All signatures are tagged by message type.

We're assuming node-to-node messages are always in order.

With every message, send our current timestamp. This serves as a promise that we won't introduce any new transactions with a timestamp earlier than that timestamp.

VerifiableBroadcast(msg):

- Broadcast(`|vbc|`: msg)
- For every other node: Receive(`|vbc-ok|`, message)
- Broadcast(`|vbc-allok|`: msg, all `|vbc-ok|` signatures received)

HandleVerifiableBroadcast(Handler):

- Receive(`|vbc|` -> msg)
- Send(`|vbc-ok|`, signed(msg))
- Receive(`|vbc-allok|` -> msg, all signatures)
- Verify all signatures => every other node saw the same message
- Handler(msg)

IntroduceTransaction(x):

- If x.name in set of my currently introducing transactions or others' currently introducing transactions:
- Return Fail
- Else:
- t <- current time
- VerifiableBroadcast(`|intro|`, t, x)
- For every other node: Receive(`|intro-response|`)
  - If any fail: set return value to Fail
- VerifiableBroadcast(`|canpublish|`, x, all signed OKs/Fails)

HandleIntroduction():

- Receive `|intro|` -> t, x
- If x.name in set of my currently introducing transactions or others' currently introducing transactions:
- Send('|intro-response|', signed(OK, x))
- Else:
- Send(`|intro-response|`, signed(Fail, x))

HandleCanPublish(x, signatures):

- Verify all signatures & make sure they're all "OK"s
- (Remember x)

When at some point I have a transaction x in my currently OK'd transactions whose timestamp t <= min of (min of timestamps of all others' currently introducing transactions, min over when we last heard from each server)