**Algorithm 1**: Implement and initialize modules on ni

**Input**: The map of supported Tx types and pool name it allocated TPM, pool names PNs, pool filter names PFNs, pool selector names PSNs, pool orderer names PONs, pool mid-processer names PMNs

**Dictionary** Prov\_PC\_FILTER = {“Fn”: Filtern, …}, Prov\_PC\_SELECT = {“Sn”: Selectn, …}, Prov\_PC\_ORDER = {“On”: Ordern, …}, Prov\_PC\_MIDPROC = {“Mn”: MidProcn, …}

**Function** Filtern(Tx): // Input Tx, Output ifPass (true/false)

According your rules to filter Tx, for example whether Tx’s signature is true

**Function** Selectn(Txs): // Input Txs, Output TxsSelected

According your rules to choose Tx subset from Txs

**Function** Ordern(Txs): // Input Txs, Output TxsOrdered

According your rules to order Txs

**Function** MidProcn(Txs, sandbox): // Input Txs, Output TxsProcessed

**foreach** Tx in Txs **do** execute Tx in sandbox to obtain Tx.RS and Tx.WS, let Tx.Executed=true, then write Tx.WS to sandbox

ni.Sorter = TPM

ni.PCers = CreatePCers()

**Function** CreatePCers():

**foreach** PN in PNs **do**

PCer.Filter = prov\_PC\_FILTER[PFNs[Idx]]

PCer.Selector = prov\_PC\_SELECT[PSNs[Idx]]

PCer.Orderer = prov\_PC\_ORDER[PONs[Idx]]

PCer.MidProcesser = prov\_PC\_MIDPROC[PMNs[Idx]]

PCers[PN] = PCer

**return** PCers

**Algorithm 2**: Programmable consensus workflow Order and Execution at node ni

**Upon** reception of Tx **parallel** **do**

PCer = ni.PCers[ni.Sorter[Tx.Type]]

**if** PCer.Filter(Tx) == true **then**

Append Tx to PCer.Pool.PendingTxs

**foreach** PCer in ni.PCers **parallel** **do**

**Upon** PCer’s block generation condition is triggered **do** // START strategy, may be waiting timeout or pool is overflow

// programmable consensus modules’ using are inserted into consensus protocol flow

TxsSelected = PCer.Selector(PCer.Pool.PendingTxs)

TxsOrdered = PCer.Orderer(TxsSelected)

TxsProcessed = PCer.MidProcesser(TxsOrdered, sandbox) // sandbox based on local world state LWS is cleared when LWS is updated to newest world state

Blockn.Txs = TxsProcessed

Blockn.BlockID = Hash(Blockn) // Blockn.PrevBlockHash is not set

Append Blockn to ni.BlockPool // BlockPool receives blocks from several PCers

其他可以解释的，关于共识的内容(ni signs BlockID and broadcast <BlockID, SIGi>, meanwhile receives <BlockID, SIGx> from other nodes then put them to Blockn.Sigs when the number of SIG of BlockID meets condition)

**Algorithm 3**: Programmable consensus phase Validation at node ni

All nodes fetch block B from BlockPool in a deterministic select method

**foreach** Tx in B.Txs **do** DetectConflict(Tx, sandbox) // sandbox based on world state

check B’s signatures of nodes and then set B.PrevBlockHash to newest block’s hash

commit B to blockchain and update world state

**Function** DetectConflict(Tx, sandbox):

**foreach** r in Tx.RS **do**

**if** r.value != sandbox[r.key] **then** return

**foreach** w in Tx.WS **do** sandbox[w.key] = w.value