node for a particular PDL node when the PDL node goes offline. Some example approaches for monitoring PDL node status are described below:

- **Peer-to-Peer Monitoring:** PDL nodes can mutually check each other's status. For instance, a PDL node A can actively send a request to its neighbouring PDL nodes (e.g. B and C) to check their status. After that, the PDL node A can report the status of itself and its neighbouring PDL nodes to the orchestration node.

- **Reactive Self-Monitoring:** A PDL node A itself may observe the gradually degraded quality of its communication links to other PDL nodes. As a result, PDL node A may actively send a status report message to the orchestration node and/or its neighbouring nodes, before it loses connectivity.

- **Proactive Monitoring:** The orchestration node may directly solicit the status report from a PDL node at given intervals, which then sends a status report message to the orchestration node.
- **Networked Monitoring:** A PDL node A may continuously and indirectly monitor the status of its neighbouring PDL nodes (e.g. B and C), for example, by measuring the number of transactions from neighbouring PDL nodes, or by measuring the Round Trip Time (RTT) between itself and neighbouring nodes. As an example, if the PDL node A suddenly stops receiving transaction from PDL node B but there are still some transactions coming from PDL node C, PDL node A can assume PDL node B's current status as offline. Then, PDL node A can report PDL node B's status (or assumed status) to the orchestration node and/or other PDL nodes that may, on their end, perform further tests or reconcile information from additional sources to determine the status of PDL node B.

7.6.5 Orchestration

Addressing challenge #20, orchestration capabilities are required to ensure proper function of the PDL in offline mode. The exact functionality of each orchestration entity is not specified here; however, important operating guidelines are outlined which underpin the difference between a single trusted entity (where an orchestrator, such as ETSI MANO, would typically reside) versus PDL being a distributed consortium of non-trusted entities:

- **Single Point of Failure:** The orchestration entity, which could be a virtual machine or a physical node, should not become a single point of failure. Whilst one can imagine running an orchestrator from a single entity, backup entities need to be always available. The ownership structure of the running orchestration entity and the backup entities should reflect the ownership structure of the PDL or be implemented through trusted third-party escrows. Situations where a minority of PDL consortium members could bring down the orchestration entity should be avoided.
- **Orchestration Decision Logic:** Similar to the hosting of the orchestration entity, the logic behind orchestration should also not depend on a minority of PDL consortium members. Non-distributed but trusted decision instances are possible if trust is established through offline contracts; this, however, is not ideal since it counteracts the very nature of PDL.
- **Distributed Orchestration Operation:** Ideally, the orchestration framework should be implemented through Smart Contracts residing on the same PDL or an associated orchestration PDL. That ensures that the security, privacy, and transparency gains made with the introduction of the PDL are not lost with a poorly designed orchestration framework.

8 Offline PDL Architecture and Procedures

8.1 Introduction

This clause introduces possible architecture embodiments containing the technical solutions discussed in the previous clause. Associated operational procedures are also outlined and discussed.

8.2 PDL Architecture Embodiments

8.2.1 Logical Architecture Elements

The canonical elements of a PDL are discussed in more detail in this clause.

The Client node, depicted in figure 8-1, has internal capabilities (TEE, storage, algorithms, etc.), external interfaces (sensory data, control data, wired/wireless networking, etc.) as well as monitoring and orchestration interfaces. The Validator node, depicted in figure 8-2, has similar interfaces as the Client node with the main difference that it only interfaces with Client nodes, Ledger nodes and peer Validator nodes, and lacks the sensory and control data interfaces. The Ledger node, depicted in figure 8-3, has the same interfaces as the Validator node with the exception that it lacks an interface with Client nodes.

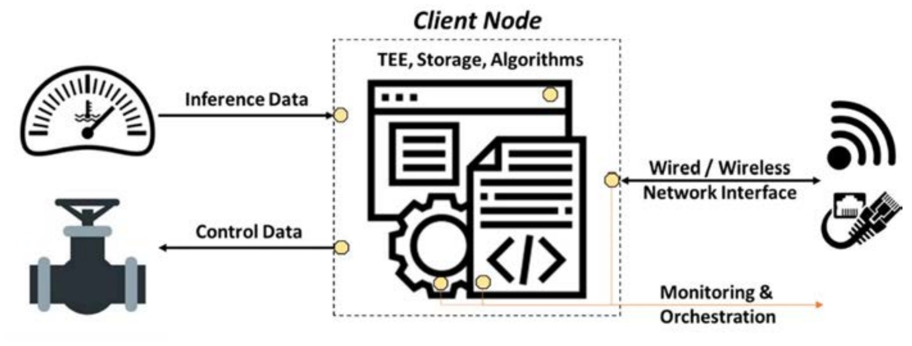

Figure 8-1: Client node with embedded TEE, secure storage and the required algorithmic frameworks which ingest inferred data and/or send control commands

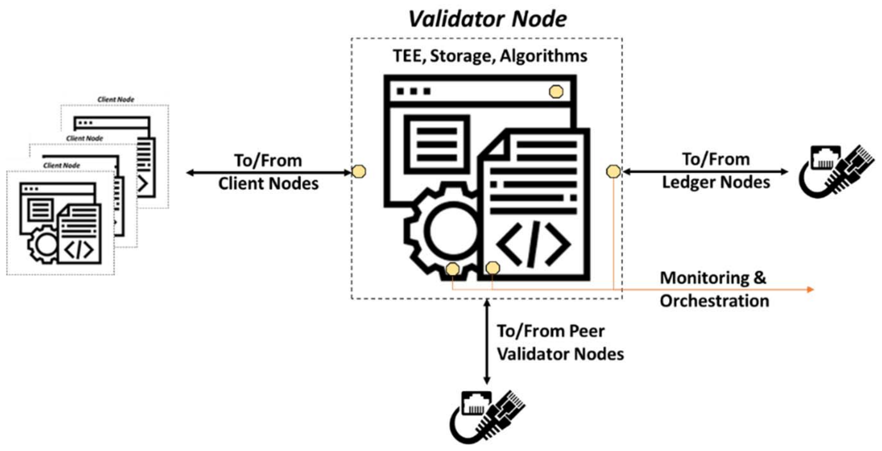

Figure 8-2: Validator node with embedded TEE, secure storage and the required algorithmic frameworks to handle the validation process

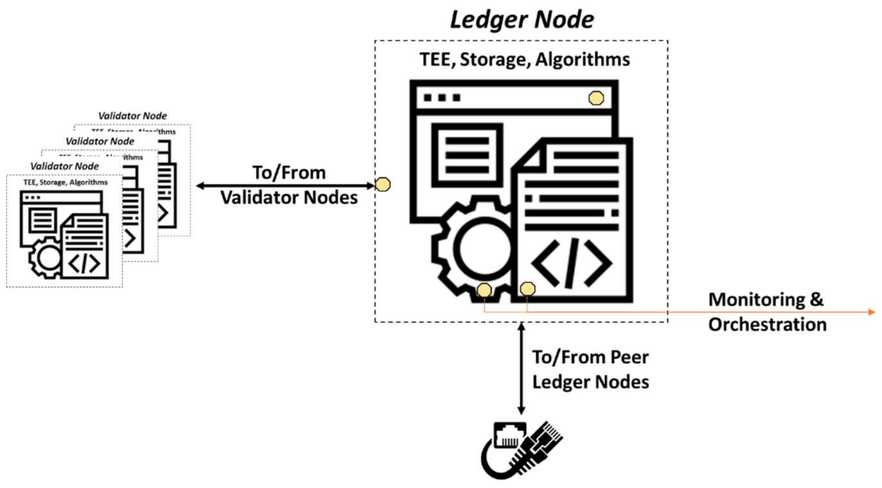

Figure 8-3: Ledger node with embedded TEE, secure storage and the required algorithmic frameworks which enable the operations of the PDL

8.2.2 High-Level Architecture

A high-level PDL architecture with offline capabilities is shown in figure 8-4, with more details provided below.

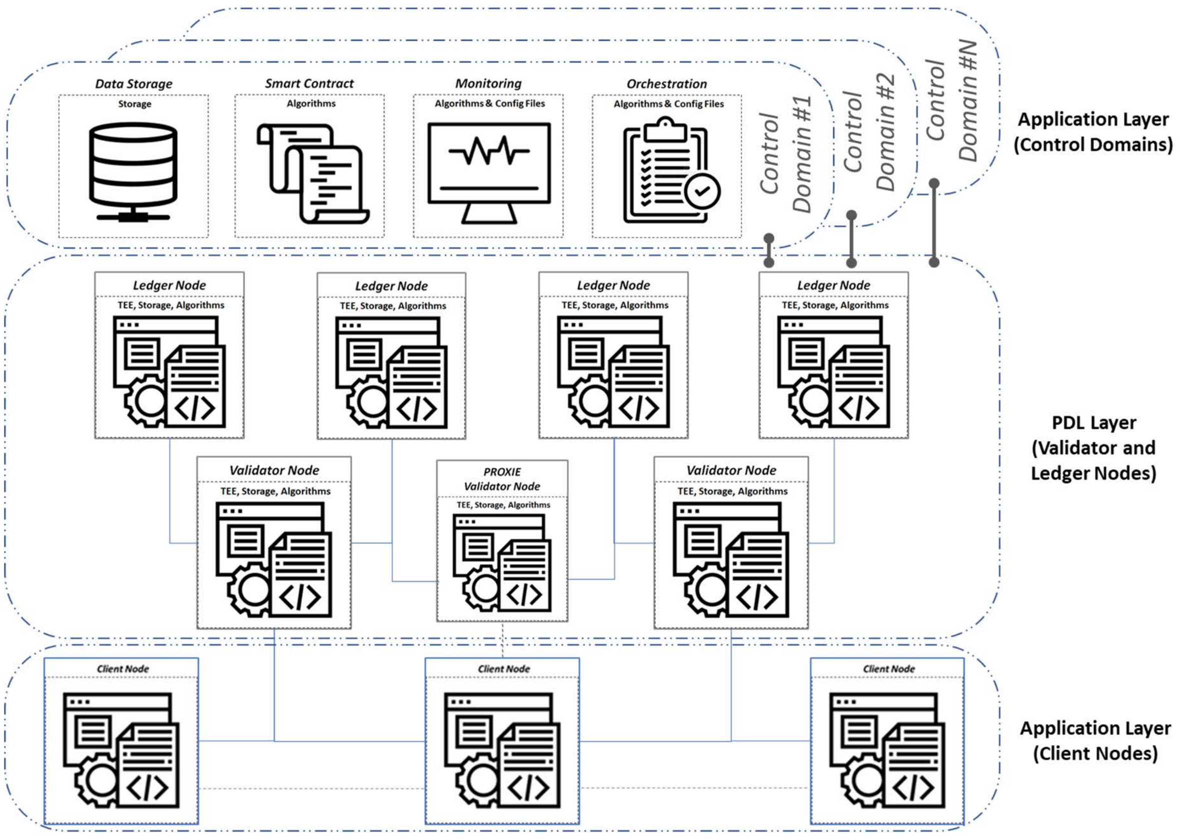

Figure 8-4: High-level architecture comprised of Client nodes, Validator nodes and their proxies as well as Ledger nodes, and several control domains

Notably, the Client nodes are connected to the Validator nodes. There are also Proxy nodes which the Client nodes are connected to, even though the connection is only activated when the associated Validator node goes offline.

The Validator nodes are connected to the Ledger nodes. These nodes store the data and host as well as execute the Smart Contracts. The Ledger nodes may also host monitoring as well as orchestration capabilities.

Note that there are different control domains for certain components, controlled by a different set of PDL participants.

8.2.3 3GPP-Aligned Architecture

A PDL Function (PDLF) can be created in the Control Plane (CP) of 3GPP systems, containing the operational elements shown in figure 8-4.

The advantage of hosting the PDLF as part of the 3GPP stack is that many issues related to security and authentication are easily provided within the secure envelope of 3GPP systems.

A new network function interface would need to be created too, referred to here as *Npdlf*. It would use HTTP or JSON to communicate with the CP message bus via the Service-Based Interface (SBI) as defined in 3GPP.

This has been illustrated in figure 8-5. Note that PDLF would only be the control function. The physical elements of the PDL could be distributed in the mobile telecommunications system.

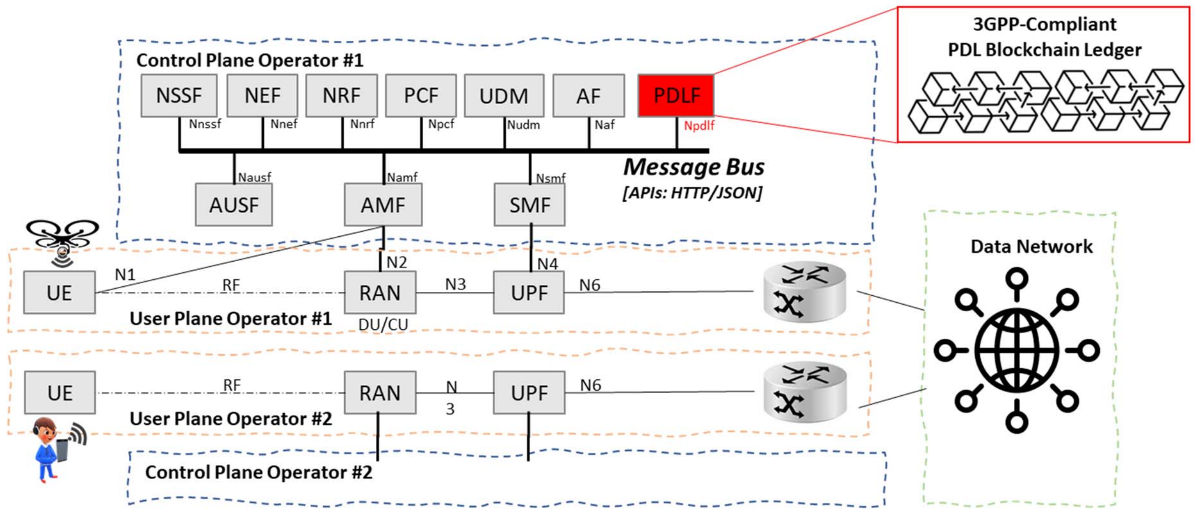

Figure 8-5: 3GPP architecture and the new network function PDLF

8.3 MANO-Aligned Orchestration Framework

ETSI MANO has been introduced as a standardized orchestration framework in networking and telecommunications systems (see ETSI GS NFV-MAN 001 [i.3]).

The orchestration of all functionalities not related and not impacting the PDL can be done in a traditional centralized way as there is no apparent benefit to distribution in a MANO framework from a centralized orchestration perspective.

However, orchestration of the PDL should follow stringent design guidelines in order not to break the paradigm of distributed ledgers. The implication is that an ETSI-MANO orchestration, when applied to a PDL, may remain centralized in context but implemented in a distributed manner.

As shown in figure 8-6, it is suggested to host the orchestration capabilities on the same or a separate PDL. If that is not possible, then a trusted escrow is also an option as it ensures the impartial execution of the orchestration tasks. The standardized management and orchestration ETSI MANO framework [i.3] can be adapted to PDL by ensuring that the orchestrator is implemented through a PDL (option 1) or escrow (option 2).

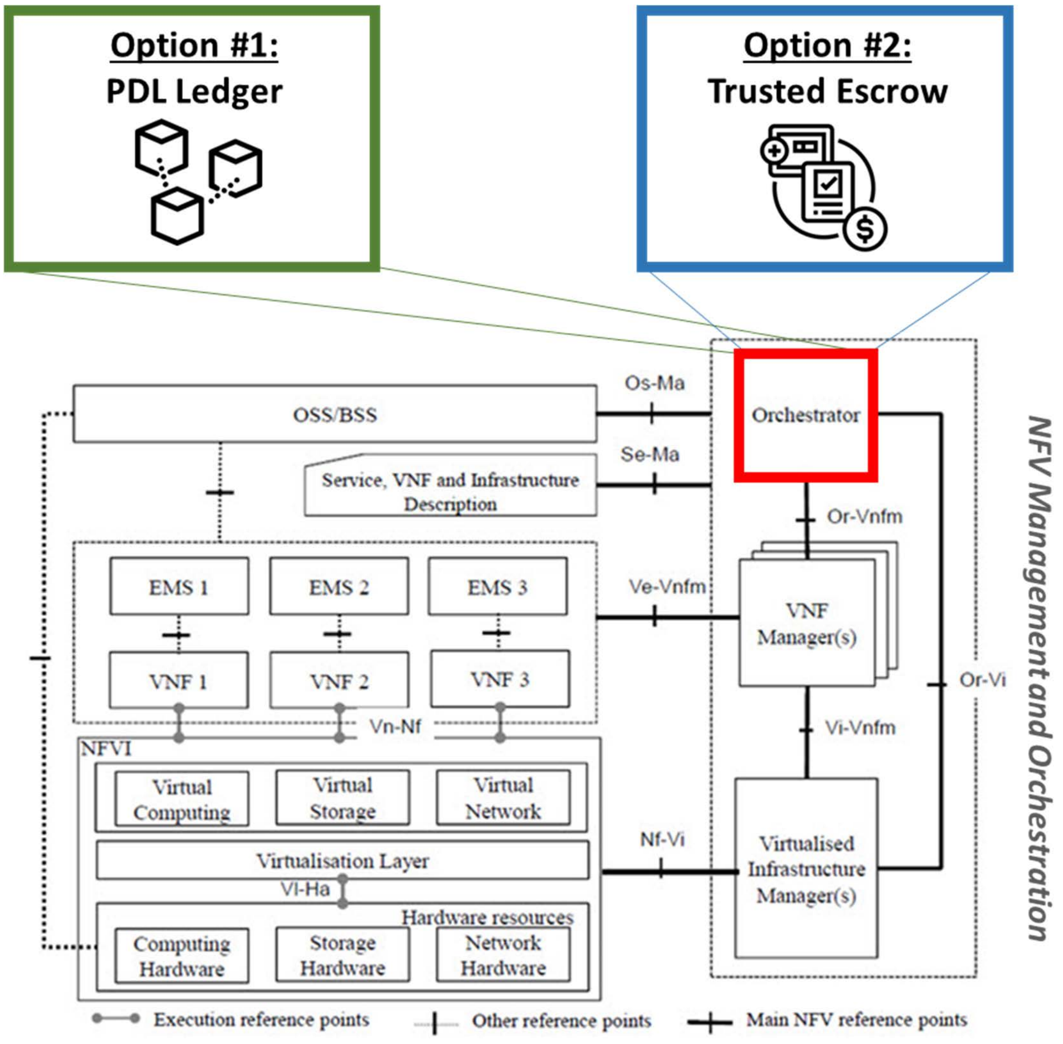

Figure 8-6: ETSI MANO framework orchestrator implemented through a PDL (option 1) or escrow (option 2)

8.4 Deployment and Operational Procedures

8.4.1 Introduction

This clause introduces deployment and operational procedures related to the technical solutions and architectures introduced in previous clauses. Notably, the procedures related to the Client node preparations as well as the offline operations of client and Validator nodes are discussed herewith.

8.4.2 Offline Client Node Preparations

Figure 8-7 illustrates the procedure for preparing the Client node for a potential offline operation.

First, the Client node is switched on. The Client node then commences the PDL discovery process using static or dynamic discovery mechanisms.

Once discovered, it connects to the ledger. The ledger validates the TEE and authenticity of the Client node. Once verified, it requests orchestration information which, together with the operational information, is passed on to the Client node.

The Client node verifies the received information and provisions accordingly. Said information will include details on how to operate under the different offline scenarios discussed above. It will also include information about Validator nodes and their Proxies.

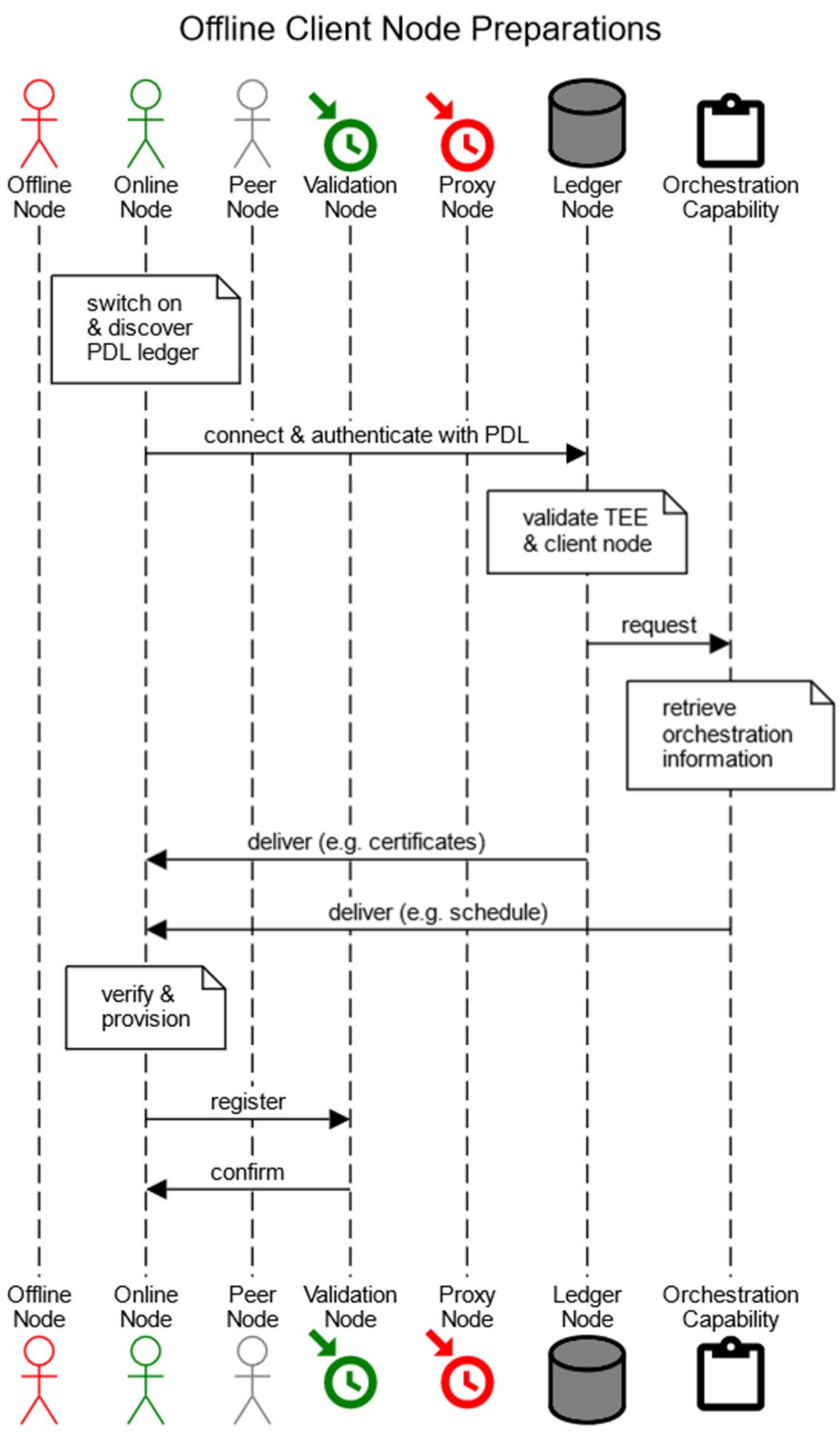
Offline Client Node Preparations
Offline Node
Online Node
Peer Node
Validation Node
Proxy Node
Ledger Node
Orchestration Capability
switch on & discover PDL ledger
connect & authenticate with PDL
validate TEE & client node
request
retrieve orchestration information
deliver (e.g. certificates)
deliver (e.g. schedule)
verify & provision
register
confirm

Figure 8-7: Procedure to prepare a Client node for a possible offline mode

8.4.3 Offline Client Node Operations

Figure 8-8 illustrates the procedures occurring when a Client node is rendered offline.

Notably, upon going offline, the connection between the Client and Validator nodes is lost both ways.

The Validator node then registers the node as offline and informs the Ledger nodes as well as the monitoring and orchestration framework.

The Client node, in the meantime, enacts the offline operational protocol which was provisioned as per clause 8.4.2 and discussed in the present document. If no such offline operational protocol had been defined or provisioned, the Client node will stop data collection from the sensory interface and distribution of control data through the control interface, and will wait for the connection with the Validator/Proxy node to resume operations.

The first approach is for the Client node to establish a networking connection with peer Client nodes. If successful, it could establish an operational side-chain ledger sPDL.

A second approach, if no peers are discoverable, is to handle the data and Smart Contract operations locally using the prior-provisioned TEE.

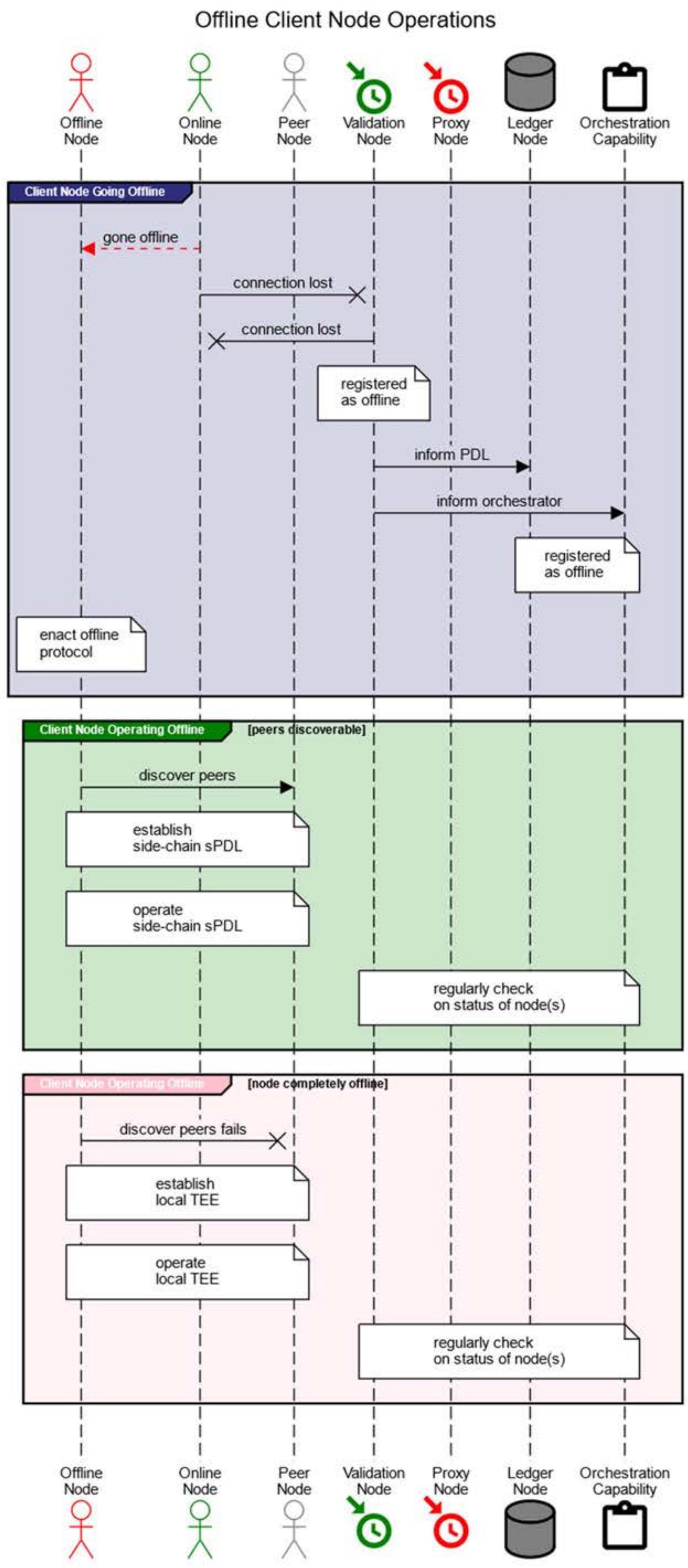

Figure 8-8: Procedure in case a Client node is rendered offline

8.4.4 Offline Validation Node Operations

Figure 8-9 shows the procedures occurring when a Validator node is rendered offline.

In that case, the connection from both the Client node(s) as well as Ledger node(s) is lost.

The Proxy node is then replacing the Validator node which has gone offline. The choice of the Proxy node will have been communicated via the orchestrator.

If no Proxy node is available (either by design or due to absence of sufficient resources), then the validation procedures should be stopped or paused until a Proxy node emerges.

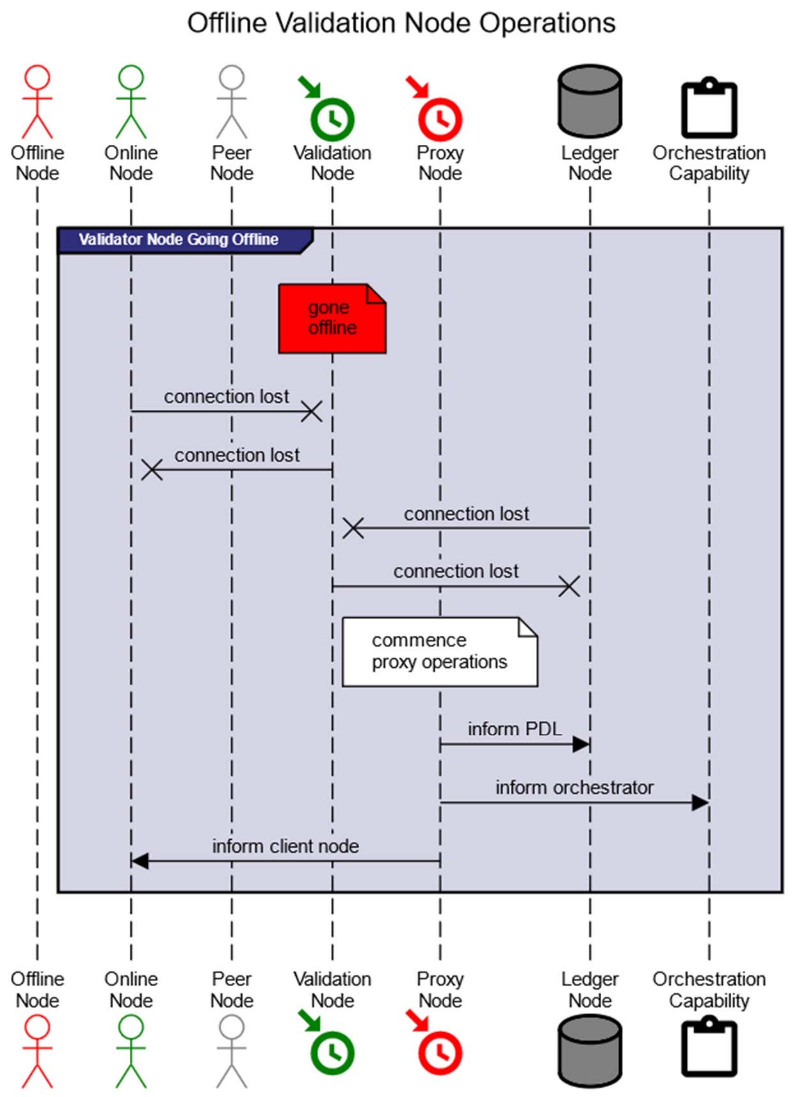

Figure 8-9: Procedure in case a Validator node is rendered offline

9 Conclusions and Next Steps

9.1 Concluding Remarks

Distributed ledgers have proven to be an efficient technology to establish trusted operations among non-trusted parties. Whilst protocols and architectures typically assume a constant availability of all nodes involved, in the case of PDL, nodes forming the ledger could go offline. Such events could have enormous repercussions for the viability of the PDL.

The issues arising in the context of offline operations are covered in the present document. Notably, the reasons due to which nodes may go offline are discussed. This underpins a discussion on the operational issues arising from nodes going offline.

The present document then offers a selection of technical solutions which would address the identified challenges. This is complemented with suggestions of architecture embodiments and operational procedures.

9.2 Next Steps

The following is suggested in terms of next steps:

- Develop specifications of the offline preparation of the Ledger nodes.
- Develop specifications of the offline operations of the PDL.
- Develop specifications for reconciliation of data upon return from offline mode.
- Develop specifications of the orchestration capabilities of the PDL under offline operations.
- Develop a viable architecture encompassing above specifications demonstrated through a PoC proving operational viability of the designed system.

History

Document history		
V1.1.1	August 2021	Publication