# Identity Management

## Overview

Identity management is a subsystem that provides the overall identification capability for a node's identity. This includes: - Delegate's private IP/port advertisement - Tx-Acceptor public IP/port advertisement - Handshake between Tx-Acceptor and the delegate - Handshake between the delegates - P2p commands to request delegates IP/port - Private key management

## Use cases

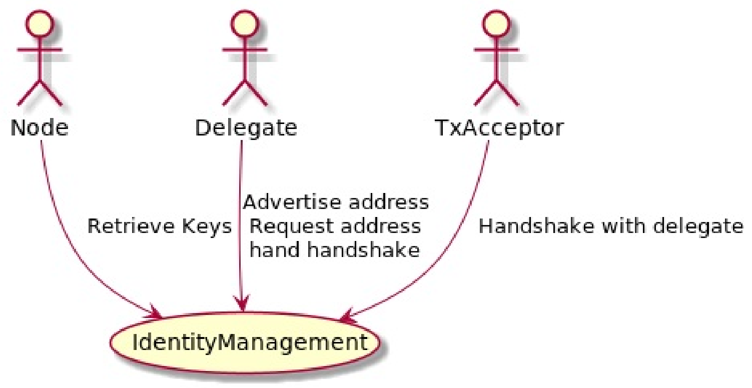

Figure 1. Identity Management use case diagram.

## Execution concept

Identity manager is driven by the following events: *Node start up to advertise delegate and tx-acceptor address* One hour before the epoch start to advertise delegate and tx-acceptor address *Thirty minutes before the epoch start to advertise delegate and tx-acceptor address* Every ten minutes starting at one hour before the epoch start to request delegate's addresses *RPC call to add or delete tx-acceptor* Unsolicited P2p delegate's address advertisement message

Details of these events are covered in the sections below.

## Delegate and Tx-Acceptor IP/port advertisement

### Class Diagram

Identity management extends existing and implements new classes: - AddressAd is the delegate's address advertisement message. The message has ip and port encrypted with ECIES public key of every delegate except for the sender. The hash of the message is signed with the sender's private BLS key. - AddressAdTxAcceptor is the delegate's tx acceptor advertisement message. The hash of the message is signed with the sender's private BLS key. - P2pConsensusHeader precedes P2p consensus message. The header identifies receiver of the consensus message. - P2pHeader is the main p2p message header. If app_type is Consensus then it is followed by P2pConsensusHeader, which in turn is followed by corresponding consensus message. If app_type is AddressAd then it is followed by AddressAd. If app_type is AddressAdTxAcceptor then it is followed by AddressAddTxAcceptor. - DelegateIdentityManager handles all identity manager functionality related to advertisement and persistence of the delegate and tx acceptor addresses.

The class diagram is shown on Figure 2.

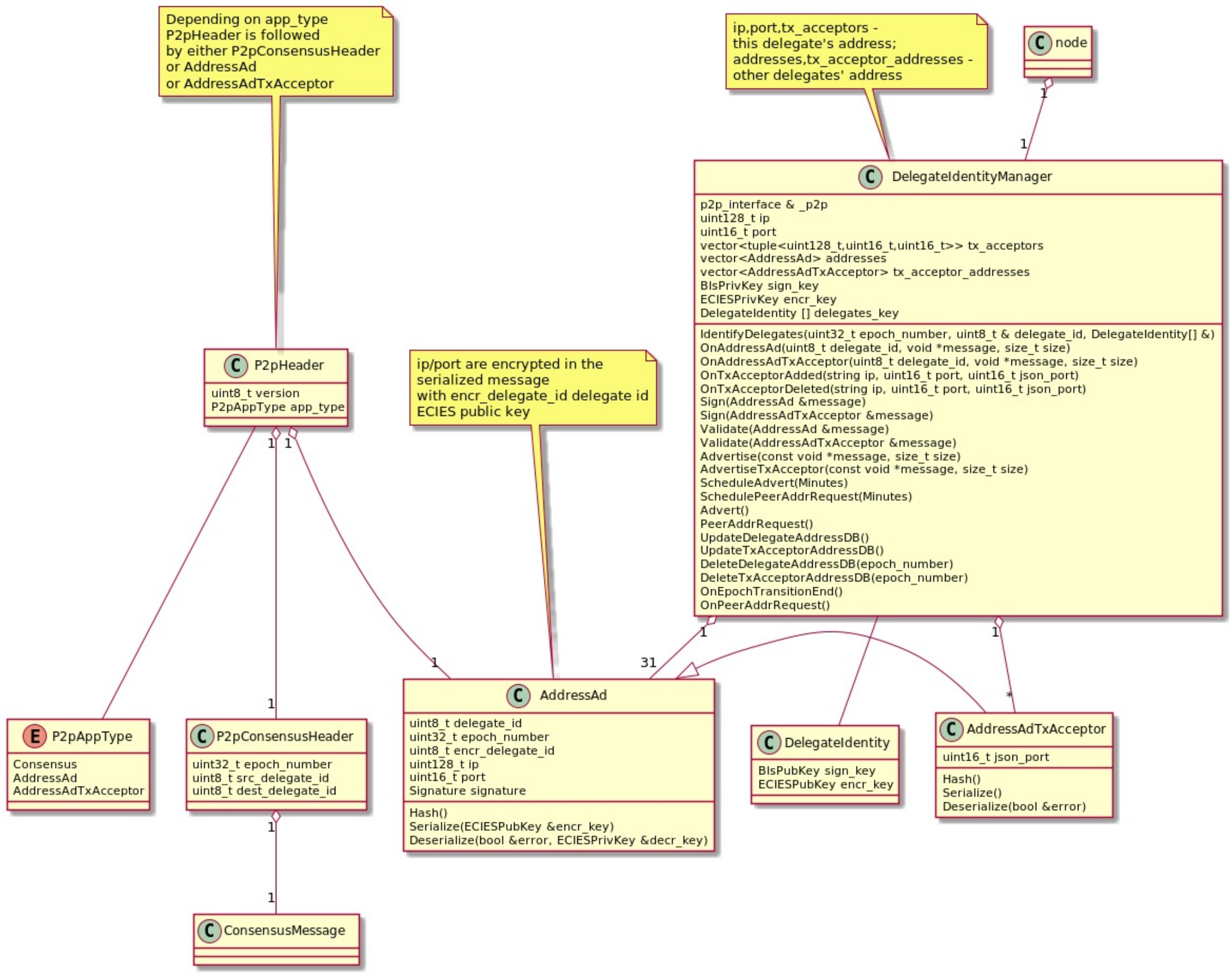
Depending on app_type P2pHeader is followed by either P2pConsensusHeader or AddressAd or AddressAdTxAcceptor
ip,port,tx_acceptors - this delegate's address; addresses,tx_acceptor_addresses - other delegates' address
node
1
1
DelegateIdentityManager
p2p_interface & _p2p
uint128_t ip
uint16_t port
vector<tuple<uint128_t,uint16_t,uint16_t>> tx_acceptors
vector<AddressAd> addresses
vector<AddressAdTxAcceptor> tx_acceptor_addresses
BlsPrivKey sign_key
ECIESPrivKey encr_key
DelegateIdentity [] delegates_key
IdentifyDelegates(uint32_t epoch_number, uint8_t & delegate_id, DelegateIdentity[] &)
OnAddressAd(uint8_t delegate_id, void *message, size_t size)
OnAddressAdTxAcceptor(uint8_t delegate_id, void *message, size_t size)
OnTxAcceptorAdded(string ip, uint16_t port, uint16_t json_port)
OnTxAcceptorDeleted(string ip, uint16_t port, uint16_t json_port)
Sign(AddressAd &message)
Sign(AddressAdTxAcceptor &message)
Validate(AddressAd &message)
Validate(AddressAdTxAcceptor &message)
Advertise(const void *message, size_t size)
AdvertiseTxAcceptor(const void *message, size_t size)
ScheduleAdvert(Minutes)
SchedulePeerAddrRequest(Minutes)
Advert()
PeerAddrRequest()
UpdateDelegateAddressDB()
UpdateTxAcceptorAddressDB()
DeleteDelegateAddressDB(epoch_number)
DeleteTxAcceptorAddressDB(epoch_number)
OnEpochTransitionEnd()
OnPeerAddrRequest()
P2pHeader
uint8_t version
P2pAppType app_type
ip/port are encrypted in the serialized message with encr_delegate_id delegate id ECIES public key
1
1
1
1
1
31
1
*
P2pAppType
Consensus
AddressAd
AddressAdTxAcceptor
P2pConsensusHeader
uint32_t epoch_number
uint8_t src_delegate_id
uint8_t dest_delegate_id
AddressAd
uint8_t delegate_id
uint32_t epoch_number
uint8_t encr_delegate_id
uint128_t ip
uint16_t port
Signature signature
Hash()
Serialize(ECIESPubKey &encr_key)
Deserialize(bool &error, ECIESPrivKey &decr_key)
DelegateIdentity
BlsPubKey sign_key
ECIESPubKey encr_key
AddressAdTxAcceptor
uint16_t json_port
Hash()
Serialize()
Deserialize(bool &error)
1
1
ConsensusMessage

Figure 2. Class diagram.

## Delegate Address Encryption

Elliptic curve integrated encryption scheme (ECIES) is used for the delegate's address encryption. ECIES private key is retrieved by the delegatate via the private key management capability. ECIES public keys are distributed via the epoch block along with the BLS public keys. During AddressAd message serialization, IP and port members of the message are encrypted with the ECIES public key of every other delegate depending on client/server relationship in the delegate's connection schema. This reduces by half the number of AdMessage required to be propagated via P2p subsystem. In the TCP/IP connection process, one delegate is a client (delegate with lower delegate id) and another delegate is a server. For instance, when delegate 1 and delegate 2 establish connection with each other, delegate 1 is the client and delegate 2 is the server. Since delegate 2 doesn't need delegate's 1 IP for connection, delegate 1 doesn't have to advertise it's ip/port, which could be sent instead to delegate 2 as part of the handshake after delegate 1 connects to delegate 2.

## Sequence Diagram Delegate and Tx Acceptor Address Advertisement

On start up the node checks if it is the delegate in the current or next epoch (IdentifyDelegates()).

If it is the delegate then it advertises via P2p subsystem its encrypted address and all of its tx acceptors' address (Advertise(), AdvertiseTxAcceptor()). If the node is the delegate in the next epoch then depending on the current time the node schedules two timers at 1 hour and at 30 min before the start of the next epoch (ScheduleAdvert()). On timer's timeout (Advert()), the node advertises via P2p subsystem its encrypted address and all of its tx acceptors' address. On EpochTransitionEnd event, if the node is not the delegate in the current or next epoch then it repeats the advertisement sequence (AdvertCheck). When a node receives tx acceptor address via RPC call (OnTxAcceptorAdded), it saves the address to the database (UpdateTxAcceptorAddressDB()). The address is also stored by DelegateIdentityManager (tx_acceptor_addresses). If the node is the delegate in the current or next epoch then it advertises tx acceptor address. When a node receives tx acceptor address deletion RPC call then the node advertises tx acceptor deletion via P2p subsystem. The node deletes tx acceptor address from the database. Upon receiving a recall event, if the node is the delegate in the next epoch, it advertises via P2p subsystem its encrypted address and all of its tx acceptors' address. Sequence diagram for the address advertisement is shown on Figure 3.

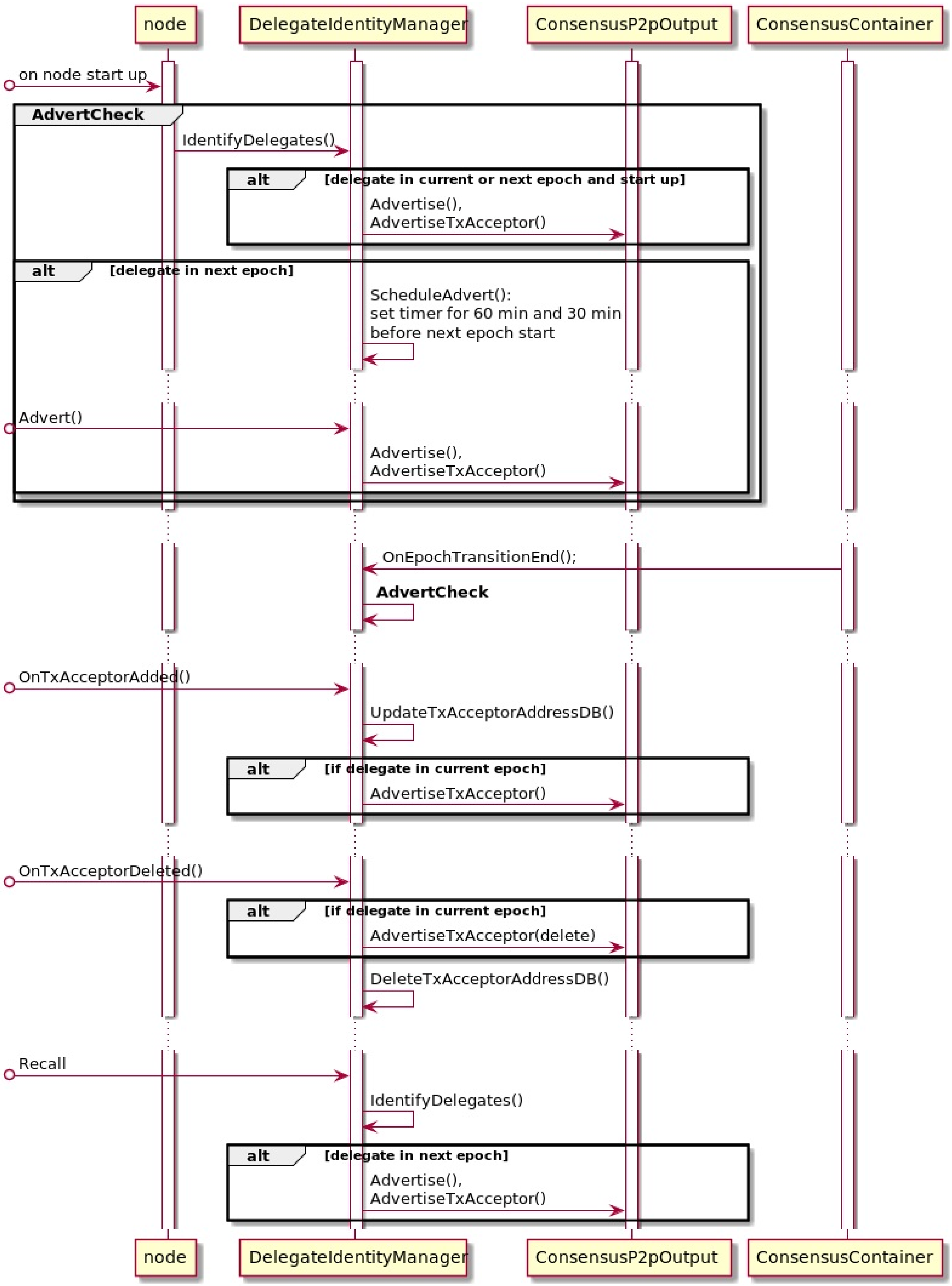
node
DelegateIdentityManager
ConsensusP2pOutput
ConsensusContainer
on node start up
AdvertCheck
IdentifyDelegates()
alt [delegate in current or next epoch and start up]
Advertise(), AdvertiseTxAcceptor()
alt [delegate in next epoch]
ScheduleAdvert(): set timer for 60 min and 30 min before next epoch start
Advert()
Advertise(), AdvertiseTxAcceptor()
OnEpochTransitionEnd();
AdvertCheck
OnTxAcceptorAdded()
UpdateTxAcceptorAddressDB()
alt [if delegate in current epoch]
AdvertiseTxAcceptor()
OnTxAcceptorDeleted()
alt [if delegate in current epoch]
AdvertiseTxAcceptor(delete)
DeleteTxAcceptorAddressDB()
Recall
IdentifyDelegates()
alt [delegate in next epoch]
Advertise(), AdvertiseTxAcceptor()
node
DelegateIdentityManager
ConsensusP2pOutput
ConsensusContainer

Figure 3. Sequence diagram, address advertisement

## Sequence Diagram Delegate Address and Tx Acceptor Address Receiving

On start up the node ascertains if it is the delegate in the current or next epoch (IdentifyDelegates()). If it is the delegate and it doesn't have addresses of all other delegates then the delegate requests missing addresses from its peers via P2p subsystem (RequestDelegateAddresses()) and schedules 10 min timer (SchedulePeerAddrRequest()). When the delegate receives the response back from its peers (OnPeerAddrRequest()), it updates delegates address (UpdateDelegateAddressDB()) and connects to the received peers address (ConnectToPeer()) if it is the delegate in the current epoch. Connected peer sends it's AdAddress message as part of the handshake and the delegate updates the delegates address. On timeout (PeerAddrRequest()) the delegate repeats the sequence (CheckDelegateAddress). When a node receives a delegate's or tx acceptor address advertisement (OnAddressAd(), OnAddressAdTxAcceptor()), and if the node is the delegate in the current or the epoch then it updates the address database. The address is also stored by DelegateIdentityManager (addresses, tx_acceptor_addresses).If the node is the delegate in the current epoch then it connects to the received delegate address (ConnectToPeer()). The node also deletes older epoch entries for the given delegate id. Sequence diagram for the address receiving is shown on Figure 4.

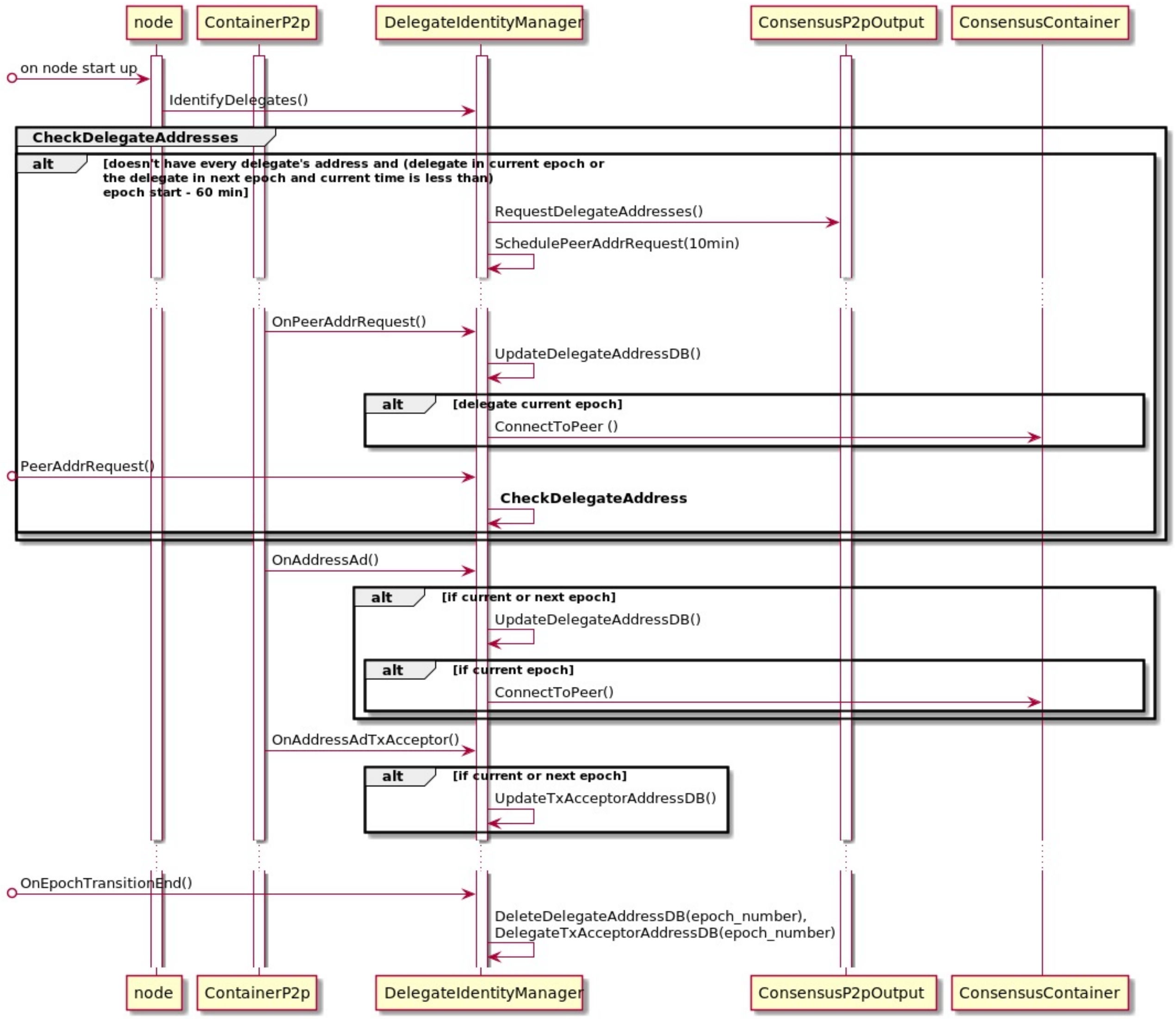
node
ContainerP2p
DelegateIdentityManager
ConsensusP2pOutput
ConsensusContainer
on node start up
IdentifyDelegates()
CheckDelegateAddresses
alt
[doesn't have every delegate's address and (delegate in current epoch or the delegate in next epoch and current time is less than) epoch start - 60 min]
RequestDelegateAddresses()
SchedulePeerAddrRequest(10min)
OnPeerAddrRequest()
UpdateDelegateAddressDB()
alt
[delegate current epoch]
ConnectToPeer ()
PeerAddrRequest()
CheckDelegateAddress
OnAddressAd()
alt
[if current or next epoch]
UpdateDelegateAddressDB()
alt
[if current epoch]
ConnectToPeer()
OnAddressAdTxAcceptor()
alt
[if current or next epoch]
UpdateTxAcceptorAddressDB()
OnEpochTransitionEnd()
DeleteDelegateAddressDB(epoch_number),
DelegateTxAcceptorAddressDB(epoch_number)
node
ContainerP2p
DelegateIdentityManager
ConsensusP2pOutput
ConsensusContainer

Figure 4.Sequence diagram, address receiving.

## Handshake between the delegates

High level sequence diagram on Figure 5 shows AdAddress message propagation via P2p subsystem and handshake that takes place when delegates connect to each other.

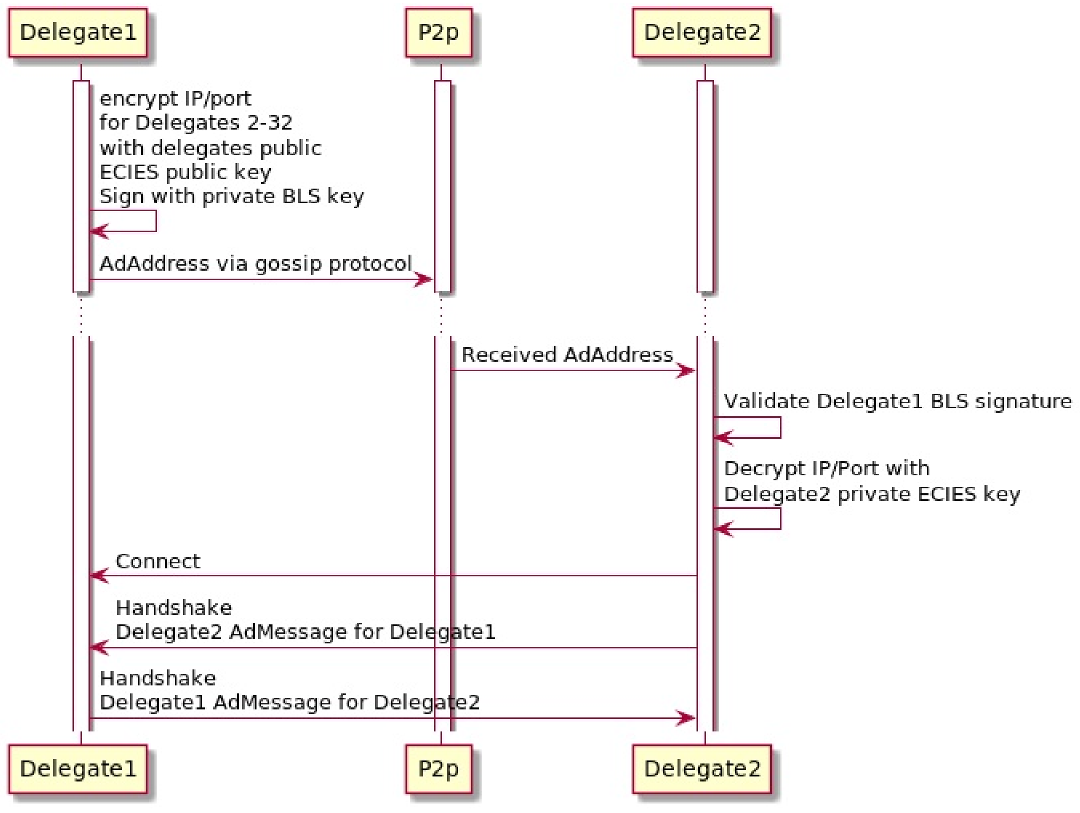
Delegate1
P2p
Delegate2
encrypt IP/port
for Delegates 2-32
with delegates public
ECIES public key
Sign with private BLS key
AdAddress via gossip protocol
Received AdAddress
Validate Delegate1 BLS signature
Decrypt IP/Port with
Delegate2 private ECIES key
Connect
Handshake
Delegate2 AdMessage for Delegate1
Handshake
Delegate1 AdMessage for Delegate2
Delegate1
P2p
Delegate2

Figure 5. Sequence diagram, AdMessage propagation and delegate's handshake.

## Handshake between Tx-Acceptor and the delegate

High level sequence diagram on Figure 6 shows handshake interaction between TxAcceptor and Delegate. Once the handshake completes, TxAcceptor forwards to TxAcceptor transactions received from the client.

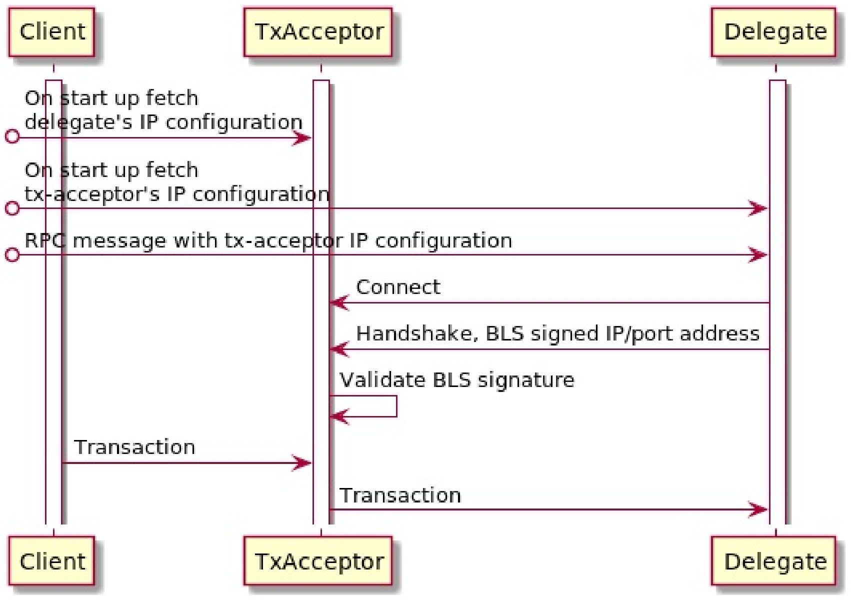
Client
TxAcceptor
Delegate
On start up fetch
delegate's IP configuration
On start up fetch
tx-acceptor's IP configuration
RPC message with tx-acceptor IP configuration
Connect
Handshake, BLS signed IP/port address
Validate BLS signature
Transaction
Transaction
Client
TxAcceptor
Delegate

Figure 6. Sequence diagram, TxAcceptor and delegate handshake.

## P2p commands to request delegates IP/Port (TBD)

## Private key management (TBD)