Overview

Middleware for forwarding IBC packets.

Asynchronous acknowledgements are utilized for atomic multi-hop packet flows. The acknowledgement will only be written on the chain where the user initiated the packet flow after the forward/multi-hop sequence has completed (success or failure). This means that a user (i.e. an IBC application) only needs to monitor the chain where the initial transfer was sent for the response of the entire process.

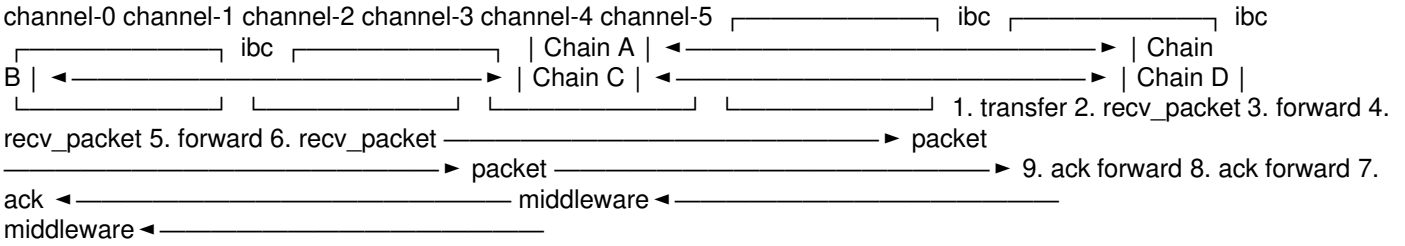
See more info and source code here .

About

The packet-forward-middleware is an IBC middleware module built for Cosmos blockchains utilizing the IBC protocol. A chain which incorporates the packet-forward-middleware is able to route incoming IBC packets from a source chain to a destination chain. As the Cosmos SDK/IBC become commonplace in the blockchain space more and more zones will come online, these new zones joining are noticing a problem: they need to maintain a large amount of infrastructure (archive nodes and relayers for each counterparty chain) to connect with all the chains in the ecosystem, a number that is continuing to increase quickly. Luckly this problem has been anticipated and IBC has been architected to accomodate multi-hop transactions. However, a packet forwarding/routing feature was not in the initial IBC release.

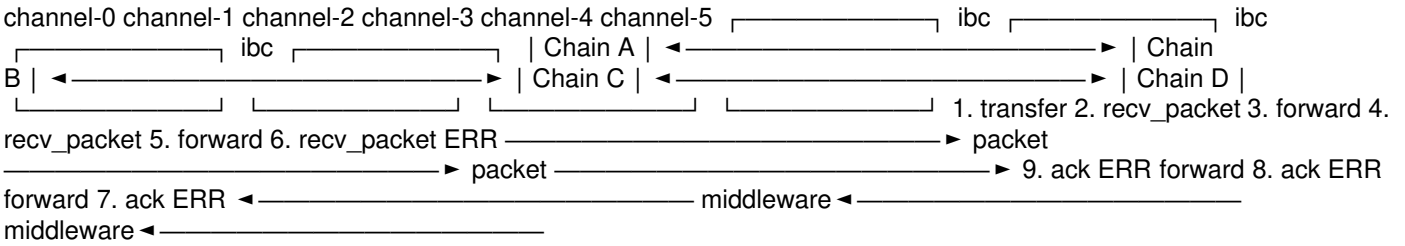
Sequence diagrams

Multi-hop A->B->C->D success

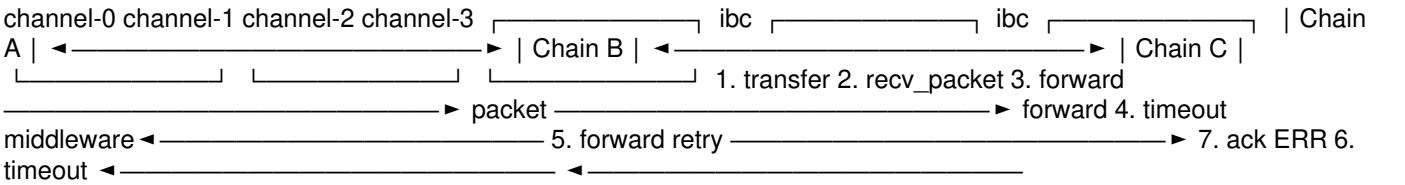


Multi-hop A->B->C->D, C->Drecv_packet

error, refund back to A



Forward A->B->C with 1 retry, max timeouts occurs, refund back to A



Examples

Utilizing the packetmemo field, instructions can be encoded as JSON for multi-hop sequences.

Minimal Example - Chain forward A->B->C

- The packet-forward-middleware integrated on Chain B.
- The packetmemo
- is included inMsgTransfer
- by user on Chain A.

{ "forward": { "receiver": "chain-c-bech32-address", "port": "transfer", "channel": "channel-123" } }

Full Example - Chain forward A->B->C->D with retry on timeout

- The packet-forward-middleware integrated on Chain B and Chain C.
- The packetmemo
- is included inMsgTransfer
- by user on Chain A.
- A packet timeout of 10 minutes and 2 retries is set for both forwards.

In the case of a timeout after 10 minutes for either forward, the packet would be retried up to 2 times, at which case an error ack would be written to issue a refund on the prior chain.

next is thememo to pass for the next transfer hop. Permemo intended usage of a JSON string, it should be either JSON which will be Marshaled retaining key order, or an escaped JSON string which will be passed directly.

next as JSON

{ "forward": { "receiver": "chain-c-bech32-address", "port": "transfer", "channel": "channel-123", "timeout": "10m", "retries": 2, "next": { "forward": { "receiver": "chain-d-bech32-address", "port": "transfer", "channel":"channel-234", "timeout":"10m", "retries": 2 } } } } next as escaped JSON string

{ "forward": { "receiver": "chain-c-bech32-address", "port": "transfer", "channel": "channel-123", "timeout": "10m", "retries": 2, "next": "{\"forward\":{\"receiver\":\"chain-d-bech32-address\",\"port\":\"transfer\",\"channel\":\"channel-234\",\"timeout\":\"10m\",\"retries\":2}}" } }

References

- https://www.mintscan.io/cosmos/proposals/56
- https://github.com/cosmos/ibc-go/pull/373
- https://github.com/strangelove-ventures/governance/blob/master/proposals/2021-09-hub-ibc-router/README.md

Previous Overview Next IBC Relayer