TXAcceptor

Overview

TxAcceptor is an instance of a light node that participates in the public P2P network and mitigates the risk of DDoS attacks. TxAcceptor accepts transactions from a client and forwards them to the Delegate. IP’s of public TxAcceptor’s are broadcasted via key-adv. A Delegate could have multiple TxAcceptor’s.

Supported Configurations

In the first iteration of TxAcceptor’s implementation, configuration file is used to announce public IP’s of a delegate and to indicate run-time configuration of TxAcceptor. Currently there are three types of supported configuration:
- Standalone. Accept transactions and forward them via TCP/IP to the delegate. No other functionality is supported by the node.
- Standalone and P2P subsystem. Same as above but in addition P2P subsystem is instantiated and operates on a separate communication channel.
- Integrated with the Delegate. TxAcceptor runs in the same node (same process) as the Delegate and transactions are forwarded via messages (presently a function call).

On startup the core software reads configuration file and determines what objects have to be instantiated to support specified configuration. In the standalone mode only TxAcceptor and optionally P2P are instantiated. Consequently initialization and spanning of a TxAcceptor node in this mode is fast.

Class Diagram

The class diagram is shown of Figure 1.

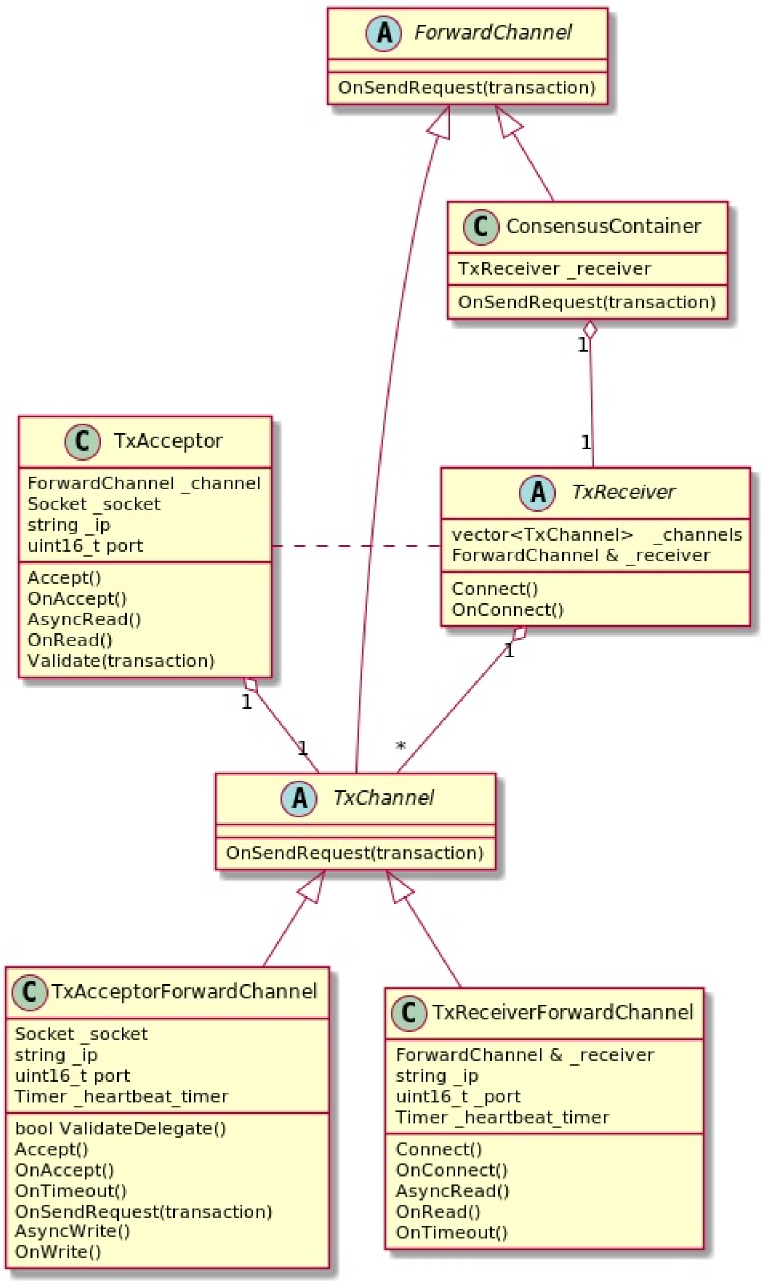

Figure 1. TxAcceptor class diagram.

TxAcceptor functionality is implemented via several classes:
- TxAcceptor accepts client’s connection (Accept(), OnAccept()) and asynchronously reads transaction(s) (AsyncRead(), OnRead()). Received transaction(s) is validated (Validate()) and on success is asynchronously forwarded to the delegate via TxChannel (_channel). The client is informed of failed and successfully forwarded transactions. The client’s connection is closed after forwarding. In standalone mode TxAcceptor and optionally P2P are instantiated without instantiating of logos::node class and with a minimal required system’s support (boost asio). In delegate’s mode TxAcceptor is instantiated and contained within logos::node object. In this mode, TxChannel is represented by ConsensusContainer’s reference.
- TxAcceptorForwardChannel accepts TCP/IP connection from a delegate. The connection is validated with ValidateDelegate() (TBD. Straw-man implementation is signed delegate’s ip) and is rejected for an invalid delegate. The channel forwards transactions with OnSendRequest(). OnSendRequest() internally buffers transactions since boost’s AsyncWrite can not handle

concurrent writes on the same socket. The channel is instantiated in standalone mode and is owned by TxAcceptor. The channel is closed on socket error. New connections are not accepted while the current connection is active (Or should just accept any?). TxAcceptorForwardChannel sends a heartbeat message after inactivity of 40 seconds (_timer, OnTimeout()).
- TxReceiver is instantiated in standalone mode and instantiates TxReceiverForwardChannel(s). It is container for all delegate’s connections. TxReceiver is not instantiated in delegate’s mode.
- TxReceiverForwardChannel is instantiated in standalone mode. It connects to configured TxAcceptor’s (Connect(), OnConnect()), receives transactions (AsyncRead(), OnRead()), and forwards them to consensus handling objects via _receiver. The channel’s socket is closed after inactivity of 60 seconds and a new connections is attempted after 5 seconds.

Messages

There are two types of messages handled by TxAcceptor: Transaction and Heartbeat. Heartbeat message is Transaction message with 0 pay-load.

Message Header

Field Name	Size	Description
Version	1	Logos core protocol version
Message Type	1	Type of message
MPF	2	Multipurpose field
Payload Size	4	Size of the message - size of the header
Single State Block(s)	Total size of all blocks	0 or more serialized Single State Block(s)

TxAcceptor instantiation and transactions forwarding

TxAcceptor instantiation and transaction forwarding is shown on sequence diagram on Figure 2.

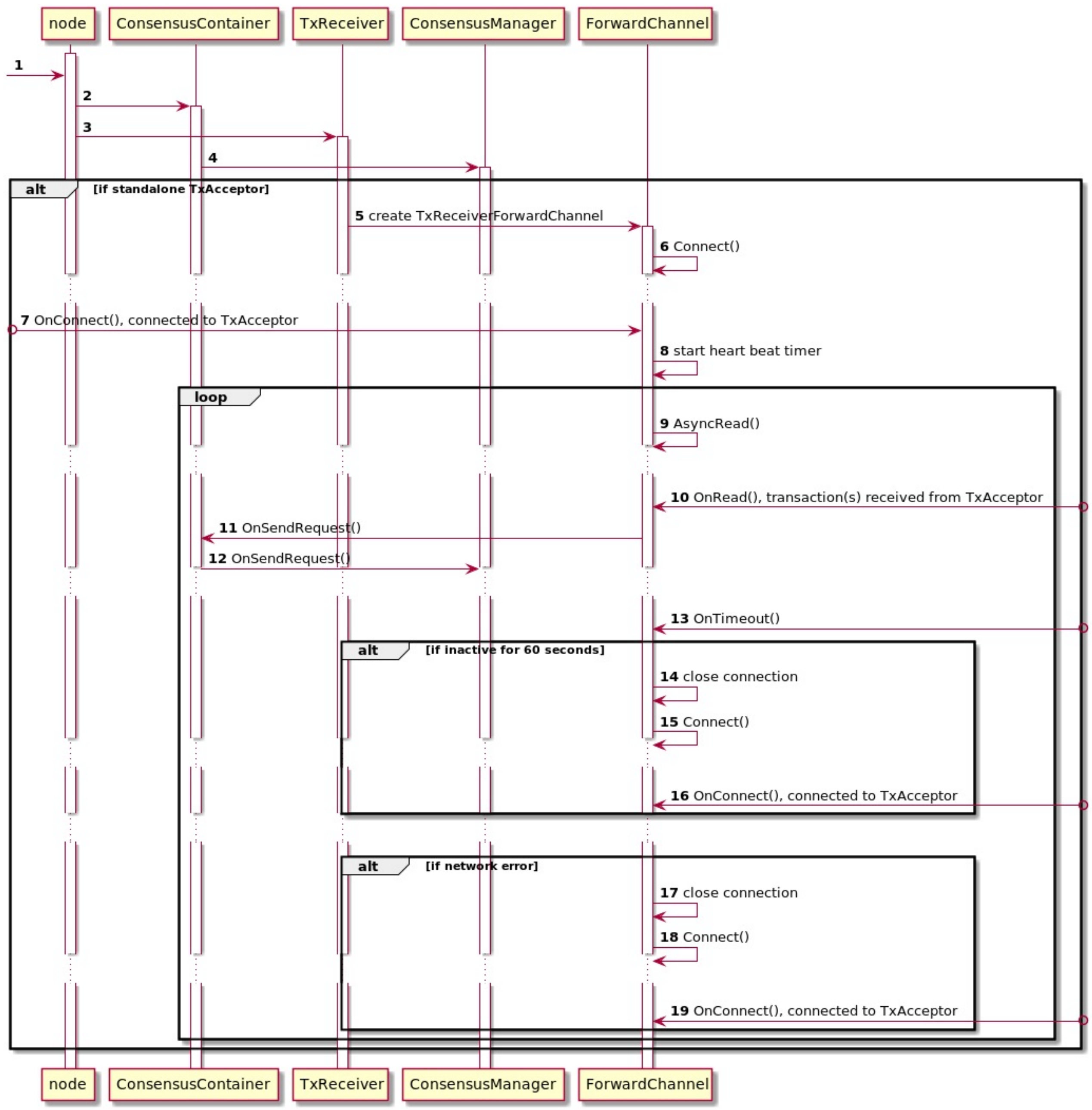
node
ConsensusContainer
TxReceiver
ConsensusManager
ForwardChannel
1
2
3
4
alt [if standalone TxAcceptor]
5 create TxReceiverForwardChannel
6 Connect()
7 OnConnect(), connected to TxAcceptor
8 start heart beat timer
loop
9 AsyncRead()
10 OnRead(), transaction(s) received from TxAcceptor
11 OnSendRequest()
12 OnSendRequest()
13 OnTimeout()
alt [if inactive for 60 seconds]
14 close connection
15 Connect()
16 OnConnect(), connected to TxAcceptor
alt [if network error]
17 close connection
18 Connect()
19 OnConnect(), connected to TxAcceptor

Figure 2.

When running in standalone mode, TxAcceptor class is instantiated outside of the node class. The node class and consequently all other contained within it classes are not instantiated. Boost asio threads and service are instantiated in order to provide asynchronous socket and timer operations. TxAcceptor instantiates TxAcceptorForwardChannel class which starts accepting connection from the delegate. Heartbeat timer is started with the timeout set at 15 seconds.

In delegate mode, TxAcceptor is instantiated by the node class and is passed ConsensusContainer reference, which in this mode represents the ForwardChannel.

In either mode TxAcceptor accepts a client's connection request. Once connected, TxAcceptor reads client's transaction request. Received transactions are validated and valid transactions are forwarded to ForwardChannel object. In the standalone mode ForwardChannel writes transactions to the socket. In the delegate's mode ForwardChannel calls ConsensusContainer's

OnSendRequest().

When Timeout() is received in standalone mode, ForwardChannel sends an empty transaction if the time since the last forwarded transaction exceeds 40 seconds.

If network error is received in standalone mode, ForwardChannel closes the socket and starts accepting connection from the delegate (step 5).

TxReceiver instantiation and transactions receiving

TxReceiver instantiation and transaction receiving is shown on sequence diagram on Figure 3.

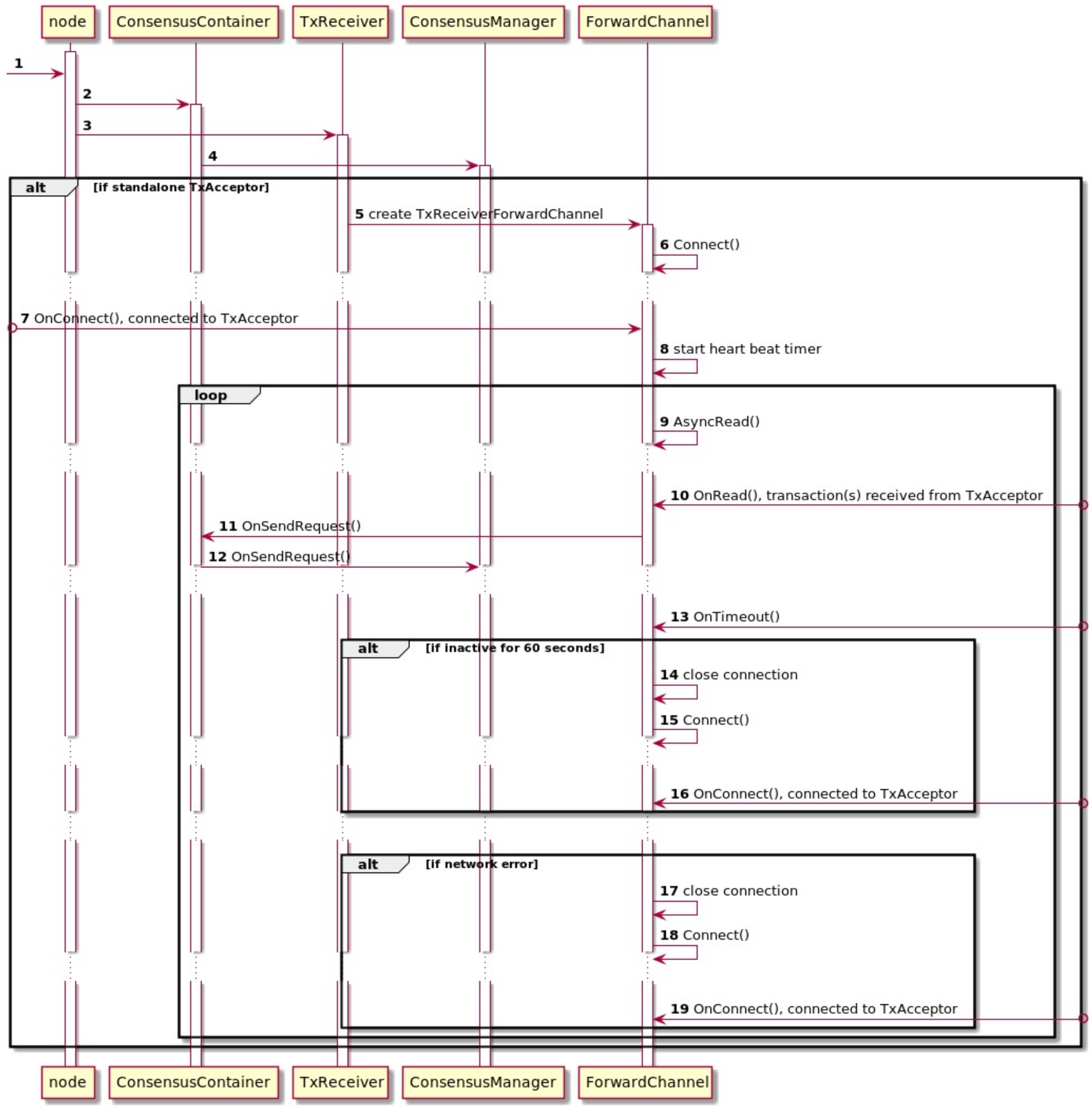

Figure 3.

TxReceiver receiver is always instantiated by the node and is passed ConsensusContainer reference.

In standalone mode, TxReceiver instantiates TxReceiverForwardChannel as ForwardChannel and adds it to the _channels. Once instantiated, ForwardChannel connects to the TxAcceptor. Once connected, the delegate has to authenticate itself to TxAcceptor (future). Currently TxAcceptor accepts any delegate’s connection. After the connection is validated, ForwardChannel asyncronously receives transactions from TxAcceptor. When transaction is received, it is forwarded to consensus via ConsensusContainer (steps 11,12).

In the delegate’s mode, TxReceiver is not instantiated and transactions are forwarded directly to ConsensusContainer. This is described on Figure 2 step 20.

When Timeout() is received in standalone mode, ForwardChannel closes connection if the time since the last received transaction exceeds 60 seconds. ForwardChannel then attempts to reconnect to TxAcceptor. The same sequence is repeated in case of a network error.