# Identity Management

## Overview

Identity management is responsible for broadcasting the delegate’s encrypted IP to other delegates. It uses the wallet service to retrieve in a secure way its private bls keys. Bls public key is used to retrieve the list of delegates in the current epoch. Asymmetric public keys of other delegates are used for encryption of the delegate’s address advertisement. The delegate authenticates the message by signing it with the private bls key.

## Class Diagram

Identity management extends existing classes and implements new classes: - AddressAdMessage 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 and the hash of the encrypted data and other data is signed with the sender's private BLS key. AddressAdMessageTxAcceptor is the delegate's tx acceptor advertisement message. The hash of the message is signed with the sender's private 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 AddressAdMessage. If app_type is AddressAdTxAcceptor then it is followed by AddressAddMessageTxAcceptor. - 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 1.

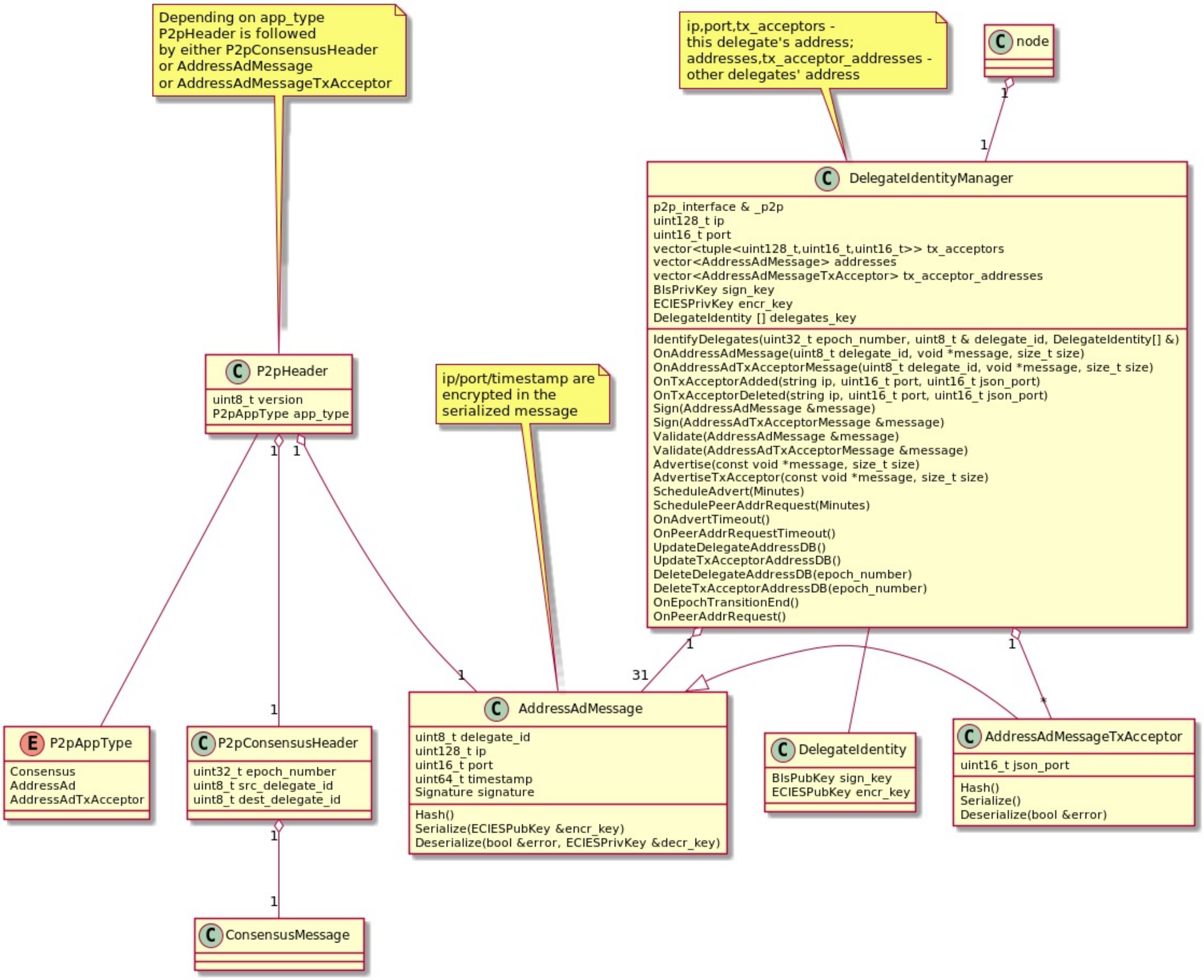
Depending on app_type
P2pHeader is followed
by either P2pConsensusHeader
or AddressAdMessage
or AddressAdMessageTxAcceptor
ip,port,tx_acceptors -
this delegate's address;
addresses,tx_acceptor_addresses -
other delegates' address
node
1
1
DelegateIdentityManager
p2p_interface & _p2p
uint128_t ip
uint16_t port
vector<tuple<uint128_t,uint16_t,uint16_t>> tx_acceptors
vector<AddressAdMessage> addresses
vector<AddressAdMessageTxAcceptor> tx_acceptor_addresses
BlsPrivKey sign_key
ECIESPrivKey encr_key
DelegateIdentity [] delegates_key
IdentifyDelegates(uint32_t epoch_number, uint8_t & delegate_id, DelegateIdentity[] &)
OnAddressAdMessage(uint8_t delegate_id, void *message, size_t size)
OnAddressAdTxAcceptorMessage(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(AddressAdMessage &message)
Sign(AddressAdTxAcceptorMessage &message)
Validate(AddressAdMessage &message)
Validate(AddressAdTxAcceptorMessage &message)
Advertise(const void *message, size_t size)
AdvertiseTxAcceptor(const void *message, size_t size)
ScheduleAdvert(Minutes)
SchedulePeerAddrRequest(Minutes)
OnAdvertTimeout()
OnPeerAddrRequestTimeout()
UpdateDelegateAddressDB()
UpdateTxAcceptorAddressDB()
DeleteDelegateAddressDB(epoch_number)
DeleteTxAcceptorAddressDB(epoch_number)
OnEpochTransitionEnd()
OnPeerAddrRequest()
P2pHeader
uint8_t version
P2pAppType app_type
1
1
ip/port/timestamp are
encrypted in the
serialized message
1
31
1
1
*
1
P2pAppType
Consensus
AddressAd
AddressAdTxAcceptor
P2pConsensusHeader
uint32_t epoch_number
uint8_t src_delegate_id
uint8_t dest_delegate_id
AddressAdMessage
uint8_t delegate_id
uint128_t ip
uint16_t port
uint64_t timestamp
Signature signature
Hash()
Serialize(ECIESPubKey &encr_key)
Deserialize(bool &error, ECIESPrivKey &decr_key)
DelegateIdentity
BlsPubKey sign_key
ECIESPubKey encr_key
AddressAdMessageTxAcceptor
uint16_t json_port
Hash()
Serialize()
Deserialize(bool &error)
1
1
ConsensusMessage

Figure 1. Class diagram.

# Sequence Diagram Delegate and Tx Acceptor Address Advertisement

On start up the node ascertains if it is the delegate in the current epoch (IdentifyDelegates()). If it is the delegate then it advertises via P2p substym its encrypted address and all of its tx acceptor addresses (Advertise(), AdvertiseTxAcceptor()). If the node is the delegate in the next epoch and the current time is greater than one hour before the epoch start then the node schedules a timer (ScheduleAdvert) at (epoch start - now - 60min). On timeout (OnAdvertTimeout()), the node advertises via P2p substym its encrypted address and all of its tx acceptor addresses and schedules a timer (epoch start - now - 30min). On timeout, the node advertises via P2p subsystem its encrypted address and all of its tx acceptor addresses. The node also executes CheckDelegateAddress (Figure 3). If the node is the delegate in the next epoch and the current time is less than one hour before the epoch start and greater than 30min before epoch start then the node advertises via P2p subsystem its encrypted address and all of its tx acceptor addresses and schedules a timer at (epoch start - now - 30min). On timeout, the node advertises via P2p substym its encrypted address and all of its tx acceptor addresses. If the node is the delegate in the next epoch and the current time is less than 30min before the epoch start then the node advertises via P2p substym its encrypted address and all of its tx acceptor addresses. If the node is not the delegate in the current or next epoch then it doesn't execute and action and on

EpochTransitionEnd event it repeats the advertisement sequence (AdvertCheck). When a node receives tx acceptor address via rps call (OnTxAcceptorAdded), it saves the address to the database (UpdateTxAcceptorAddressDB()). If the node is the delegate in the current 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 from the database. Sequence diagram for the address advertisement is shown on Figure 2.

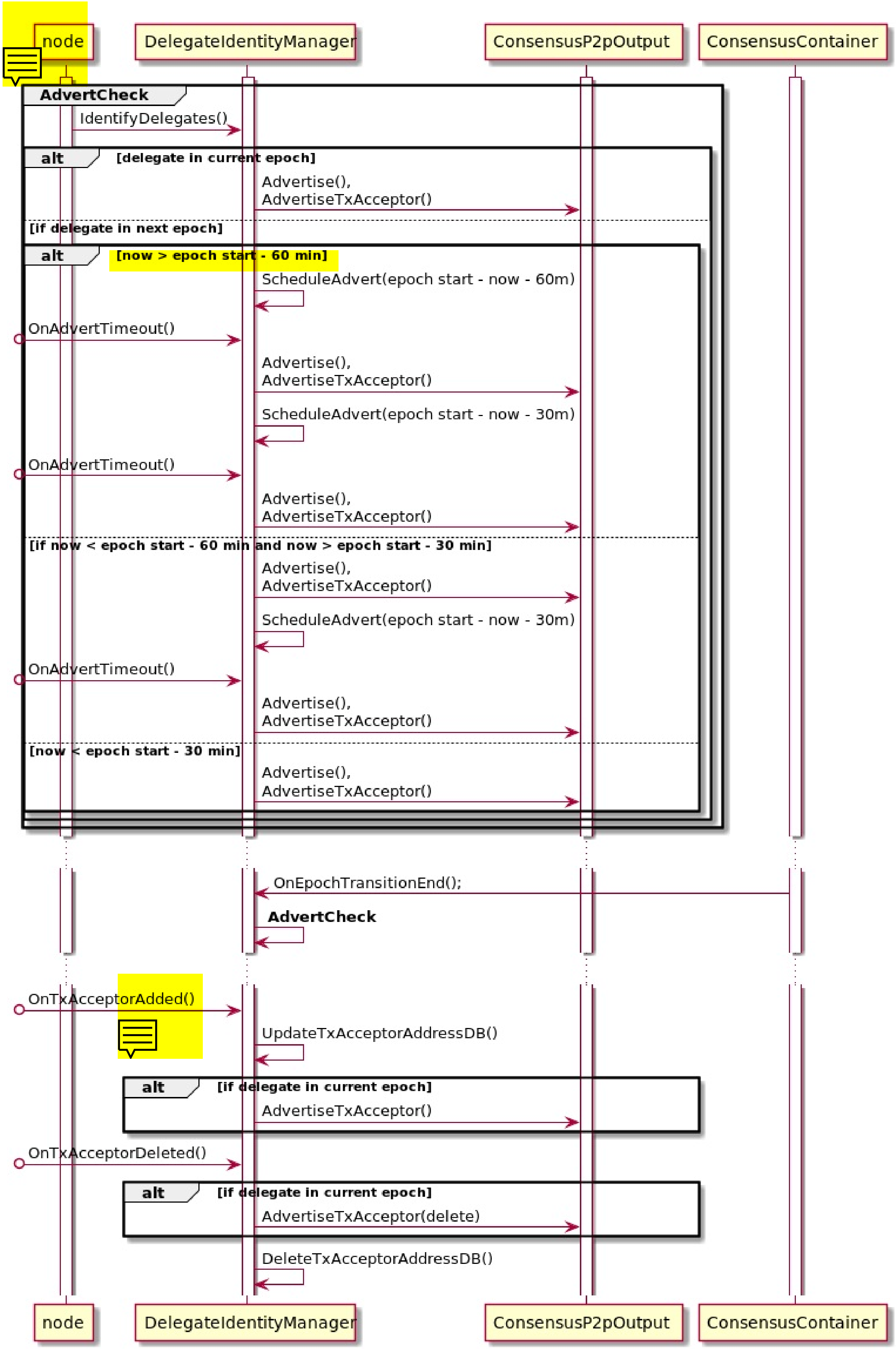
node
DelegateIdentityManager
ConsensusP2pOutput
ConsensusContainer
AdvertCheck
IdentifyDelegates()
alt
[delegate in current epoch]
Advertise(),
AdvertiseTxAcceptor()
[if delegate in next epoch]
alt
[now > epoch start - 60 min]
ScheduleAdvert(epoch start - now - 60m)
OnAdvertTimeout()
Advertise(),
AdvertiseTxAcceptor()
ScheduleAdvert(epoch start - now - 30m)
OnAdvertTimeout()
Advertise(),
AdvertiseTxAcceptor()
[if now < epoch start - 60 min and now > epoch start - 30 min]
Advertise(),
AdvertiseTxAcceptor()
ScheduleAdvert(epoch start - now - 30m)
OnAdvertTimeout()
Advertise(),
AdvertiseTxAcceptor()
[now < epoch start - 30 min]
Advertise(),
AdvertiseTxAcceptor()
OnEpochTransitionEnd();
AdvertCheck
OnTxAcceptorAdded()
UpdateTxAcceptorAddressDB()
alt
[if delegate in current epoch]
AdvertiseTxAcceptor()
OnTxAcceptorDeleted()
alt
[if delegate in current epoch]
AdvertiseTxAcceptor(delete)
DeleteTxAcceptorAddressDB()
node
DelegateIdentityManager
ConsensusP2pOutput
ConsensusContainer

Figure 2. 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 epoch or it is the delegate in the next epoch and current time is less than epoch start - 1 hour (IdentifyDelegates()). If it is the delegate and it doesn't have addresses of all other delegates then the delegate requests all 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. On timeout (OnPeerAddrRequestTimeout()) the delegate repeats the sequence (CheckDelegateAddress). When a node receives a delegate's or tx acceptor address advertisement (OnAddressAdMessage(), OnAddressAdTxAcceptorMessage()), and if the node is the delegate in current or next epoch then it updates the address database. If the node is the delegate in the current epoch then it connects to the received delegate address (ConnectToPeer()). On epoch transition end (OnEpochTransitionEnd()) the node deletes delegate and tx acceptor advertisment messages that are older than the current epoch (DeleteDelegateAddressDB(), DeleteTxAcceptorAddressDB()). Sequence diagram for the address receiving is shown on Figure 3.

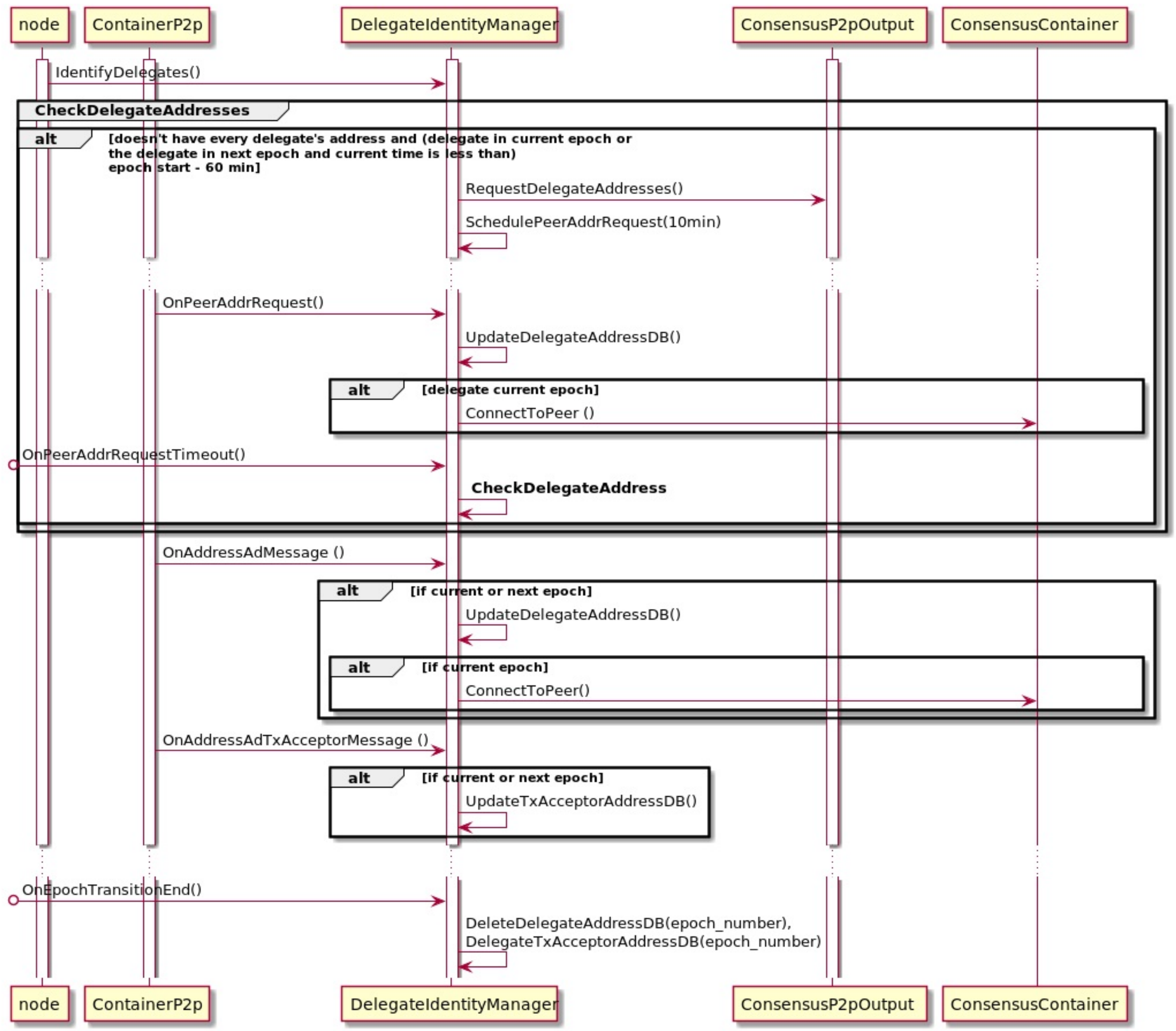
node
ContainerP2p
DelegateIdentityManager
ConsensusP2pOutput
ConsensusContainer
IdentifyDelegates()
CheckDelegateAddresses
alt
[doesn't have every delegate's address and (delegate in current epoch or the delegate in next epoch and current time is less than) epoch start - 60 min]
RequestDelegateAddresses()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

Figure 3.Sequence diagram, address receiving.

# Database

The database id to store the delegate's address is delegate_address_db and is keyed by the delegate's ip, port, and epoch number. The value is blank. The database id to store the delegate's tx acceptor is txacceptor_address_db is keyed by the delegate's id, tx acceptor ip, json port, binary port, and epoch number. The value is blank. The keys serialization is described in the Table 1 and 2.

| Field Name |Size (Byte)| Description | | --- | -------------| ----------------- | | delegate id | 1 | Delegate id| | IP | 39 | ip as the string | Port | 2 | Binary port| | Epoch number | 4 | Binary epoch number|

Table 1. Delegate address key.

| Field Name |Size (Byte)| Description | | --- | -------------| ----------------- | | delegate id | 1 | Delegate id| | IP | 39 | ip as the string | Json port | 2 | Binary formatted port| | Binary port | 2 | Binary formatted port| | Epoch number | 4 | Binary formatted epoch number|

Table 2. Tx Acceptor address key.