Inter-Blockchain Communication

The Protocol and Cosmos SDK Module

Luke Collins
luke@simply-vc.com.mt

21st February, 2022

What is IBC?

- A *protocol* that allows blockchains to talk to each other

What is IBC?

- A *protocol* that allows blockchains to talk to each other
- **Not** just a Cosmos thing

What is IBC?

- A *protocol* that allows blockchains to talk to each other
- **Not** just a Cosmos thing
- **Not** just for token transfers

What is IBC?

- A *protocol* that allows blockchains to talk to each other
- **Not** just a Cosmos thing
- **Not** just for token transfers
- Analogy: it's like TCP/IP for blockchains

Applications

- Token transfers,
- Atomic swaps,
- Contracts running on multiple chains,
- Layer on which to build object capability system,
- Sharding protocols

Components needed for IBC

- (At least) two chains, A and B
- Modules (smart contracts, packages, etc.) on those chains which can speak IBC
- Off-chain relayers
- Light client for consensus (e.g. SPV for Bitcoin)

Abstraction Hierarchy

- Blockchains establish *connections* between each other using *ports* (similar conceptually to the role of ports in TCP/IP)

  A connection essentially means that two (light) chains implemented clients to verify each other's consensus

- Modules establish *channels* between each other

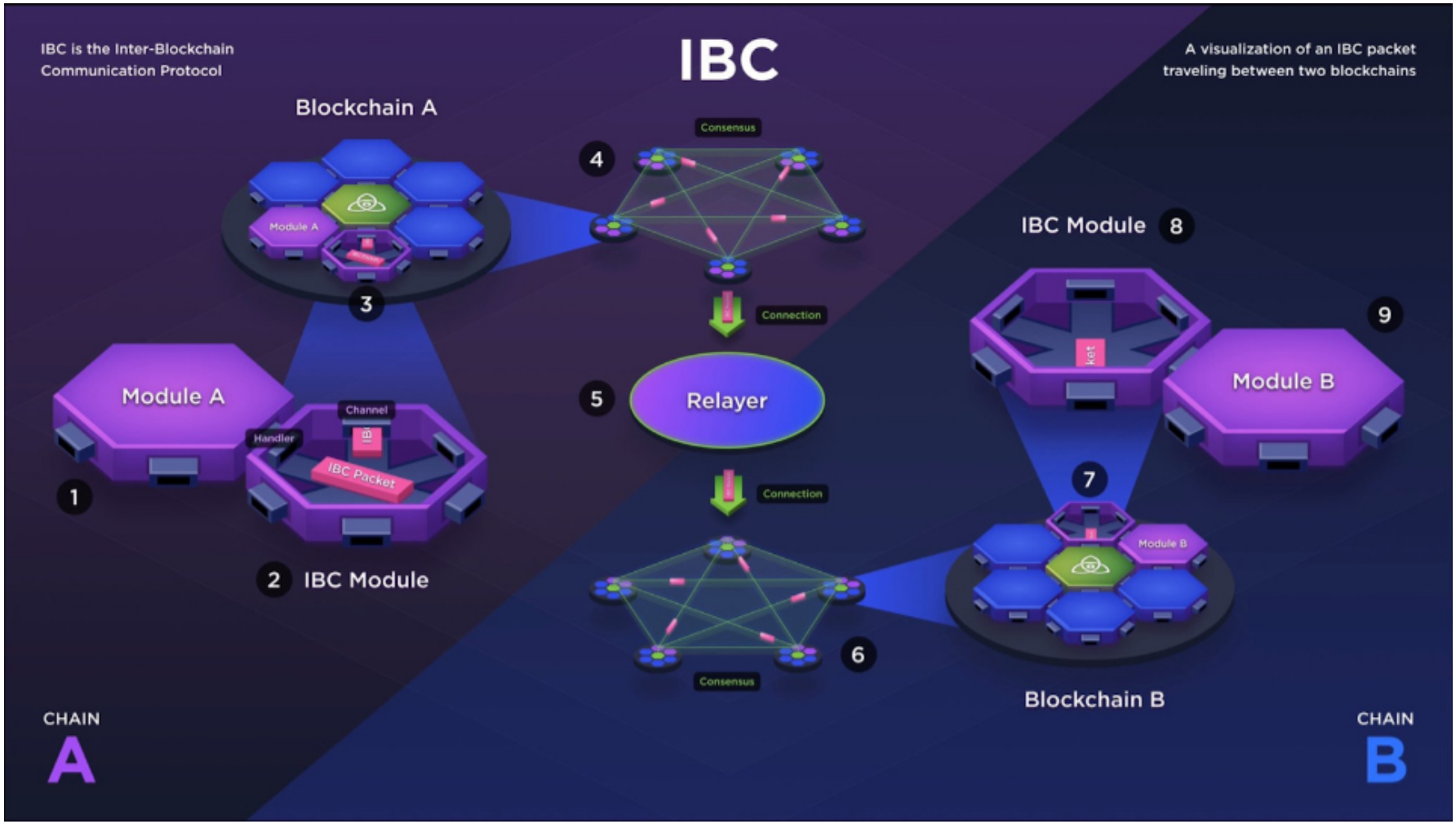
  They are ordered exactly (technically using DAGS)

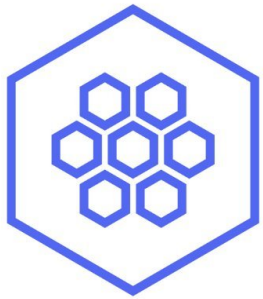
"TAO" of IBC

Transport, authentication and ordering of packets sent from blockchain A to blockchain B

IBC doesn't care about the content of the packages, only the TA(O).

IBC At a Glance





PART II:

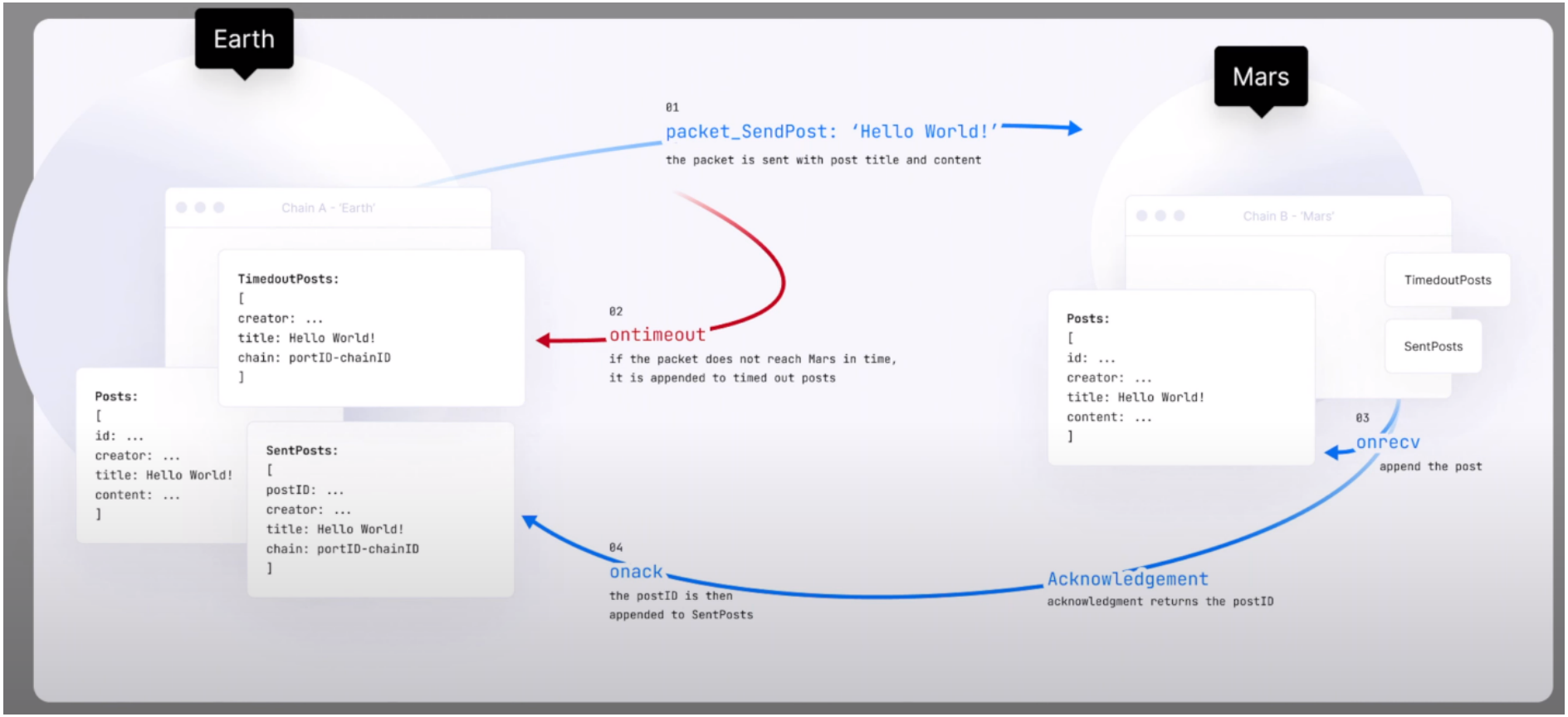
The Cøsmos SDK Module

The IBC Module

Integrating IBC into an SDK-based application is surprisingly easy.

- Add required modules to the `module.BasicManager`
- Define additional `Keeper` fields for the new modules on the `App` type
- Add the module's `StoreKeys` and initialize their `Keepers`
- Set up corresponding routers and routes for the `ibc` and `evidence` modules
- Add the modules to the module `Manager`
- Add modules to `Begin/EndBlockers` and `InitGenesis`
- Update the module `SimulationManager` to enable simulations

Example Application



Packet specification

Defining Packets

```
// Custom packet data defined in application module
type CustomPacketData struct {
    // Custom fields ...
}

EncodePacketData(packetData CustomPacketData) []byte {
    // encode packetData to bytes
}

DecodePacketData(encoded []byte) (CustomPacketData) {
    // decode from bytes to packet data
}
```

Sending

```
// Sending custom application packet data
data := EncodePacketData(customPacketData)
packet.Data = data
IBCChannelKeeper.SendPacket(ctx, packet)
```

Receiving

```
// Receiving custom application packet data (in OnRecvPacket)
packetData := DecodePacketData(packet.Data)
// handle received custom packet data
```

Any questions?