AIP[1]-013: P2P Protocol Version Handling

platfowner, minsulee2, liayoo 2021-03-06

Goals

- Handle p2p protocol version compatibility so that blockchain nodes can be upgraded minimizing service discontinuity

Problem Definition

When we need to upgrade blockchain node's version to an incompatible one, the following issues need to be addressed:

- Consensus process (block proposal, voting sharing)
- Chain segment sharing
- Transaction sharing

Requirements

We can define different types of incompatibility:

- Consensus process incompatibility
- Data sharing incompatibility (e.g. chain segment, transaction)

So it's required to handle each of the above cases properly.

Proposed Design

Key Ideas

- Define three levels of compatibility/incompatibility:

Incompatibility level	Consensus	Data sharing	Note
L0: Compatible	Compatible	Two-way compatible	
L1: Consensus-incompatible	Incompatible	Two-way compatible	

[1] AI Network Improvement Proposal. Visit https://docs.ainetwork.ai for the full list.

L2: Data-incompatibile	Incompatible	One-way compatible	

- Define versions of consensus protocol (CONSENSUS_PROTOCOL_VERSION) and data protocol (DATA_PROTOCOL_VERSION) separate from the package version
  - **Major version changes** mean two versions are incompatible while **minor or patch version changes** mean compatible. For example, 1.0.0 and 2.0.0 are incompatible versions while 1.0.0 and 1.2.3 are compatible with each other.
- The protocol versions are attached to each P2P message so that the receiver can check the compatibility and handle them properly
- Introduce **shadowing** for faster switching to higher version

Design Details

Consensus-Incompatibility Handling

Message type	Action for too-low versioned message	Action for too-high versioned message	Notes
PROPOSE	Drop message	Drop message	
VOTE	Drop message	Drop message	
(Unknown type)	Drop message	Drop message	

Data-Incompatibility Handling

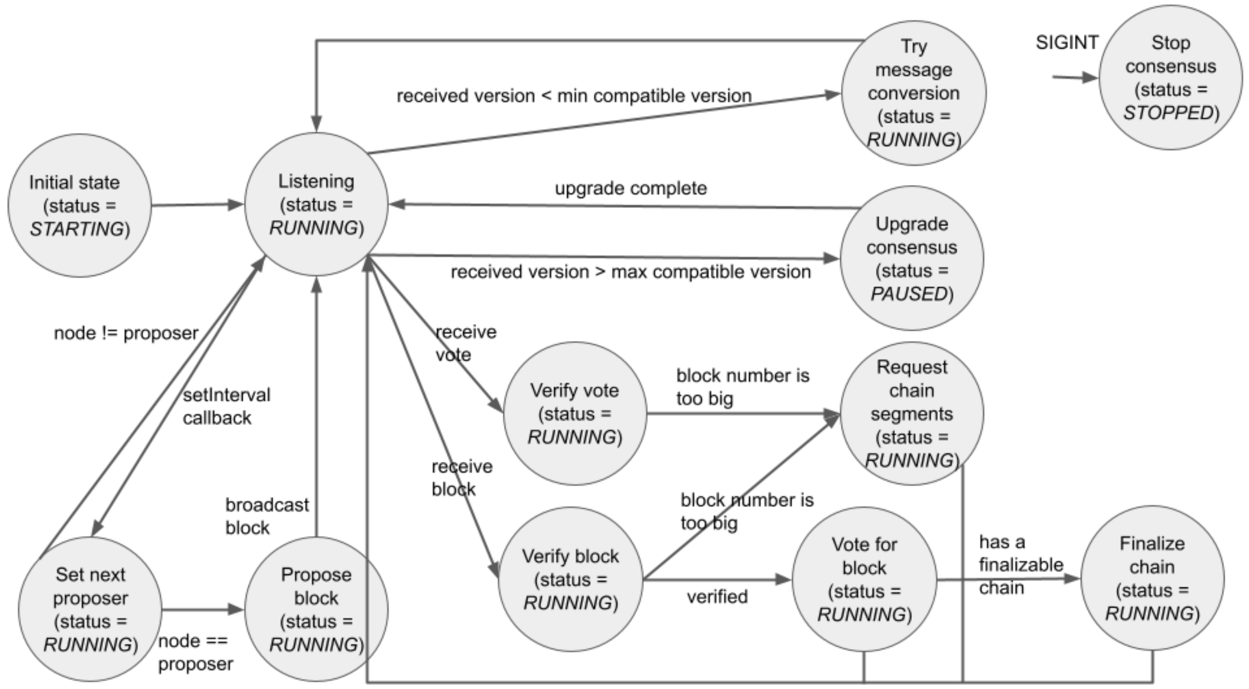
Message type	Action for too-low versioned message	Action for too-high versioned message	Notes
ADDRESS_REQEUST	Convert message	Convert message	
ADDRESS_RESPONSE	Convert message	Convert message	
CHAIN_SEGMENT_REQUEST	Respond with INCOMPATIBLE_VERSION	Respond normally	
CHAIN_SEGMENT_RESPONSE	Convert message	Drop message	
TRANSACTION	Convert message	Drop message	
CONSENSUS	Convert message	Drop message	
(Unknown type)	Drop message	Drop message	

Version Upgrade Phases

To upgrade the blockchain cluster's protocol version, the nodes will be upgraded one by one going through the following phases (with 5-node example):

Phase	Lower version nodes	Higher version nodes	Note
1. Normal	5	0	
2. Shrinking	4	1	
3. Discontinuous	3 ~ 2	2 ~ 3	
4. Expanding	1	4	
5. Normal	0	5	

Consensus State Transition


Try message conversion (status = RUNNING)
SIGINT
Stop consensus (status = STOPPED)
received version < min compatible version
Initial state (status = STARTING)
Listening (status = RUNNING)
upgrade complete
Upgrade consensus (status = PAUSED)
received version > max compatible version
node != proposer
receive vote
Verify vote (status = RUNNING)
block number is too big
Request chain segments (status = RUNNING)
setInterval callback
receive block
broadcast block
block number is too big
Set next proposer (status = RUNNING)
Propose block (status = RUNNING)
Verify block (status = RUNNING)
verified
Vote for block (status = RUNNING)
has a finalizable chain
Finalize chain (status = RUNNING)
node == proposer

Milestones

We have the following milestones, which can be achieved in parallel:

- Step 1: Successful version upgrade with data-incompatibility
- Step 2: Successful version upgrade with consensus-incompatibility

Conclusion

- Classified the incompatible cases into three levels: compatible, data-incompatible, consensus-incompatible
- Provided a design of p2p protocol version compatibility handling so that blockchain nodes can be upgraded minimizing service discontinuation

Further Extension

Shadowing

Let’s call the original node *light node* and its version-upgraded node *shadow node*. Shadowing is done in the following steps:

- Step 1: Shadow node started and it sync’s all blocks with the light node
- Step 2: When the shadow node is ready to serve, the light node stops serving
- Step 3: The shadow node starts serving
- Step 4: The light node terminates

When the shadow node is in syncing mode:

- A trusted channel is established between the two nodes for faster syncing
- The shadow node joins the P2P network but remains in a passive mode, i.e., do not actively participate in data sharing or consensus
- The light node and the shadow node shares node keys

Document History

Date	Who	Change	Notes
2021-03-06	platfowner	Initial draft	
2021-03-08	platfowner, minsulee2	Synced with minsulee2	
2021-03-15	minsulee2, liayoo, cshcomcom, platfowner	Internal review	
2021-04-06	platfowner, liayoo, minsulee2	Major revision with the separate protocol versions ideas	
2021-05-12	platfowner	Github IDs Link to full list	

2021-05-12	platfowner	Published	