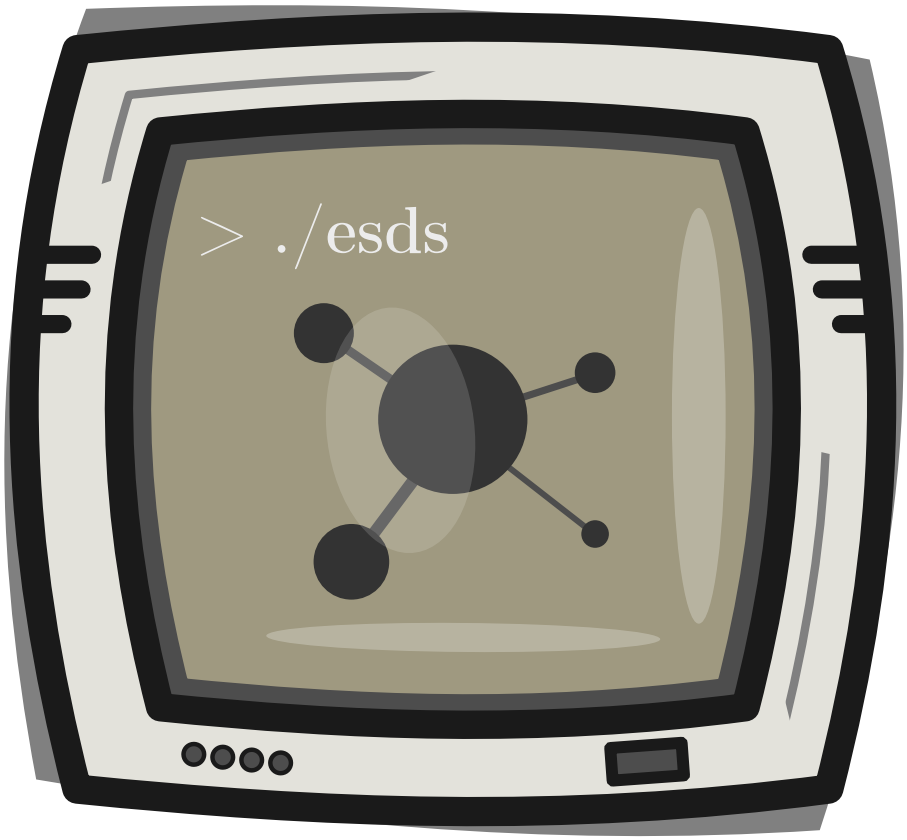
User Manual

— ESDS v0.0.1 —

September 13, 2022



ESDS an Extensible Simulator for Distributed Systems

Written by Loic Guegan and Issam Raïs

1 Architecture of ESDS

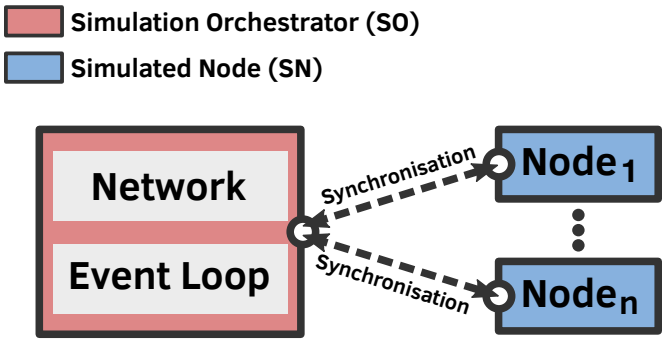


Figure 1: Simulation architecture used by ESDS

ESDS simulator comprises two major components: 1) The Simulation Orchestrator (SO) 2) The Simulated Nodes (SN). This architecture is depicted in Figure 1. The SO is the main process in charge of implementing the simulation main loop. It instantiates the network (e.g bandwidths and latencies), collects and processes the events (e.g communications,turn on/off). On the other hand, nodes are threads that implement the node behaviors.

2 Running your first simulation

To run a simulation, at least 2 files are required: 1) a platform file 2) a node implementation source code. The platform file defines the simulated network platform (network links and performances etc.) and sets various simulation parameters. The node implementation source code provides the logic of the simulated nodes.

2.1 Platform file

Platform files are written in YAML and contains 3 sections namely: 1) *general* 2) *nodes* 3) *interfaces*. The *general* section is optional but all the other sections must be present. Here is an example of a simple platform file to simulate 2 wireless nodes:

assets/platform.yaml

```yaml
general:
    interferences: on # Turns on interferences

nodes:
    count: 2
    implementations:
        - all node.py
    arguments: {
      "0": "sender",
      "1": "receiver"
    }

interfaces:
    wlan0:
```

```
        type: "wireless"
        links:
            - all 50kbps 0s all # All nodes are reachable by each other
        txperfs:
            - all 50kbps 0s
```

2.2 Node implementation file

Nodes implementations are written using python. Here is the implementation of the node mentioned in the last `platform.yaml` file:

assets/node.py

```
def execute(api):
    role=api.args # "sender" or "receiver" cf. platform.yaml
    if role == "sender":
        api.send("wlan0","MY MESSAGE",10,None)
    else:
        api.receive("wlan0")
```

2.3 Execution

To execute our first simulation, the following command should be executed from the same folder that contains `platform.yaml` and `node.py`:

```
> esds run platform.yaml
```

Here is the output of the simulation:

```
[t=0.000,src=n0] Send 10 bytes on wlan0
[t=0.002,src=n1] Receive 10 bytes on wlan0
[t=0.002,src=esds] Simulation ends
```

In this case, simulation tooks $0.002s$ and 10 bytes were sent on the wlan0 interface from node 0 (src=n0) to node 1 (src=n1).

2.4 Custom orchestrator instantiation

Instead of using a `platform.yaml` file, it is possible to instantiate manually the esds orchestrator. To do so, you need to implement that procedure in a python file. Here is and example that performs the exact same simulation presented in Section 2.3 but with a custom instantiation of the orchestrator:

assets/orchestrator.py

```
#!/usr/bin/env python

import esds
import numpy as np
```

```
n=2 # 2 nodes
B=np.full((n,n),50*1000) # Bandwith+txperfs 5bps
L=np.full((n,n),0) # Latency 0s

s=esds.Simulator({"wlan0":{"bandwidth":B, "latency":L, "is_wired":False}})

##### Instantiate nodes with their implementation
s.create_node("node",args="sender") # Use node.py for the first node with "sender" as argument
s.create_node("node",args="receiver") # Now the second node

##### Run the simulation
s.run()
```

Next we can run the simulation:

```
> ./orchestrator.py
```