# IPv8 Simulator Guide

February 5, 2023

## 1 A27 - Fundamentals and Design of Blockchain-based Systems

In this course you will be designing your own blockchain using the IPv8 peer to peer networking library. The project template provided to you, supplies you with all the tools necessary to perform the lab. The project is structured as follows:

```
asci-a27-blockhain/
|-- bami/                 // contains the IPv8 simulator
|-- |-- ...
|
|
|-- simulation/
|   |-- common/           // contains building block for your solution
|   |   |-- ...
|   |   |-- ...
|   |
|   |-- example/
|   |-- |-- example.py    // contains a simulation example.
|
```

```
## PingPong Community
```

IPv8 relies on network overlays: virtual networks that are built on top of existing physical networks. The PingPong simulation makes use of such an overlay. Next, we'll explain how this code works.

[ ]:
```python
import os
from asyncio import ensure_future, get_event_loop

from ipv8.community import Community
from ipv8.configuration import ConfigBuilder
from ipv8.lazy_community import lazy_wrapper
from ipv8.messaging.lazy_payload import VariablePayload, vp_compile

from simulation.common.settings import SimulationSettings
from simulation.common.simulation import SimulatedCommunityMixin, BamiSimulation
from simulation.common.utils import time_mark, connected_topology
```

After the required imports, we define our network overlay `PingPongCommunity` — which IPv8 refers to as a Community. We extend from the IPv8 `Community` class and define our message types and handlers.

```
[ ]: @vp_compile
     class PingMessage(VariablePayload):
         msg_id = 1

     @vp_compile
     class PongMessage(VariablePayload):
         msg_id = 2


     class PingPongCommunity(Community):
         """
         This basic community sends ping messages to other known peers every two␣
      ↪seconds.
         """
         community_id = os.urandom(20)

         def __init__(self, my_peer, endpoint, network):
             super().__init__(my_peer, endpoint, network)
             self.add_message_handler(1, self.on_ping_message)
             self.add_message_handler(2, self.on_pong_message)

         def started(self):
             self.register_task("send_ping", self.send_ping, interval=2.0, delay=0)

         def send_ping(self):
             self.logger.info("  <t=%.1f> peer %s sending ping", get_event_loop().
      ↪time(), self.my_peer.address)
             for peer in self.network.verified_peers:
                 self.ez_send(peer, PingMessage())

         @lazy_wrapper(PingMessage)
         def on_ping_message(self, peer, payload):
             self.logger.info("  <t=%.1f> peer %s received ping", get_event_loop().
      ↪time(), self.my_peer.address)
             self.logger.info("  <t=%.1f> peer %s sending pong", get_event_loop().
      ↪time(), self.my_peer.address)
             self.ez_send(peer, PongMessage())

         @lazy_wrapper(PongMessage)
         def on_pong_message(self, peer, payload):
             self.logger.info("  <t=%.1f> peer %s received pong", get_event_loop().
      ↪time(), self.my_peer.address)
```

Next, we setup the BAMI Simulator and add our `PingPong` overlay to the simulator IPv8 nodes.

```
[ ]: class BasicPingPongSimulation(BamiSimulation):

         def get_ipv8_builder(self, peer_id: int) -> ConfigBuilder:
             builder = super().get_ipv8_builder(peer_id)
             builder.add_overlay("PingPongCommunity", "my peer", [], [], {},␣
      ↪[('started',)])
             return builder
```

We also use the a mixin to handle delays.

```
[ ]: class SimulatedPingPong(SimulatedCommunityMixin, PingPongCommunity):
         send_ping = time_mark(PingPongCommunity.send_ping)
         on_ping_message = time_mark(PingPongCommunity.on_ping_message)
```

Finally, we define the parameters for our simulation and run it for 10 seconds.

```
[ ]: # We use a discrete event loop to enable quick simulations.
     if __name__ == "__main__":
         settings = SimulationSettings()
         settings.peers = 6
         settings.duration = 10
         settings.indefinite = True
         settings.topology = connected_topology(settings.peers)
         settings.community_map = {'PingPongCommunity': SimulatedPingPong}

         simulation = BasicPingPongSimulation(settings)
         ensure_future(simulation.run())
         simulation.loop.run_forever()
```