PONTON
WE ARE THE 2 IN B2B

WRMHL

Whitepaper

Version: 1.1
Date: 28th October 2019

PONTON GmbH
Dorotheenstraße 64
22301 Hamburg, GERMANY
Email: wrmhl@ponton.de
http://www.ponton.de

Table of Contents

1.Why WRMHL?

WRMHL (“Wormhole”) is an enterprise-ready, low-latency blockchain framework for industry consortia or communities. WRMHL is inspired by using a wormhole to slip into the future of blockchain technology, skipping the Gartner’s Hype Cycle[1] trough of disillusionment.

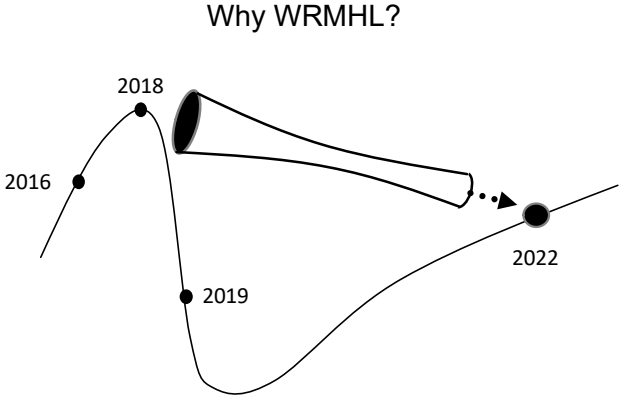

Figure 1: WRMHL gives us a lift to the future of using blockchain for B2B integration

Throughout all industry sectors you find thousands of business-to-business processes where participating organisations require an affordable high-speed tool for the synchronisation of data and process statuses across organisational boundaries. PONTON has developed the WRMHL framework based on user requirements from industry participants in order to provide a high-quality software layer which meets these requirements.

Data consistency across a larger number of organisations is one of the main features of blockchain technology. As with Bitcoin, Ethereum, or other, faster blockchain technologies, WRMHL assures a consistent data state across all nodes and all participants connected to these nodes. However, in contrast to other blockchain technologies, WRMHL goes beyond the basics by providing additional features such as:

- Fast data synchronisation within less than a second,
- Flexible adaptation to use case requirements,
- Application development in Java or other industry-grade languages,
- State-of-the art maintainability of on-chain logic (process/ application specific extensions),
- Modularised and extensible architecture design,
- Multitude of deployment scenario’s which can be easily be adopted,
- Ability to store arbitrary data e.g. representing digital records or assets.

These features are presented in the following sections of this Whitepaper. However, before starting with WRMHL, we would like to present a few typical industry processes and analyse when implementing them on blockchain technology makes sense, and when not. PONTON’s expertise is based on 18 years of experience in B2B integration which has strongly influenced the WRMHL architecture and process design.

[1] https://en.wikipedia.org/wiki/Hype_cycle

1.1. Why use blockchain for B2B integration?

Often, designers of B2B integration infrastructures have blockchain in mind – but in reality, blockchain is only in certain cases a true alternative to the “classical ways of integration” via central platforms or bilateral data exchange (see also [Merz19]). In most cases, one of the classical solutions will serve as a sufficient and efficient way to coordinate distributed processes:

- think of exchanges, Uber, AirBnB, Sabre, and similar platforms as centralised solutions or
- think of bilateral data exchange between businesses as practised alongside supply chains of all different sectors for decades.

Blockchain comes into play if a central platform appears as too expensive or not trustworthy and bilateral data exchange does not fit the business process requirements: high-cost and low trust are the drivers leading towards blockchain based solutions. But this alone is not sufficient. The process itself needs to fit the blockchain pattern: in the best case, it requires data to be shared among all participants.

Think of a Bitcoin transaction: the transaction is written into a block which is visible to all participants and beyond. Even non-users of Bitcoin may inspect the transaction using a block explorer. On the contrary, if transaction data is always encrypted only for one single receiver, we will fall back to the bilateral data communication pattern. There, we find enough messaging protocols that allow for end-to-end security and non-repudiation of submission and reception. With efficiency and scaling not being a primary focus of blockchain technology, it might be not a suitable solution for scenarios with large volumes of transactions. Also, don’t consider a blockchain solution if the business value of the transaction is extremely low or as a carrier of, e.g., IoT transactions. In the latter case, participants will end up with gazillions of megabytes of redundant data that need to be managed decentrally and that cannot be processed efficiently by (big data) applications.

All-in-all, using blockchain in B2B consortia makes sense in particular if:

- most of the data communication follows the broadcast pattern of 1:N communication,
- most of the transaction data is transparently shared with other participants,
- transactions carry a certain minimal value and occur at a moderate rate,
- process participants perceive a centralised solution as less efficient, too costly, or the central operator as not trustworthy,
- highest level of availability, ordering and immutability of the messages are important.

1.2. Unique value proposition of WRMHL

The unique value proposition of WRMHL can be summarised as follows: WRMHL makes it easy for users to build end-to-end business application utilising blockchain technology. WRMHL is very well suited to support decentralised processes and secure 1:N communication. It is scalable as well as fast and provides resilience and security at low cost, while there is no single point of failure. It allows clients to bring their use case to market much faster and at a fraction of the cost compared to developing without a blockchain framework. Since WRHML is not limited to specific features of the carrier blockchain, WRMHL is process independent and supports all kinds of different use cases for various

industries. Therefore, there are also no process limitations for the business application developed on WRMHL. In comparison to public blockchains no transaction fees apply and members have full control over governance.

1.3. Case studies

Why did PONTON develop WRMHL? When PONTON analysed a range of typical processes that seemed suitable for a blockchain solution and understood the exact requirements, PONTON realised the need for a flexible blockchain infrastructure for B2B processes and tailored WRMHL around those requirements. Below are three case studies for use cases that are particularly well suited for WRMHL:

Use case 1: Car registry

The administration of vehicle ownership involves multiple administrative processes across many stakeholders from various sectors. Many of those processes are still paper based and data is not shared between shareholders. This poses a risk of information manipulation, data duplication, synchronization issues, prolonging the processes more than necessary and is prone to error.

A WRMHL based blockchain solution allows each stakeholder to add their data as an immutable record to a common log while having access to data entered by other stakeholders. Below are process examples of various stakeholders in the ecosystem:

- Manufacturer: e.g. registration of new car, warranties, spare parts
- Car owner: change in personal data, transfer of ownership
- Registration authority: e.g. Issue and transfer of car registration certificate I and II
- Insurance: e.g. proof third-party liability insurance cover, claims procedure
- Banks: e.g. car certificate as loan or lease collateral
- Garages: e.g. service history, damages reports
- General inspectors (TÜV): Issue and renewal of road-worthiness certificate
- Authorities: e.g. payment of road tax
- Other private persons: e.g. private car hire
- Police: Flag vehicle as stolen

Bringing those processes to a WRMHL based business application enables a decentralised system which provides trust and enables cross sector collaboration of the entire ecosystem of stakeholders, promising improved data integrity and reduced risk of fraud.

A WRMHL based vehicle registry system would also make the entire process more efficient as well as cost effective and convenient. PONTON did develop a prototype based on WRMHL for vehicle data management over the blockchain as part of the technical documentation.

Use Case 2: Land registries

Land registries keep records about the ownership of property. Any record amendment refers to the change of ownership or the change of ownership details. E.g., a sales contract or the registration / deregistration of a mortgage represent such record amendment. Land registries (at least in some countries) are operated by local courts in a federated form. There is no

central “master registry”. I.e., introducing a master registry would contradict the federated structure and would introduce additional cost.

Any record amendment takes place by one of the federated registries and needs to be replicated to the other registries (among others, for backup reasons). It is further required to keep data in sync across the participants. Process participants need to be managed, i.e., registered and authenticated. Access is exclusively available to registered participants, while no external player shall be able to decipher the exchanged data, be able to attack the overall system or manipulate any data or code.

By using blockchain, this is avoided by the redundant distribution of data across nodes and by using encryption and authentication for every data link between nodes as well as between nodes and the software applications on top of them.

Use Case 3: Distributed trading

All kinds of products are traded using centralised marketplaces. However, there are cases for certain products or industries where no trust in the platform operator is given or where the third-party cost is viewed as too high.

While blockchain technology in principle allows to decentralise the execution of trades, a blockchain based application must fulfil the same requirements which apply to centralised marketplaces: transactions need to be processed at near-real-time, participants need to act anonymously and, at the same time, orders and trade data needs to be shared with all participants without delays. In addition, a blockchain solution makes only sense when the block time is required to be as close to zero as possible. Many other requirements, not listed here, need to be met as well.

With Enerchain[2] a WRMHL based communication infrastructure for trading wholesale energy products such as power and gas, PONTON has achieved to meet all requirements. Using the PoA (Proof-of-Authority) of Tendermint[3], consensus can be reached in just 200ms within a group of Validator Nodes distributed across the Internet. Participants of Enerchain are authenticated and identified as consortium members by the blockchain infrastructure and the overall system is secure with regards to tamper resistance, data leakage and cyberattacks. Also, historic market data in the blockchain was made available for later analysis enabling market participants to synchronise with the “data truth” if their respective trade data gets out of sync. Furthermore, WRMHL provides anonymity when it is needed, allowing for data encryption which makes it impossible to identify participants by their trading pattern after some time.

PONTON is currently developing various further distributed trading solutions[4] based on WRMHL together with industry partners.

Further down it is explained how the architecture and features of WRMHL address the requirements of blockchain use cases.

[2] More information on Enerchain can be found here: www.enerchain.ponton.de
[3] See www.tendermint.com/
[4] E.g. www.etiblogg.com

2. WRMHL overview

WRMHL is a blockchain based software framework that is used for the development of distributed applications. Primary WRMHL users are software developers working for industry consortia or system integrators. Indirect users are members of B2B consortia or communities who intend to simplify data exchange within their group.

WRMHL is a use-case independent framework that can be tailored to a given target business application solution design. WRMHL leaves a range of design decisions to the application layer so that developers are flexible in their decision which function should be used from the WRMHL framework and which to implement as part of the business application.

WRMHL builds upon experience with several distributed blockchain applications that PONTON has developed over the past years and that require an efficient and robust technical foundation.

WRMHL features:

- **Message-driven**: WRMHL data communication is message driven, i.e., participants send and receive messages.
- **Message API**: The WRMHL Message API is generic in the sense that any message exchange is tunnelled through the same interface. This simplifies application development so that existing applications can easily be re-used for the next project. The API documentation is available to software developers.
- **Programming API**: The WRMHL Programming API simplifies application development. The API documentation is available to software developers.
- **Extremely short block time**: For projects like Enerchain, PONTON uses WRMHL with a block time of just one second. This was measured using 10 validator nodes in a multi-cloud environment. The block time could also be configured for any longer duration if the application allows.
- **Extremely low end-to-end latency**: A message that is sent by an application through all WRMHL components, with several cryptographic steps will reach the other participants on average in less than one second. If blocks are formed at a rate of one second, the worst-case latency is here one second for the block time plus the consensus time, which is around 200ms, depending on the number of validators. The best case is just the consensus delay if a transaction is processed right before the formation of a block. On average, latency is thus around 800ms.
- **Security**: All network communication between the components of the WRMHL architecture is encrypted and authenticated. Application developers may add further cryptographic functions as it may be required by the business process. They are not restricted by WRMHL and may apply any type of end-to-end private / public key usage, group keys, ring signatures, etc.
- **Easy to maintain**: WRMHL is not using smart contracts. Instead, process/ application specific extensions are deployed on the blockchain and on the user side. As WRMHL is developed in Java, and can be extended in JAVA, it has all the benefits of an established software development language.
- **Clients (Client Applications) instead of Wallets**: Wallets, which are used by other blockchain technologies, are limited to certain use cases in particular for

cryptocurrencies. In contrast WRHML does not enforce any wallet paradigm and uses Clients/ Client Applications as an endpoint to access the blockchain, which allows for arbitrary business logic based on the WRMHL messaging concept.

- **Built-in exchange of certificates**: Public key certificates are required to authenticate participants and to grant access to the blockchain infrastructure. The blockchain itself is used to transfer and store certificates. I.e., the process of certificate publication itself suits very well to the blockchain. With WRMHL, application developers can easily design a separate PKI layer (public key infrastructure).
- **Support true end-to-end encryption**: Nodes may be hosted by other participants. I.e., any data sent through the blockchain and stored there may be accessible by those participants who operate a node. If end-to-end encryption is used, even these node operators are not able to access any data that has been encrypted.
- **User roles**: Application designers may require separate user roles with different access rights.
- **Ready for multi-cloud environments**: WRMHL was tested across several cloud infrastructures. It is important to avoid cloud vendor lock-in, so it was a design goal of WRMHL to allow participants to use the cloud environment of their choice.
- **Various operating options**: Thanks to the modularisation of WRMHL, components can be operated in different ways: Either the whole stack (blockchain plus application) may run on premise at each of a participant or the Blockchain Nodes may be operated by a third party (other participant or service provider) or all components may be outsourced. This way, WRMHL adapts to the individual preferences of each participant or the consortia.
- **Standard deployment mechanism**: WRMHL can be deployed by participants using standard technologies such as Ansible or Vagrant for example (find further details in the WRMHL documentation).
- **Based on Tendermint consensus mechanism**: Tendermint is a well-known implementation of the highly efficient PBFT (Practically Byzantine Fault-Tolerant) consensus mechanism, allowing for a high number of transactions per second with a low end-to-end latency.
- **Blockchain abstraction layer**: The interfaces between components of the different WRMHL layers abstract away from Tendermint-specific functions. This supports the migration to a different consensus mechanism if this should be more applicable to the business process requirements in the future.
- **Application library**: WRMHL allows business process specific extensions, e.g., for distributed trading. In case of a larger number of processes sharing the same logic, frameworks such as WRMHL reduce work and can easily be adopted by application developers.
- **Industry-proof**: WRMHL was developed based on the same software quality requirements that also apply for 24/7 conventional technology applications that PONTON is developing for over 18 years. All parts of the WRMHL software are tested for quality, performance, and against vulnerabilities / abuse patterns. In addition, WRMHL has been tested by a large number of industrial users in projects like Enerchain and NEW 4.0[5].

[5] www.new4-0.de, English flyer: http://www.new4-0.de/?wpdmdl=816

- **Excellent support**: PONTON is an expert in B2B integration since the year 2001. PONTON has developed central platforms, and has standardised and implemented bilateral data exchange infrastructures. PONTON has developed and distributed blockchain-based applications since 2016 and supports users and solution partners. As (a) maintainer PONTON takes care of improving and further developing WRMHL.
- **Cost reduction**: Blockchain technology helps to avoid intermediaries, so that just the infrastructure cost remains, which only consists of the hosting cost plus maintenance and support of the Blockchain Nodes. Usually this cost is a fraction of the process co-ordination cost compared to a central platform.

WRMHL is a development framework for distributed software applications. I.e.,

- **WRMHL is not a cryptocurrency.** Even tokens are not required within WRMHL.
- **WRMHL does not use smart contracts.** However, the process/ application specific extensions can be integrated in the blockchain consensus and provide the same functionality of a smart contract.
- With WRMHL, software development for process/ application specific extensions is a lot easier and not so limited regarding freedom of implementation compared to smart contracts:
    - **No constraints on data or logic complexity**: There is no limitation for the complexity of the data structure or for the integration of further software applications (databases, back-end systems), if the application logic requires it.
    - **No transaction fee:** Decentralised applications are designed to the benefit of all participants. I.e., there is no need to pay for the execution of smart contract transactions.
    - **Simplified software updates:** In case it is required to update functions and modules, the WRMHL blockchain allows (creates) the possibility of taking a node offline and replacing the application logic or the node itself. After the node which was disconnected from the network has joined the remaining nodes again, missed blocks and transactions are re-loaded.

3. WRMHL architecture

3.1. Terminology and overview

- **Blockchain Node:** the core blockchain logic as it is used from a blockchain vendor or developer such as Tendermint.
- **Client Adapter:** horizontal logic that accesses a Node Adapter and that resides within a participant.
- **Client Adapter Extension:** extension/ addon of the Client Adapter to implement application-specific logic.
- **Client/ Client Application:** application software using the WRMHL framework. In the case of distributed trading, Clients/ Client Applications are, e.g., the trading GUI front-end, automated trading logic ("algos"), or systems for post-deal processing.
- **Node Adapter:** a horizontal layer which is application agnostic and adds functions to the node. The Node Adapter is collocated with the Blockchain Node.
- **Node Adapter Extension:** extension/ addon of the Node Adapter to implement application-specific logic.
- **Non-Validator Node:** A Blockchain Node participating in the Tendermint consensus mechanism.
- **Validator Node:** A Blockchain Node not participating in the Tendermint consensus mechanism.

The WRMHL architecture is a B2B integration framework that allows business communities to set up a common layer for data communication and data coordination between participants.

As a middleware for decentralised processes, WRMHL foresees the following layers between the Blockchain Nodes and the Client/ Client Applications:

- The **Node Adapter** is co-located with each Blockchain Node. It is assumed that Node Adapters reside in a location that is public within the sphere of the consortium, i.e., all participants need to have access to any Node Adapter. For this reason, no private secrets of individual participants shall be disclosed in the sphere of a Node Adapter. If there is a need for sharing secrets, this can be implemented at the level of the Client Adapter or the Client. Typical Node Adapter functions are:
    - delegate messages for validation to the Node Adapter Extensions,
    - send and receive transactions and blocks to and from the node,
    - authenticate Client Adapters,
    - cache blockchain content,
    - manage application-specific Node Adapter Extensions,
    - reconnect and re-sync with a node in case of a fault,
    - mutually sense a heartbeat signal between the Blockchain Node and connected components (real-time health check) which can be used to switch in case of failures.

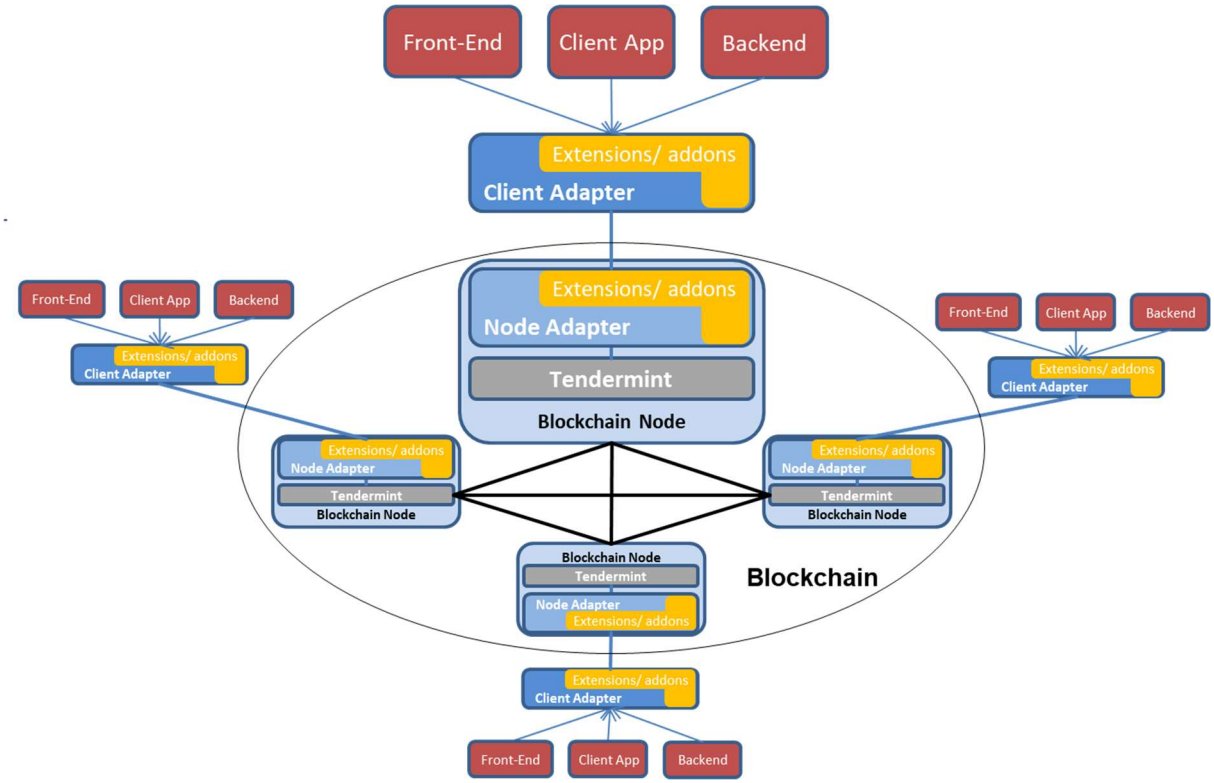

Figure 2: The WRMHL software stack.

- The **Client Adapter** resides within the participant, which is considered a private secure zone. I.e., keys, local private data and proprietary algorithms can be managed by the Client Adapter. Typical Client Adapter functions are:
    - holds the private key for authenticated communication with a node adapter,
    - switches over to an alternate Node Adapter in case of a disconnection
    - reconnects and re-synchs with the new Node Adapter. This takes place transiently within seconds for users of the Client Adapter.
    - provides ability to add application-specific logic,
    - provides an API that is used by Clients/ Client Applications in order to send and receive messages based on a WebSocket interface.

3.2. Transaction/ message processing with WRMHL

Instead of the smart contract programming model, the WRMHL Message API of the Client Adapter is used to send and receive messages. This is especially suitable for the model of 1:N communication/ B2B Integration scenarios because the application programmer's pattern is using the blockchain as a message driven communication/ transaction medium.

In the following it is described how messages/ transactions are processed with WRMHL. Starting point is the end-to-end processing of a message with the standard behaviour "push to all".

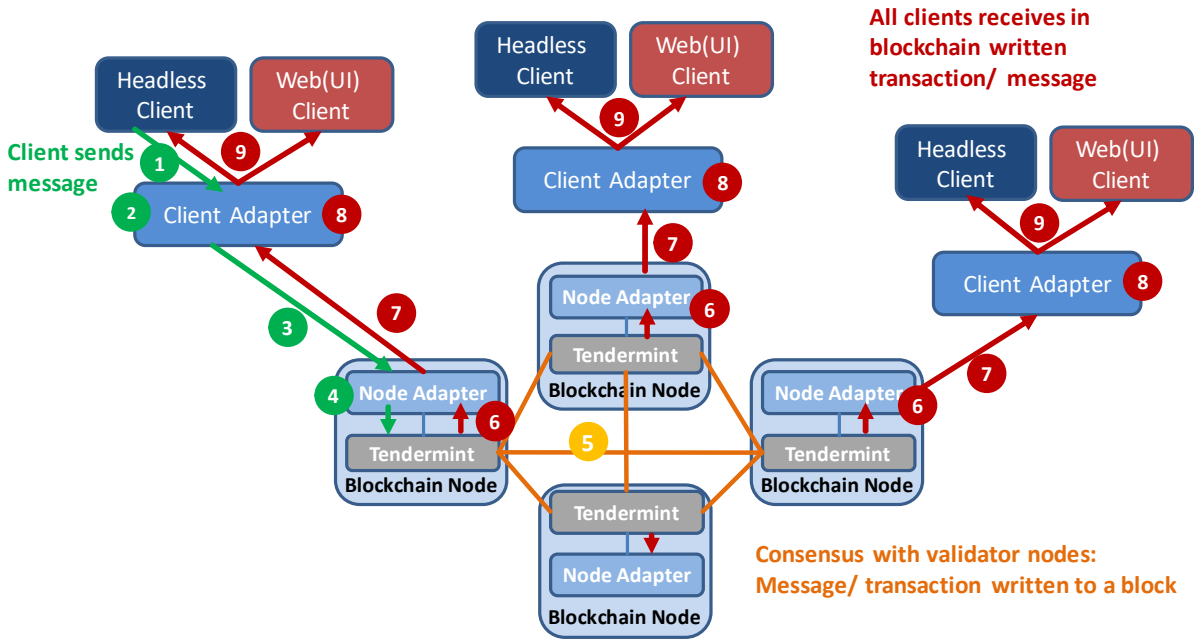

Figure 3: Full cycle of processing a WRMHL message

Step 1: The Client/ Client Application sends a message through the Client Adapter Message API to the Client Adapter. The API standard used for this is WebSocket[6]. It allows to process messages in both directions and in parallel, from the Client to the Client Adapter and vice versa.

Step 2: The Client Adapter receives a message through the Client Adapter Message API. Depending on the message type of the message and implemented application specific Client Adapter Extensions, application-level processing is performed on the messages (e.g. validation, transformation, encryption and/ or filtering).

Step 3: If the processing in "Step 2" is successfully finished, the message is forwarded to the Node Adapter. To do so, the Client Adapter must have established an authenticated and encrypted connection to the Node Adapter. This is only possible by using a private key on the Client Adapter side, which has a related public key, certified by a certificate authority. The Node Adapter verifies if the Client Adapter is certified (as the Client Adapters'

[6] see IETF RFC 6455, https://tools.ietf.org/html/rfc6455

certificates reside with all Node Adapters). Afterwards, messages are exchanged between the Node Adapter and this specific Client Adapter.

Step 4: The Node Adapter itself offers a Message API to a number of Client Adapters. The message is now received from the Client Adapter. Depending on the message type of the message and implemented application specific Node Adapter Extensions, application-level processing is performed on the messages. If this processing is successfully finished, the Node Adapter forwards the message to the node (Tendermint node).

Step 5: At the blockchain level, the message represents a Tendermint transaction (short: Tx). Any communication between Nodes and Node Adapters takes place through Tendermint's ABCI interface[7]. From a node's perspective, transactions just contain bits and bytes.

1. The Node Adapter pushes messages (Tx candidates) to the Tendermint node.
2. Before they are added to the local mempool of the node, a validation takes place by invoking CheckTx() at the Node Adapter. Here the validation of the transaction takes place. Each Tx is hashed by the Node (160bit, RipeMD) to achieve a UUID (Universally Unique Identifier).
3. In case of a positive validation, the Tx is added to the mempool
4. and sent to the neighbouring nodes. For data exchange between nodes the Gossip protocol[8] is used, which ensures that the Tx reaches all nodes after some time.
5. Each node therefore performs a local validation by calling CheckTx() at the Node Adapter and
6. In the successful case the node collects a copy of all transactions in its local mempool.

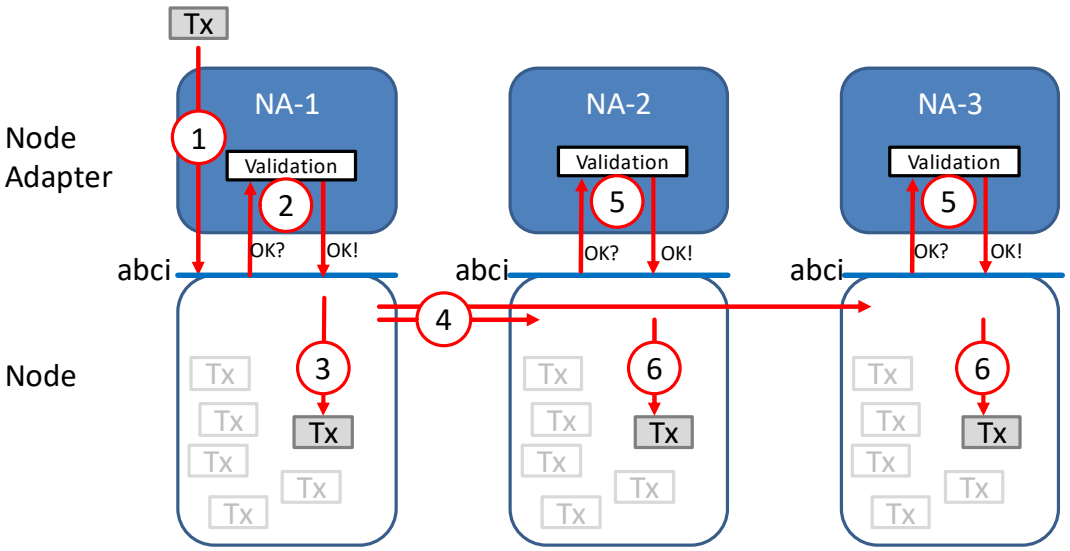

Figure 4: Distributing a message through the blockchain

7. One out of the validating nodes (the so-called proposer) takes all messages (which fit within the configured max Tendermint block size) out of the mempool, groups them into a block, which is ordered on a "first in, first out" basis, creates a block hash and signs it.

[7] http://tendermint.readthedocs.io/projects/tools/en/master/
[8] https://en.wikipedia.org/wiki/Gossip_protocol

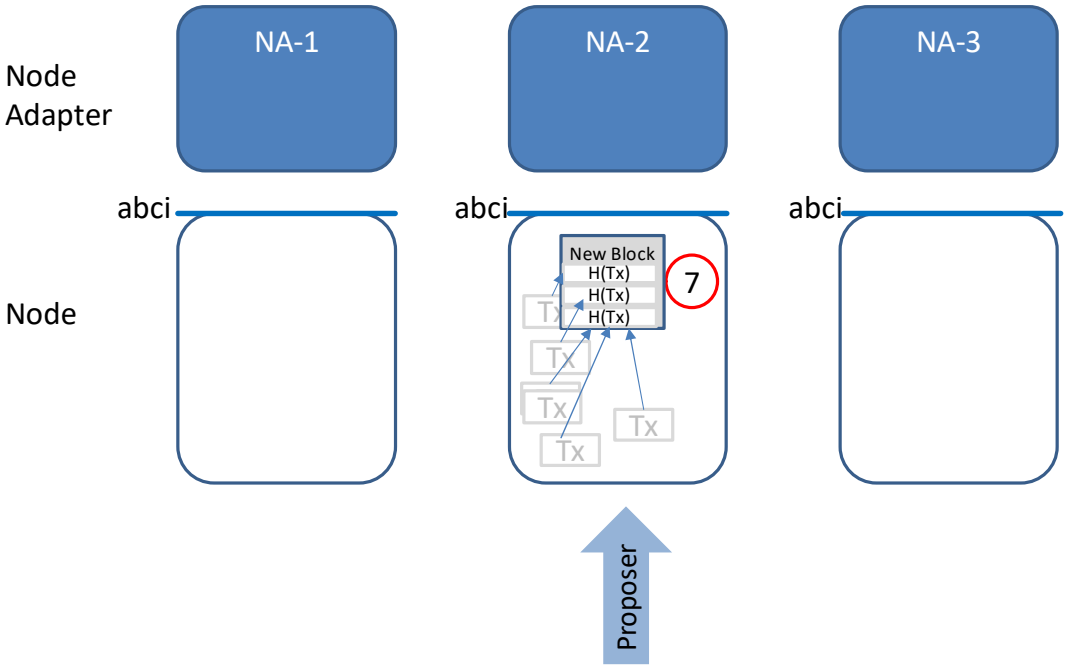

Figure 5: The proposer initiates a consensus round

The consensus round starts with the proposer node sending a propose message to which remaining nodes respond with a pre-vote message:

8. The proposer broadcasts Tx hashes of the proposed block and each validator node responds to all other validators with a pre-vote block message.
9. Based on the received pre-votes each validator calculates if the majority of pre-votes is > 2/3 and sends a pre-commit to the other validators.
10. Finally, if validators have received a pre-commit from > 2/3 of the remaining validators, the state switches to commit-block.

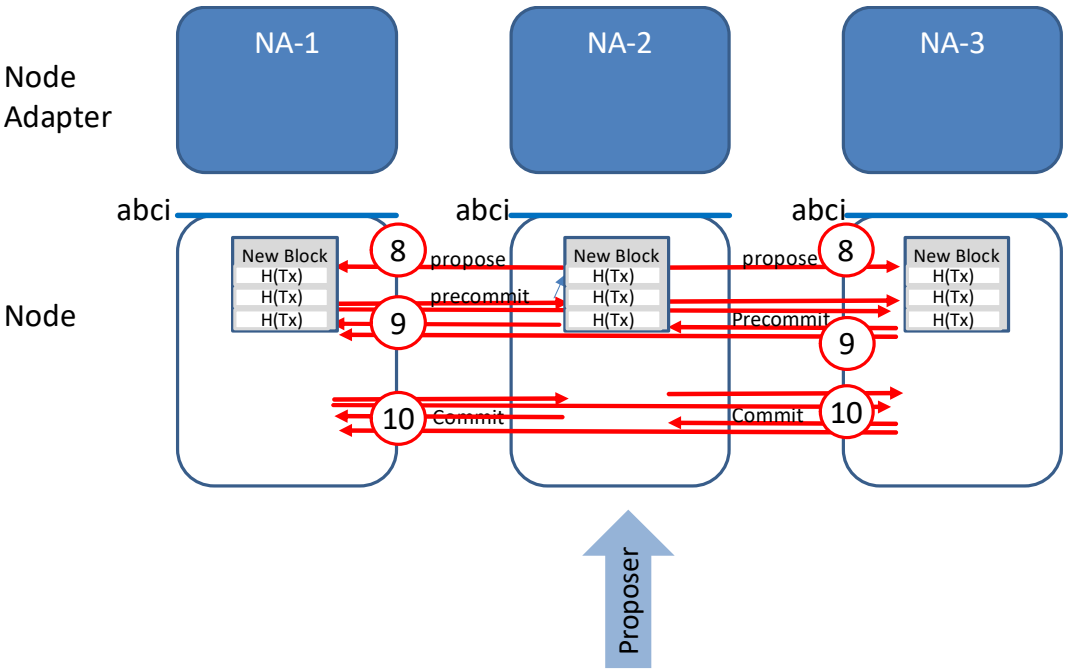

Figure 6: Consensus round

11. Finally, the block is submitted to the application (NA) through the ABCI API, by subsequently invoking BeginBlock(), N * DeliverTx(), EndBlock(), Commit().

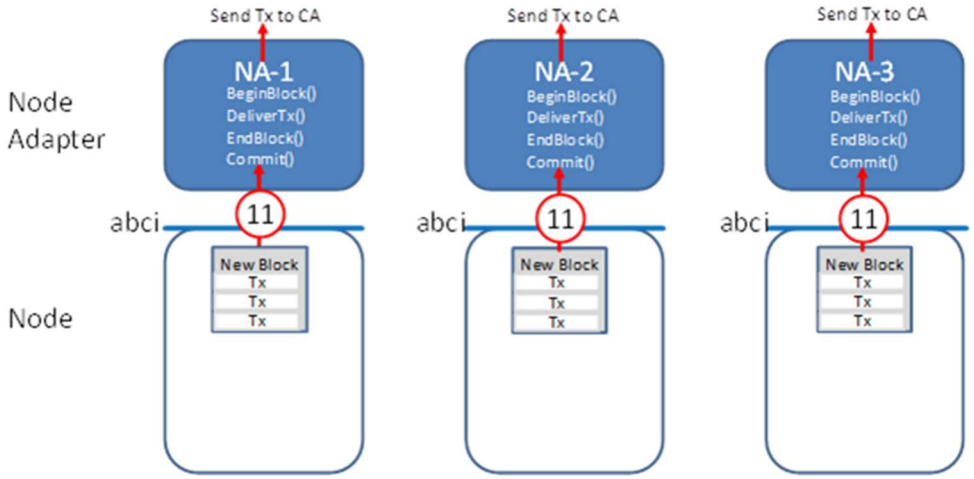

Figure 7: Propagating blocks to Node Adapters

Step 6: The Node Adapter receives the transactions (= WRMHL messages) of the new created block. Depending on the message type of the message and implemented application specific Node Adapter Extensions, application-level processing is performed on the messages.

Step 7: The messages of the new block are forwarded by the Node Adapter to each Client Adapter that is connected to a Node Adapter.

Step 8: The Client Adapter receives the messages. Depending on the message type of the messages and implemented application specific Client Adapter Extensions, application-level processing is performed on the messages.

Step 9: If the default behaviour isn't changed by application specific Client Adapter Extensions, the message is sent to all connected Clients/Client Applications.

3.3. Non deterministic message routing and proposer sequence

The Tendermint gossip protocol mentioned above ensures that the distribution of messages through the blockchain network is different every time a new message is send, so that it cannot be predicted which validator node will receive the next message.

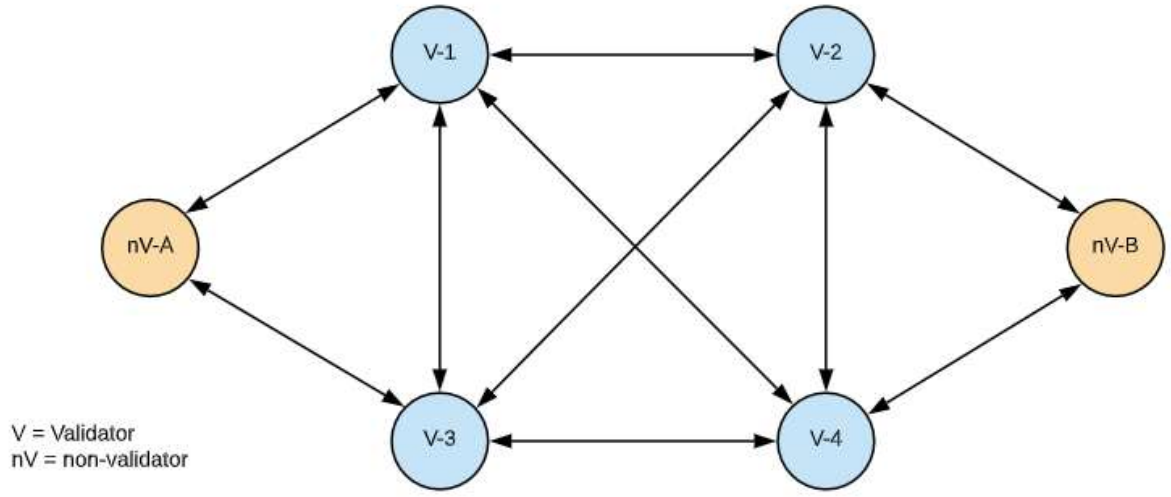

Figure 8: Non deterministic message routing

For example, when participant A sends a message from his Client Adapter to his Non-Validator Node, his node decides to which Validator Node he sends the message first, based on the list of Validators he has. It could be V-1 or any of the other Validators (V-2 to V-4). Afterwards he sends the same message again, but this time he sends it to V-2, in the meantime V-1 sent the message to V-4. This continues till all nodes have received the message. Due to the fact that the distribution after the first connection is random, the message routing will be different each time a new message is sent.

The Tendermint protocol also selects a proposer node in a non-deterministic way. The proposer node is a Validator Node which initiates a consensus round and creates a new block, if the consensus round was successful. When a Tendermint node is the current proposer, this node proposes its mempool as the new block. The proposer selects a fixed order of transactions for the block.

Because the message routing and the proposer node sequence are non-deterministic, it is impossible to say which Validator Node will receive which message first and which Validator Node is the next proposer. Therefore, if two participants try respond to a message at the same time (e.g. execute an order at the same time), no one can forecast which transaction will be accepted and which one will be rejected. Therefore, no network participant can maintain any advantage with regards to message distribution and containment of these messages in new blocks.

3.4. Extending WRMHL with business/ application-specific logic

On both layers, Client Adapter and Node Adapter, WRMHL allows to add business-specific logic, e.g., to process messages through these components in the context of the required business logic. When an empty WRMHL skeleton is deployed, it doesn't know about the message types and message content as this is application-specific. If, e.g., a message contains a data item which represents a date, the business-specific logic extension validates if the value is valid and has the right format. If the application requires, e.g., that the date is a future date, it is again the extension which has to assure that the value is within the expected range. Such logic can only be implemented by application-level developers.

Applications like Enerchain (wholesale trading) or Gridchain[9] (electricity grid operation processes) add such process-specific logic. Each of these applications use the plain WRMHL framework and verticalize it for the specific business logic of the respective application.

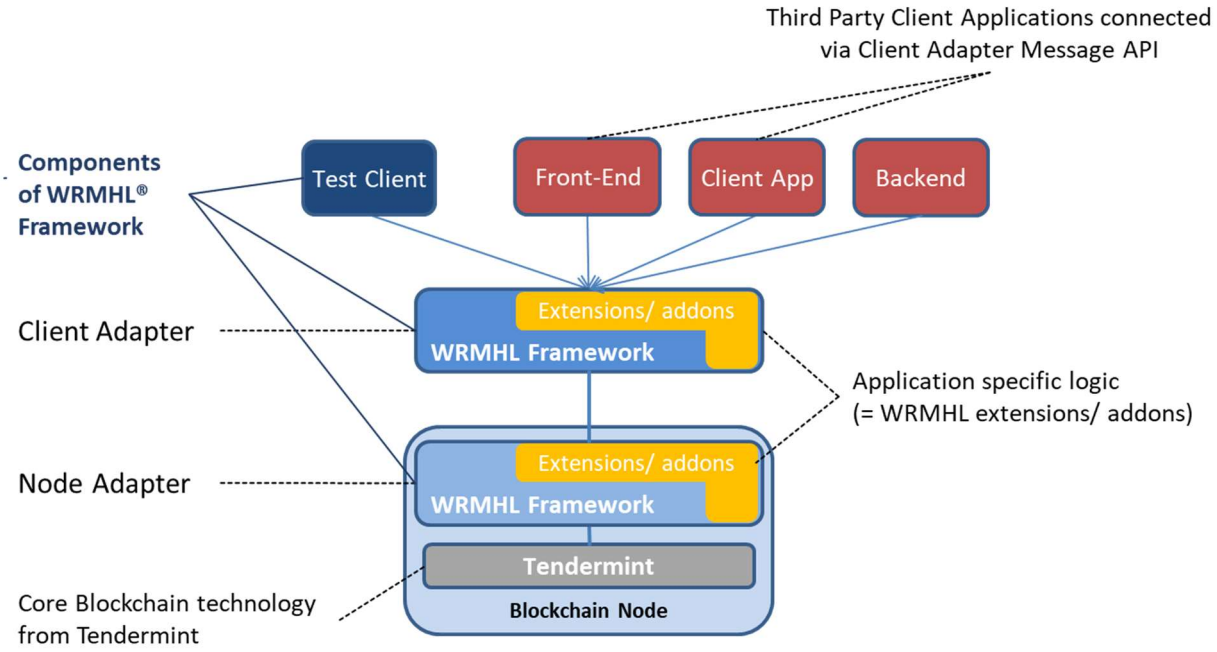

Figure 9: Adding process-specific logic within WRMHL

Below, some functions are discussed in detail. However, the WRMHL development has not yet been completed with these functions. During the course of future projects, the framework will be gradually expanded. At the same time, it is also a goal to keep the complexity of the system as low as possible. More information about future developments can be found on PONTON's WRMHL website: https://wrmhl.ponton.de.

[9] Further information about Gridchain: https://ponton.de/focus/blockchain/gridchain/

The blockchain abstraction layer

The blockchain world is still in motion. While Tendermint today is considered “state of the art” technology another improved technology might be offered as a standard “carrier blockchain” in a few years.

The WRMHL framework abstracts away from technology details. It’s decoupled from individual blockchain technology/ implementation. Thus, the blockchain abstraction represents the lower section of the Node Adapter in the WRMHL architecture.

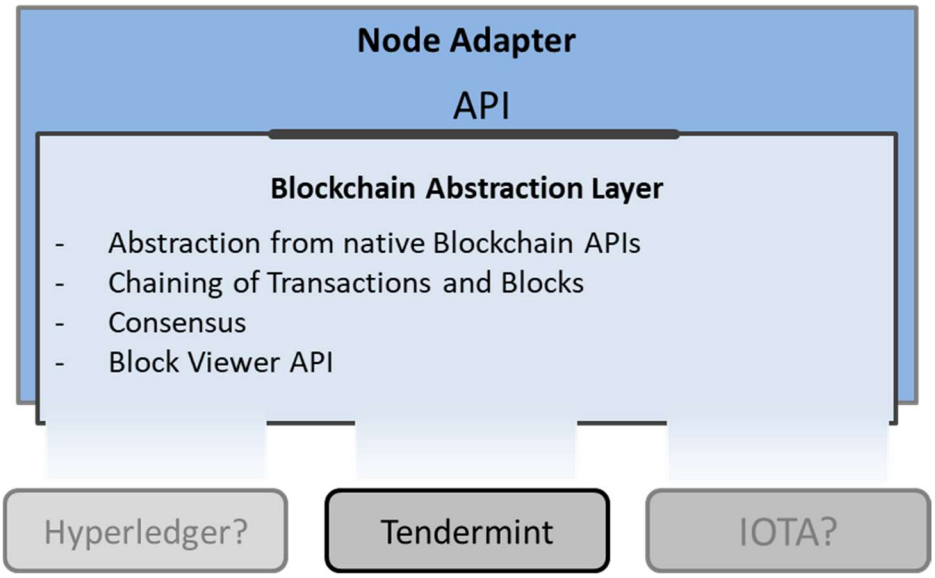

Figure 10: Abstraction of technological particularities through the blockchain abstraction layer

4. WRMHL features

4.1. APIs for developing distributed applications

Applications based on WRMHL must be able to exchange “their” data in such a manner that applications of other participants receive this data whenever they participate in the process. This means, Clients/ Client Applications connected to different Client Adapters are sending and are receiving messages. Clients/ Client Applications connect to a Client Adapter using the Client Adapter Message API of the Client Adapter. The Client Adapter Message API uses the WebSocket transport layer. Messages are represented via JSON data following the envelop-payload pattern. The generic structure shows the following JSON example:

```
"{
        „msgId“:    „a4b65d13-f92c-4b44-828a-cdffe5ae8719“,
        „type“:     „exampleBO“,
        „payload“: {
                    boContent1: “content1”,
                    boContent2: “content2”,
                    ...
        },
}"
```

Developing a distributed application based on WRMHL means that the application specific logic is implemented as a Client Adapter Extension and Node Adapter Extension using the Client Adapter programming API and the Node Adapter programming API. Message types representing business objects or transaction data have to be defined. The required specific application logic can be implemented using the Message Processing Chain of the Client Adapter and Node Adapter.

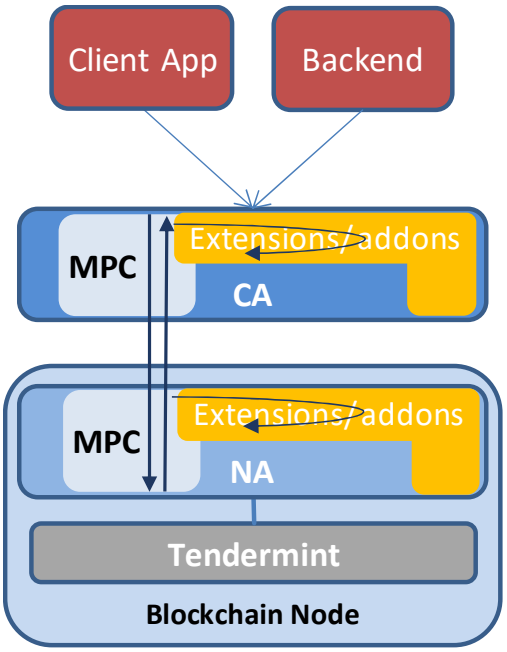

Figure 11: Message Processing Chain

The specific application logic can be for example:

- validation of messages (formal and functional validation, based on customisable set of validation rules). In case of an invalid message, an “invalid massage” response can be sent back to the Client.
- message transformation: the content of messages can be changed.
- compression of messages: messages can be compressed by the Client Adapter.
- encryption/ Anonymisation: a part of the content of a message can be encrypted to hide a part of the message data. The encryption can be also implemented in a way that the originator of the message is anonymous (no pseudonym as in Bitcoin). With this strong anonymisation a GDPR (General Data Protection Regulation) compliant implementation is possible.
- other special processing logic can be activated by the Message Processing Chain, e.g. filtering of messages (not every Client/Client Application needs every message) or a new message is created by the Client Adapter based on receiving a message.

4.2. Authentication and certificate management

Authentication

WRMHL provides an authentication process which ensures that only eligible participants get access to the blockchain. Each organisation participating in the permissioned blockchain network must provide certain information (e.g. ID and full legal name of the participating organisation). This ensures that only companies which are approved by a trusted certificate authority are able to connect with the blockchain network. The authentication process is done transparently for the application level, i.e. the application programmer does not have to handle any authentication issues.

Certificate issuance

A signed certificate is required for the authentication between Client Adapter and Node Adapter. When applying for access to the permissioned blockchain network, participating organisations create a cryptographic key pair and a certification request. Subsequently they send the certification request to a certificate authority with the request to sign it. In response to the applicant's certification request, the certification authority creates a public key certificate for the particular organisation.

Certificate distribution

WRMHL supports the distribution of the valid signed certificates (including the public keys) of the participants of the blockchain network. As soon as a new participant connects the first time with his Client Adapter to a Blockchain Node, the valid signed certificate of the new participant is distributed through the blockchain network. This means that every Node Adapter and Client Adapter stores the new valid signed certificate. As a result, every Client Adapter knows every valid permissioned participant of the blockchain network. The certificate distribution process also allows to update certificates which have been distributed over the blockchain network and the revocation of certificates (under development).

4.3. Role-based access

WRMHL provides a role-based access for connecting Clients/ Client Applications to the Client Adapter of the blockchain network. The ID of the user or of the Client/ Client Application connecting to a Client Adapter needs to have a user role assigned. A user role groups a set of permissions for sending and receiving message types. When a user authenticates against the Client Adapter and sends a message, the role-based access feature in the Client Adapter identifies the message type, checks the role of the user and if the user has the permission to send or receive a message of this message type. If the authorisation is successful the Client Adapter processes the message, if the authorisation fails, the Client Adapter responds with an authorisation violation message. A Client Adapter only sends a message to a connected Client/ Client Application if the permission of receiving this message type is defined in the user role. The use role definitions are configurable and stored by the Client Adapter. The implemented solution enables WRMHL participants to simply create new roles as well as grouping permissions, allowing for a finer authorization.

4.4. Fast data access – caching

The blockchain is not a database and it neither provides for fast access to content stored in the blockchain nor offers a query interface which can be used to select, filter, sort or combine content. The blockchain is simply the virtual location where the “truth” has been stored and this location today consists essentially of no more than a chain of blocks filled with data.

However, it is necessary for applications to efficiently access content. Typically, the most recent transaction data needs to be processed. WRMHL provides a transaction cache where messages of one or several message types can be cached. The transaction caching can be used on Client Adapter level and/ or on Node Adapter level. As the access paths and the business objects or transaction types are application-specific, the transaction cache must be individually set up in the application specific context.

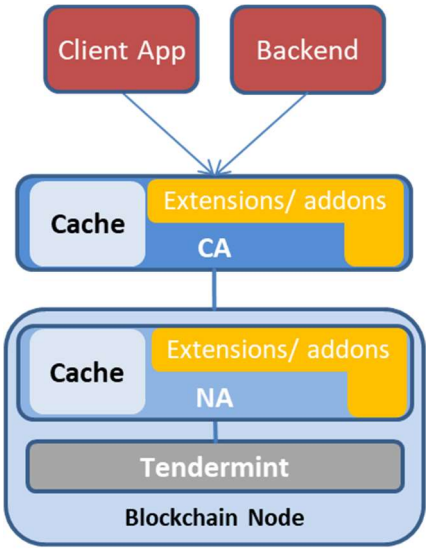

Figure 12: WRMHL transaction cache

4.5. Encryption, data Privacy and anonymisation

WRMHL supports the partial encryption of the payload of a message. The encryption takes place in the Client Adapter based on the public / private key pair stored in the Client Adapter. The private key is bound to the participant (see also “Authentication and certificate management”). As a consequence, the participant has to ensure, that the access to Client Adapters is restricted.
WRMHL supports also encryption methods which can anonymise the creator of a message. The method supports also GDPR compliance. For example, if the identifier of the participant needs to be anonymously stored in a message, the Client Adapter can encrypt the identifier of the participant in a way, that with every new message the encrypted result of the identifier (= the bytes representing the encrypted identifier) is different every time. As a consequence, the participant can’t be identified by the encrypted identifier. The method is secure against statistical analytical attacks, if the number of participants is high enough. Of course, it’s possible to disclose the encrypted identifier to only one other participant. The process of the disclosure can be limited in a way, that the receiver of the message can disclose the identifier of the sender only after an additional related anonymous message form the receiver to sender.

4.6. Monitoring and reliability

WRMHL provides several features ensuring a reliable operation of the complete blockchain network.

System Health Check

System Health Check messages inform Client Adapters and Client/ Client Applications periodically about the health status of the system. If a component is not available or does not respond in a certain configurable time interval, the health status turns to "bad". In case a Client Adapter receives such a “bad” status, the Client Adapter tries to connect to a different Blockchain Node (see below “automated node switching”). Client Applications receiving a “bad” health status can connect to a different Client Adapter of the participant.

Automated node switching

If the abovementioned health check reports a "bad" status to a Client Adapter, the Client Adapter switches to another Node Adapter (meaning Blockchain Node) of the blockchain network. To enable the node switching, other Blockchain Node(s) appropriate for connecting have to be configured in the Client Adapter.
Beyond a “bad” status of a Blockchain Node, a switch to another Blockchain Node may also be necessary if the previous one is no longer reachable. A new connection is automatically established to an alternate Blockchain Node out of the pool of configured nodes if the Client Adapter identifies a connection loss. In this case, the replay process mentioned below starts with this new node.

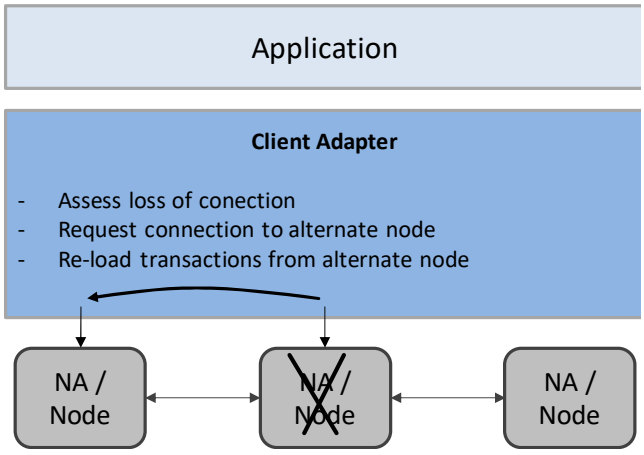

Figure 13: Transparency regarding node failures

The advantage of the automated node switch lies in the independence of the individual node availability. The blockchain remains highly available as an overall system – not just each individual node. The configurable maximum disconnection time of connections from a Client Adapter to Node Adapter (node) is usually 10-15 seconds. It can even be reduced to only a few seconds. In most cases, a Client/ Client Application does not even notice that the node switch took place.

Replay

The replay mechanism provides missed messages from the past to a Client Adapter or to a Client/ Client Application. When a Client/ Client Application connects to a Client Adapter, the Client/ Client Application can request messages from the past, with a block ID as starting point or a "full" replay. A “full” replay means every message which the Client Adapter holds in its replay cache (the cache size is configurable). The same mechanism is in place between Client Adapter and Node Adapter. If a Client Adapter connects to a Node Adapter, the Client Adapter can also request messages from the past. The replay mechanism from Node Adapter to Client Adapter is also used by the automated node switch to ensure that

the Client Adapter, which switched to a different Blockchain Node, receives the missed messages.
The replay mechanism is purely technical and was embedded in the WRMHL framework so that, from an application's perspective, one only needs to wait for a few seconds for the arrival of lost messages and to support the reliable operation of the whole blockchain network.

4.7. Tools

PONTON has developed several stand-alone tools which makes the deployment, the administration and the operation of a WRMHL based blockchain network more convenient.

4.7.1. Block Viewer

The Block Viewer enables users to view the content located in the entire blockchain which has been created through business messages. Business messages representing business transactions and certificates of the blockchain network participants are located in the blockchain and therefore visible to the Block Viewer. Messages stored in the blockchain are in JSON format and syntactically rendered in a canonically visual form by the Block Viewer. The Block Viewer connects directly to a Blockchain Node. As a consequence, the Block Viewer is not able to decrypt any encrypted parts of the stored messages.

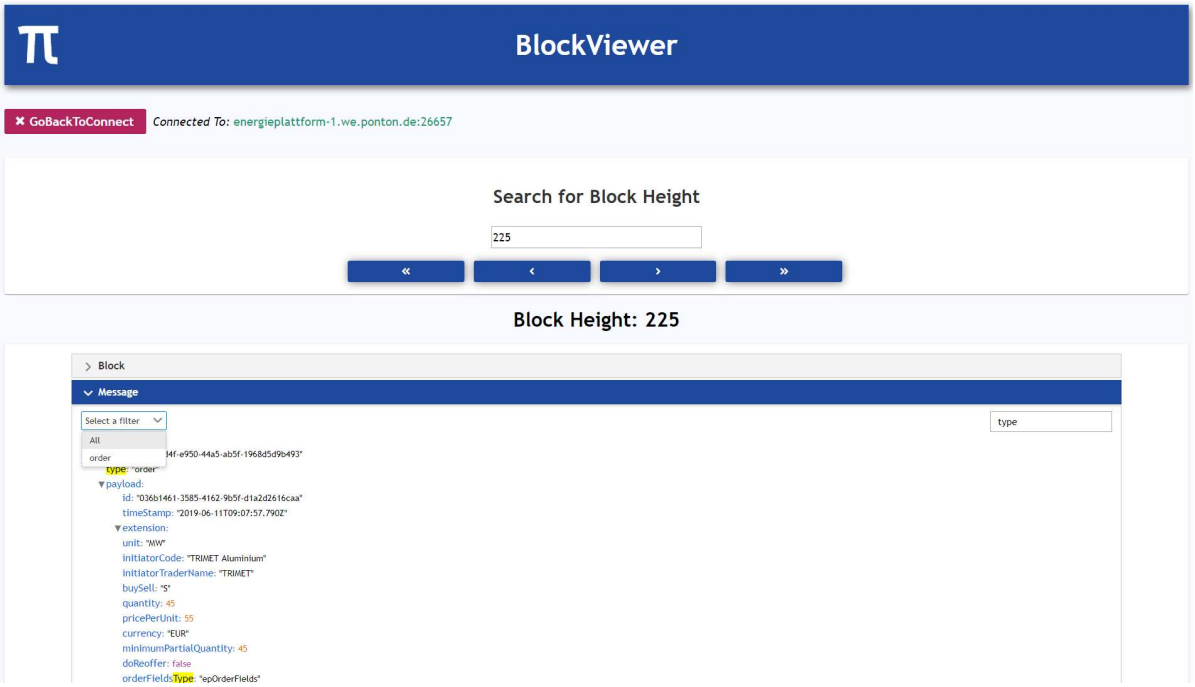

Figure 14: Block Viewer

4.7.2. User ID and Password Hash Generator

The User ID and Password Hash Generator Tool supports participants in creation of users and password-hashes. The authentication of Clients/ Client Applications to a Client Adapter requires a user ID and a password. The Client Adapter holds the user ID in plain writing and the password as hash in a user store, in order to verify credentials of the user when logging-in to the Client/Client Application. The tool is published as extra HTML-file or included in a Client Adapter installation.

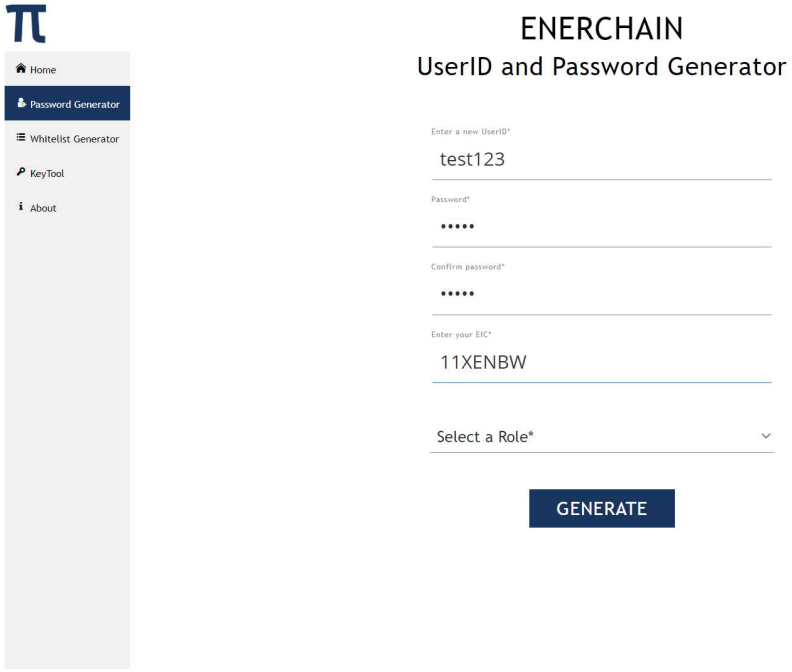

Figure 15: User ID and Password Hash Generator

4.7.3. Message Exporter

The Message Exporter enables the blockchain network participant to receive in real time all messages which were send from his own organisation as well as other organisations and stores these messages in designated file(s), allowing the participant to further process the messages. The Message Exporter connects to a configured Client Adapter of a given participant and receives messages from the Client Adapter and persists them on a filesystem. The participant can configure, which messages will be stored. All entries are stored in plain JSON and correspond to the message type definitions.

5. WRMHL blockchain deployment variants

As explained above WRMHL, which is based on Tendermint, distinguishes between Validator Nodes (part of the consensus mechanism) and Non-Validator Nodes (not part of the consensus mechanism). WRMHL provides different flexible ways to deploy a blockchain network including a minimum set of Validator Nodes and optional Non-Validator Nodes. The question how the deployment should look like is initially less a technical question but rather an organisational and governance question, depending on the organisational and governance frame of the consortia/ group / community using and operating the blockchain. Driving aspects for example are:

- Does an organisation/ legal entity formed by consortia/ group / community exist?
- How many members does the consortia/ group / community have?
- How many participants want to operate a Blockchain Node? (Technically there is no need that every member runs a Blockchain Node)?
- Do some members have a special interest in running Validator Nodes?
- Is a minimal end-to-end transaction time important?

WRMHL supports consortia blockchains with a certain number of Validator Nodes. Typical numbers are 4, 7, 10, 13, etc. This is due to the fact that, whenever a consensus is performed, there is one node playing the role of the proposer node and the remaining nodes vote with a 2/3 majority. I.e., the remaining number of Validator Nodes should be divisible by 3, if each Validator Node is configured with the same voting power and the Validator Nodes are configured as a full meshed network.

At the same time, it should be taken into consideration that the number of Validator Nodes should not be too many with a too high communication latency between each other: if 13 nodes are distributed across different internet locations, all Validating Nodes will potentially exchange messages with all the other Validator Nodes, leading to quite high data traffic.

The best case for low communication latency would be 4 Validator Nodes closely co-located– but here the likelihood of a failure is higher if they all reside in the same local network. So, to achieve a good balance between performance and availability, ideally 7 or 10 nodes should be distributed across 3-4 physical locations (e.g. cloud environments). It is key that the hosted nodes are connected through data channels with low latency and enough communication bandwidth.

Apart from Validator Nodes, participants may also use Non-Validator Nodes. They do not take part in the consensus, but are notified with the finally agreed block at the same time as the Validator Nodes.

Independent from the deployment scenario, the participant or service provider who is operating a Validator Node has to fulfil some service levels, otherwise consensus efficiency may be negatively affected. As mentioned before, a maximum of 1/3 of all Validator Nodes in a network can be not available/ down to avoid interrupting the operation of the entire blockchain network.

In the following sections different potential blockchain deployment variants are defined.

5.1. WRMHL deployment with only Validator Nodes

If there is no need to operate a high number of Blockchain Nodes and/ or the end-to-end transaction time between two participants is not critical, the blockchain network can be operated only with Validator Nodes.

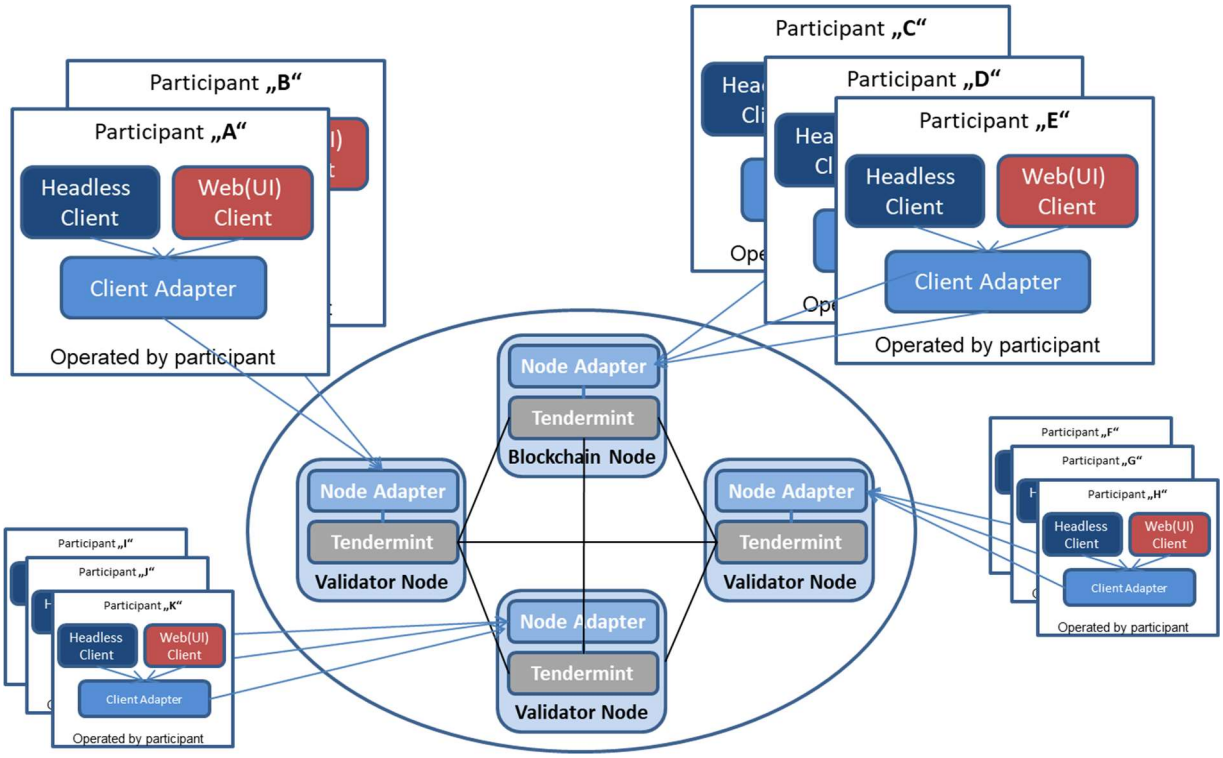

Figure 16: WRMHL Deployment with only Validator Nodes

One Client Adapter can handle many connected Clients/ Client Applications. Many Client Adapters from different participants connect to a Validator Node.

The Client Adapter should be operated by the participant or a trusted service provider of the participant as the Client Adapter stores private data of the participant (private key for encryption, user IDs and user roles).

As mentioned above, the operation of the validator nodes is more an organisational and governance question. The Validator Nodes could be operated by participants or by a service provider on behalf of the consortia.

5.2. WRMHL deployment with Validator Nodes and Non-Validator Nodes

As explained before the operation of a high number of Validator Nodes decreases the end-to-end transaction time between participants. If there is a need to operate a high number of Blockchain Nodes, the blockchain should be based on Validator Nodes and Non-Validator Nodes. With a mixture of Validator Nodes and Non-Validator Nodes there are two base scenarios on how to organise the nodes.

Scenario 1 –
Validator Nodes and Non-Validator Nodes are organised without hierarchy

The deployment defines no hierarchy between Validator Nodes and Non-Validator Nodes because from an underlying governance model of a consortia it is irrelevant if a participant connects to a Validator Node or to a Non-Validator Node. Some participants operate Validator Nodes and other participants operate Non-Validator Nodes.

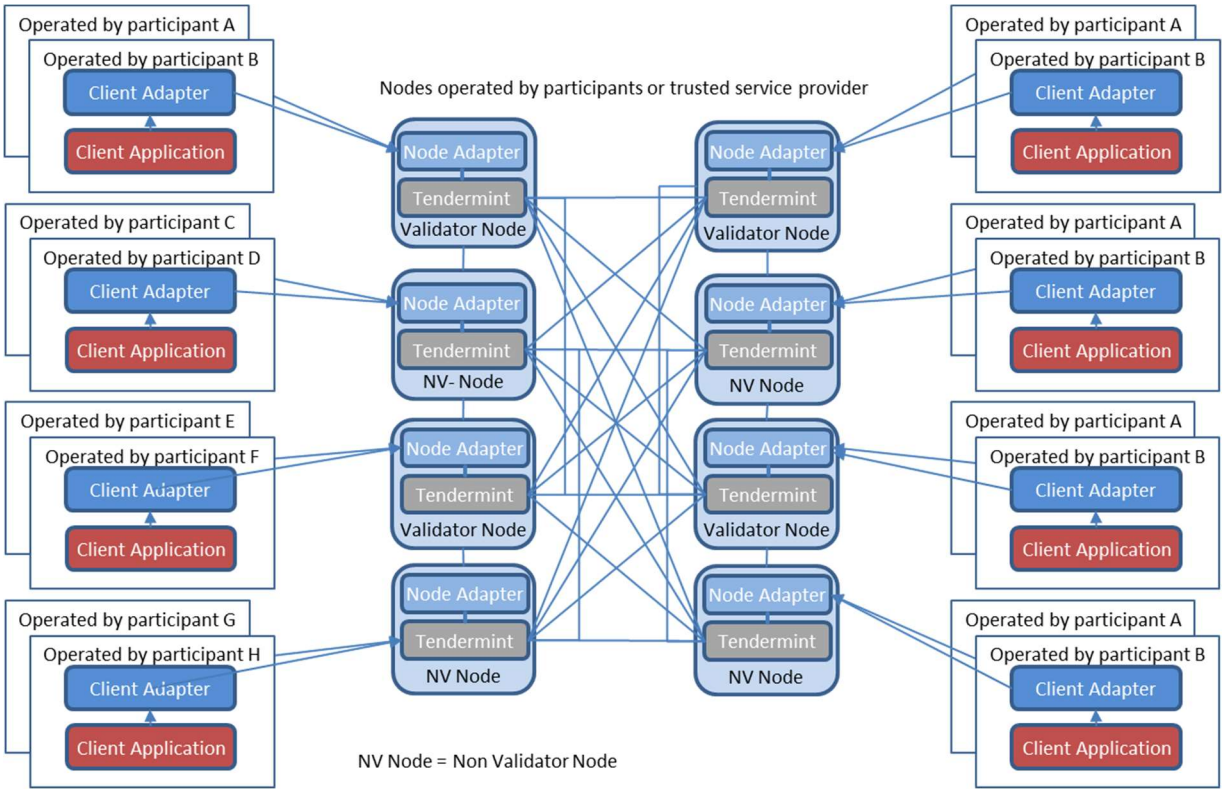

Figure 17: Deployment without Blockchain Node hierarchy

In this deployment scenario the option exists, that a participant operates no node and connects with his Client Adapter to a node of another participant or a service provider.

Scenario 2 –
Validator Nodes and Non-Validator Nodes are hierarchal organised

If it would be an advantage for a participant to operate a Validator Node instead of operating a Non-Validator Node so it is a valid scenario to organise Validator Nodes and Non-Validator Nodes in a hierarchy and allow Client Adapters to connect only to a Non-Validator Node.

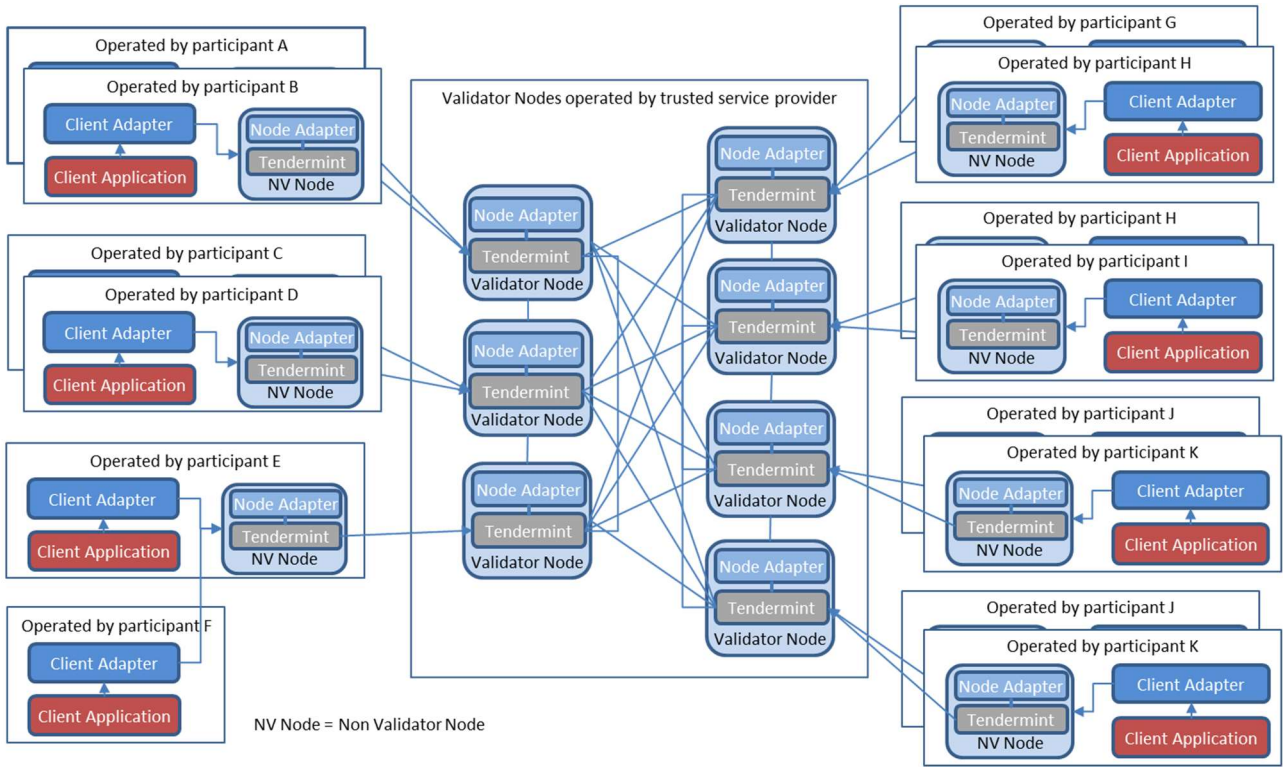

Figure 18: Deployment with hierarchal structuring of the Blockchain Nodes

In this deployment scenario the Client Adapters of every participant can only connect to Non-Validator Nodes. The Non-Validator Nodes are normally connected to a Validator Node, but could also connect to another Non-Validator Node. Participants can operate their “own” Non-Validator Nodes, a group of participants can share the operation of a Non-Validator Node or a participant can connect with his Client Adapter to a Non-Validator Node operated by another participant. The set of Validator Nodes should be operated by a trusted service provider in behalf of the consortia.

6. Contact

Are you interested in using WRMHL for your B2B integration process?

Please contact PONTON at wrmhl@ponton.de.

You can also join a webinar with PONTON and discuss how WRMHL can support your use case.

7. References

[Merz19] Michael Merz: Blockchain im B2B-Einsatz (German), MM Publishing, 2019. English translation to appear in 2019.