Boosting Simulation Performance with Python

Eran Friedman

COMMONSENSE ROBOTICS

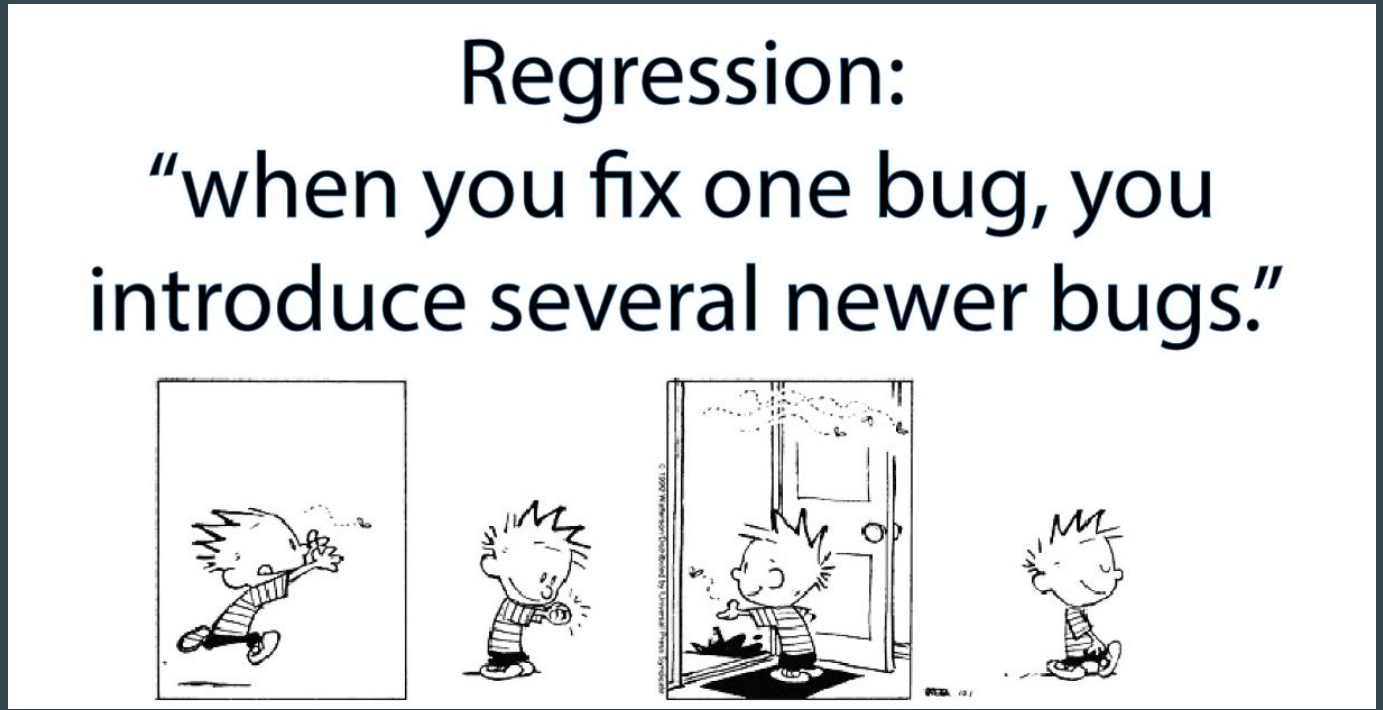
Outline

- Importance of simulations
- Simulation architecture
- SimPy library
- Implementation and challenges
- Distributed simulation


1024

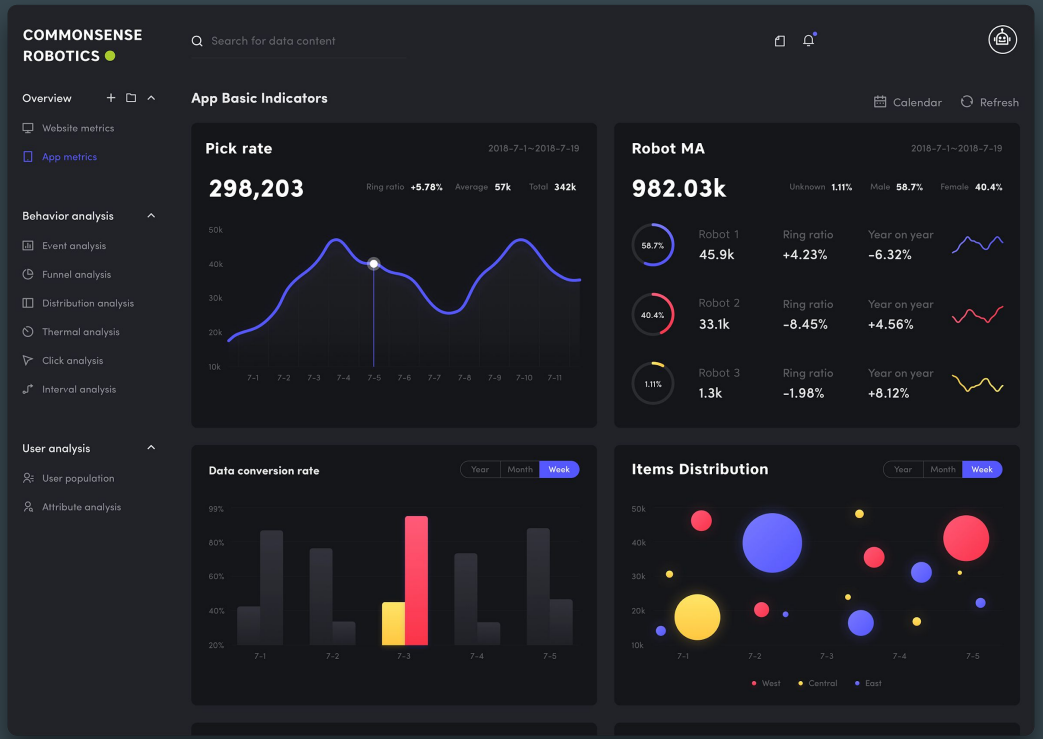
Importance of Simulations

Automated regression tests



Importance of Simulations

Analyze performance & compare algorithms



Importance of Simulations

Run in the cloud

Importance of Simulations

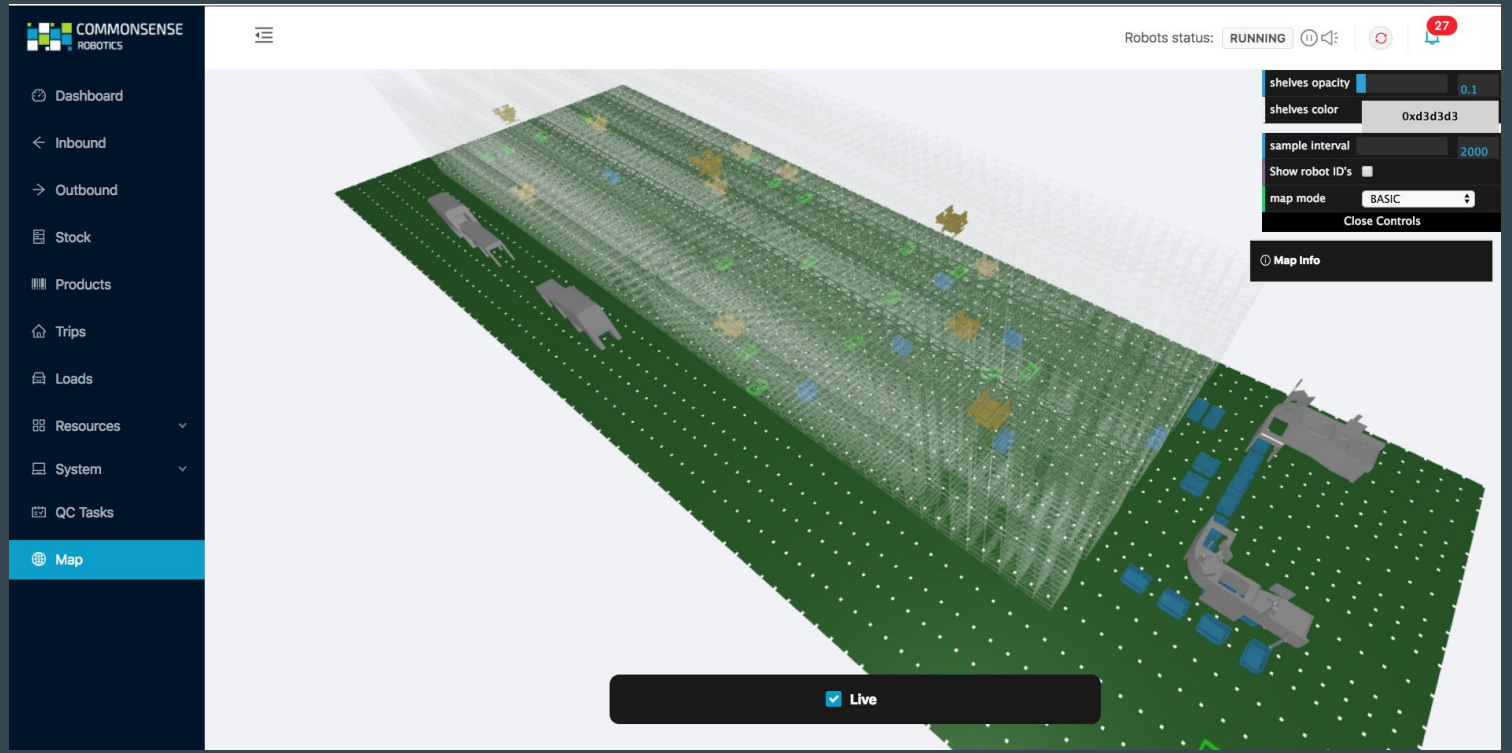
Verify warehouse layout

Importance of Simulations

Inject failures & improve robustness

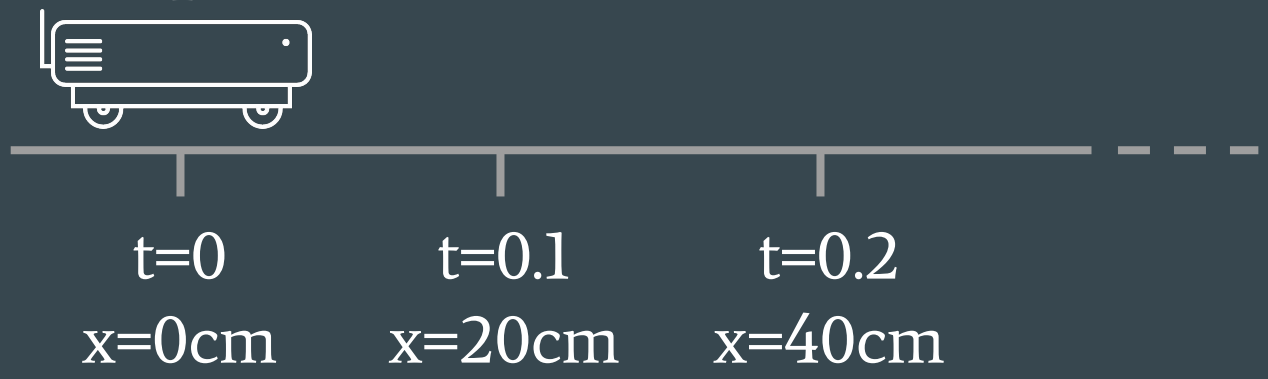
Importance of Simulations

Simulate a large facility



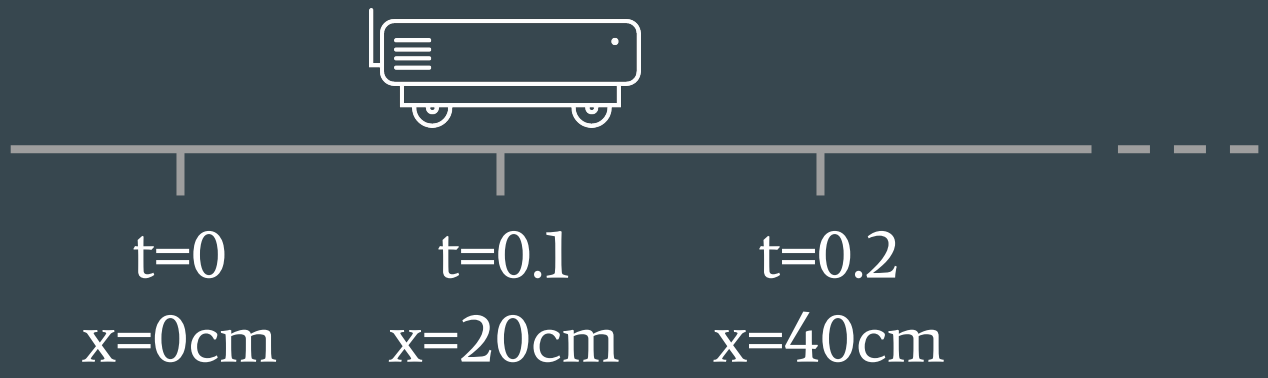
Discrete-Event Simulation (DES)

- Operations are modeled as sequence of events
- Simulation jumps to the next event
- Simulation maintains its own clock
- Example: 2 m/s, 10 time-ticks/second



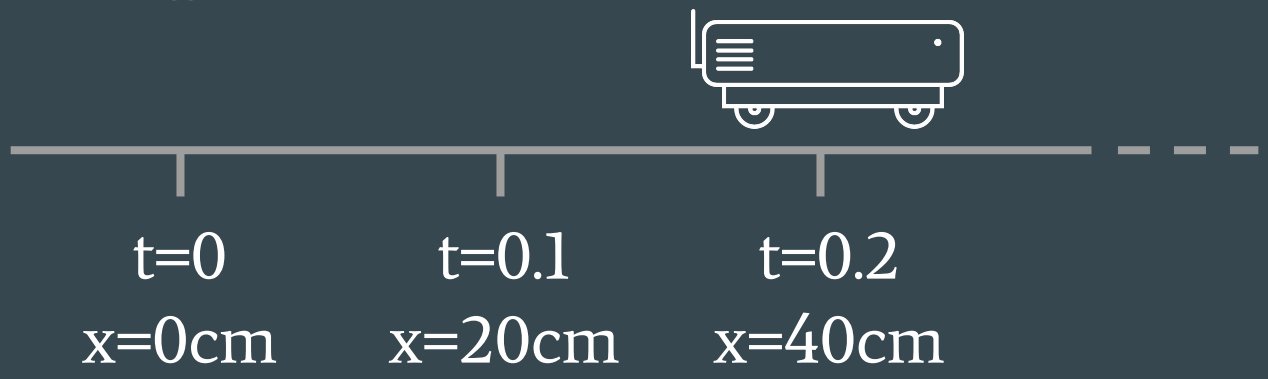
Discrete-Event Simulation (DES)

- Operations are modeled as sequence of events
- Simulation jumps to the next event
- Simulation maintains its own clock
- Example: 2 m/s, 10 time-ticks/second



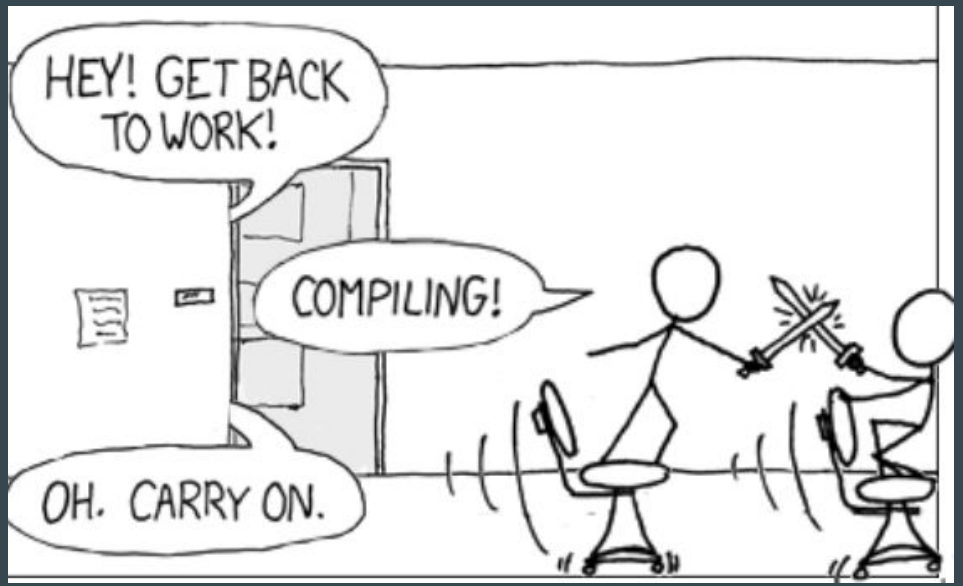
Discrete-Event Simulation (DES)

- Operations are modeled as sequence of events
- Simulation jumps to the next event
- Simulation maintains its own clock
- Example: 2 m/s, 10 time-ticks/second



Benefits of DES

- Accelerates development time and faster CI
- Realistic and deterministic simulation



Benefits of DES

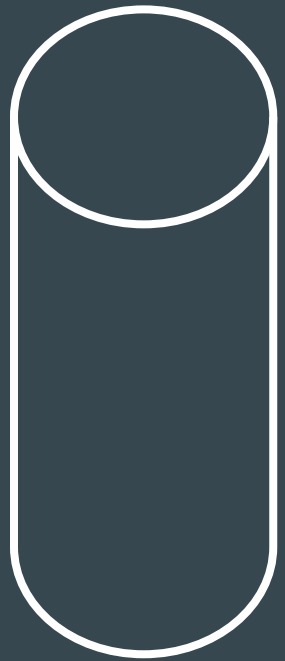
- Feedback on code efficiency
- Simulate any date and time of the day (no panic before ‘Y2K’ bug)

SimPy Library

- Discrete-event simulation (DES) framework
- Created in 2002
- MIT license
- Pure Python
- No dependencies
- Stable release - 3.0.11

SimPy Overview

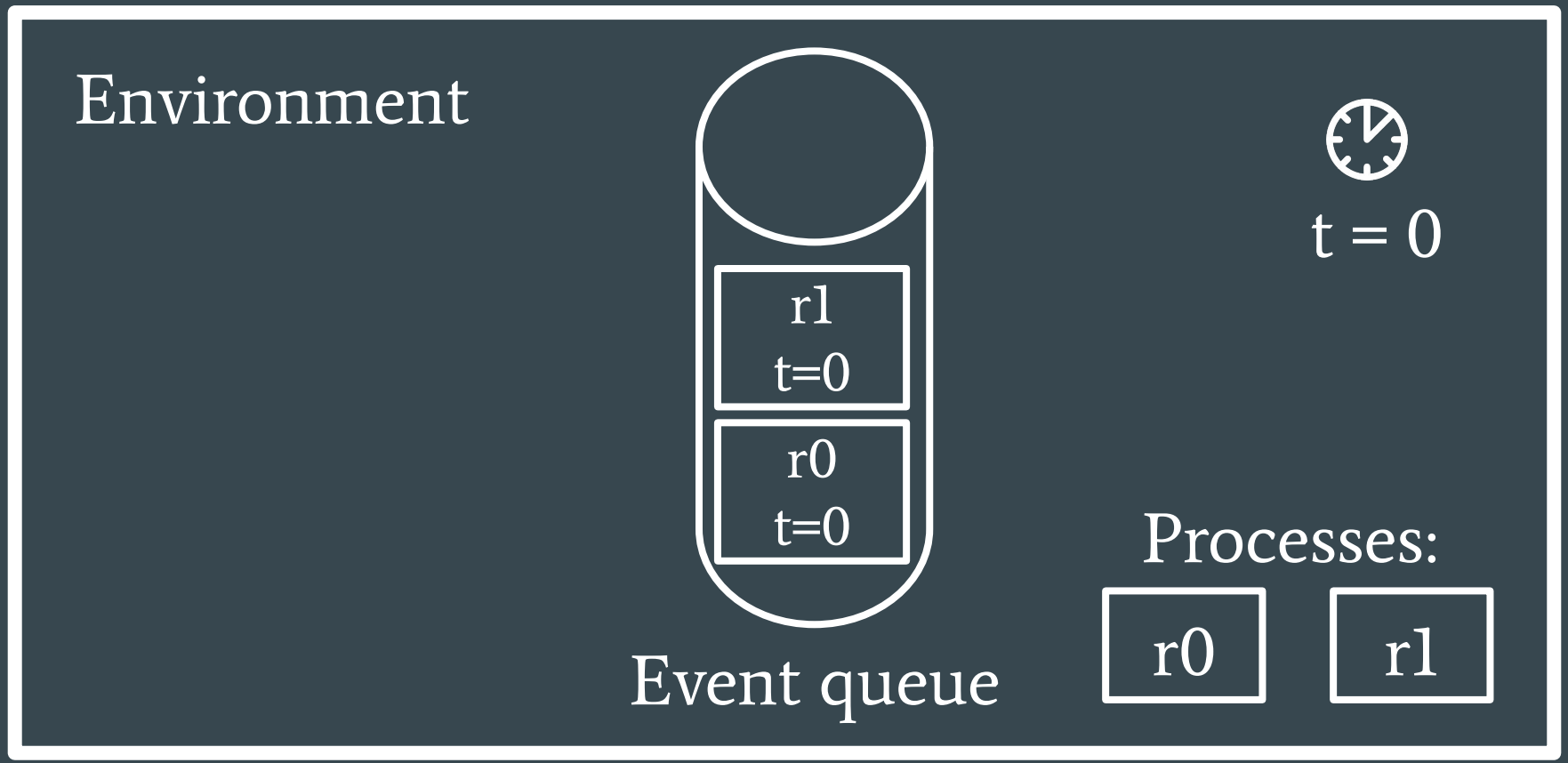
Environment



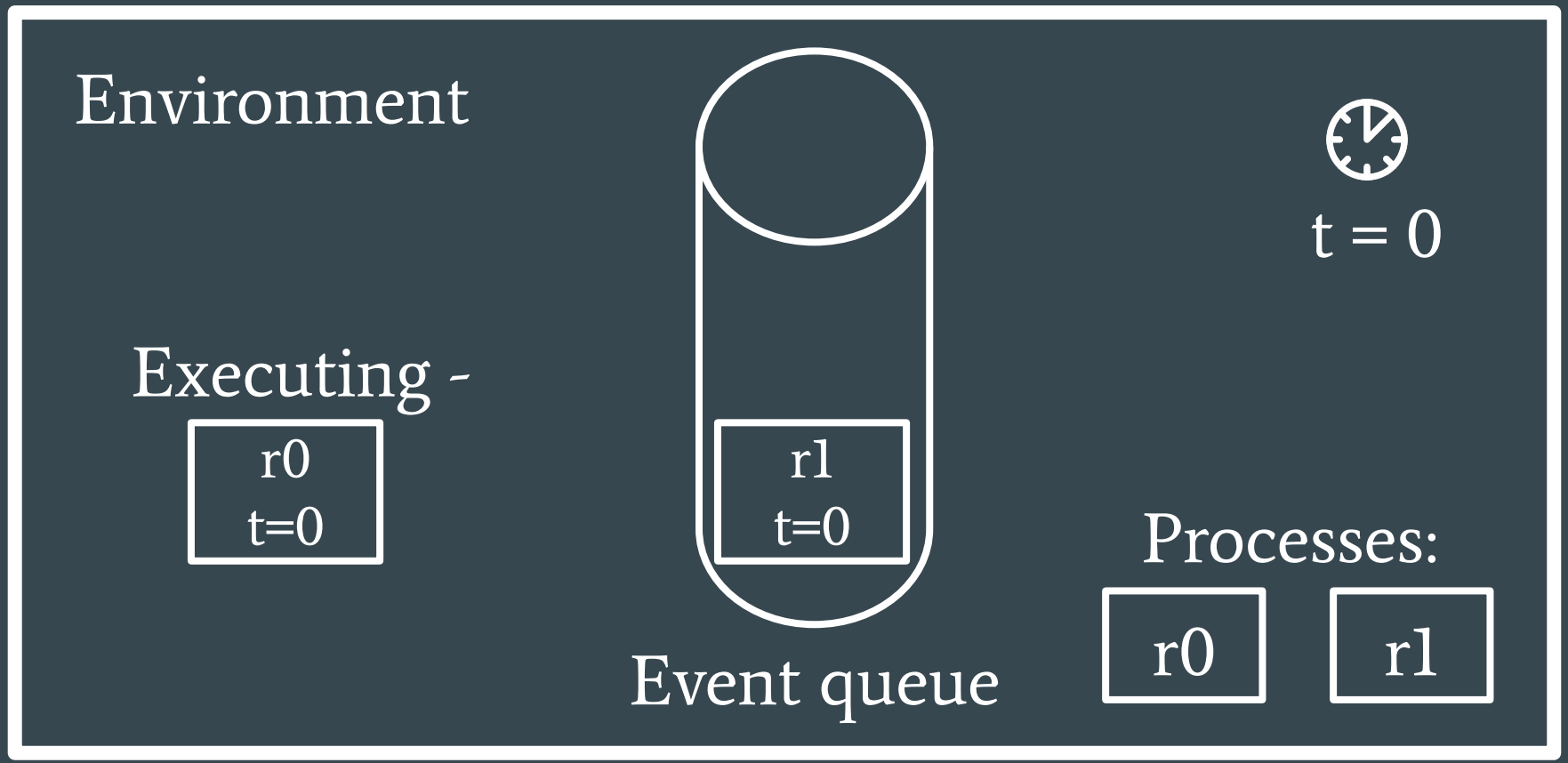
Event queue

t = 0

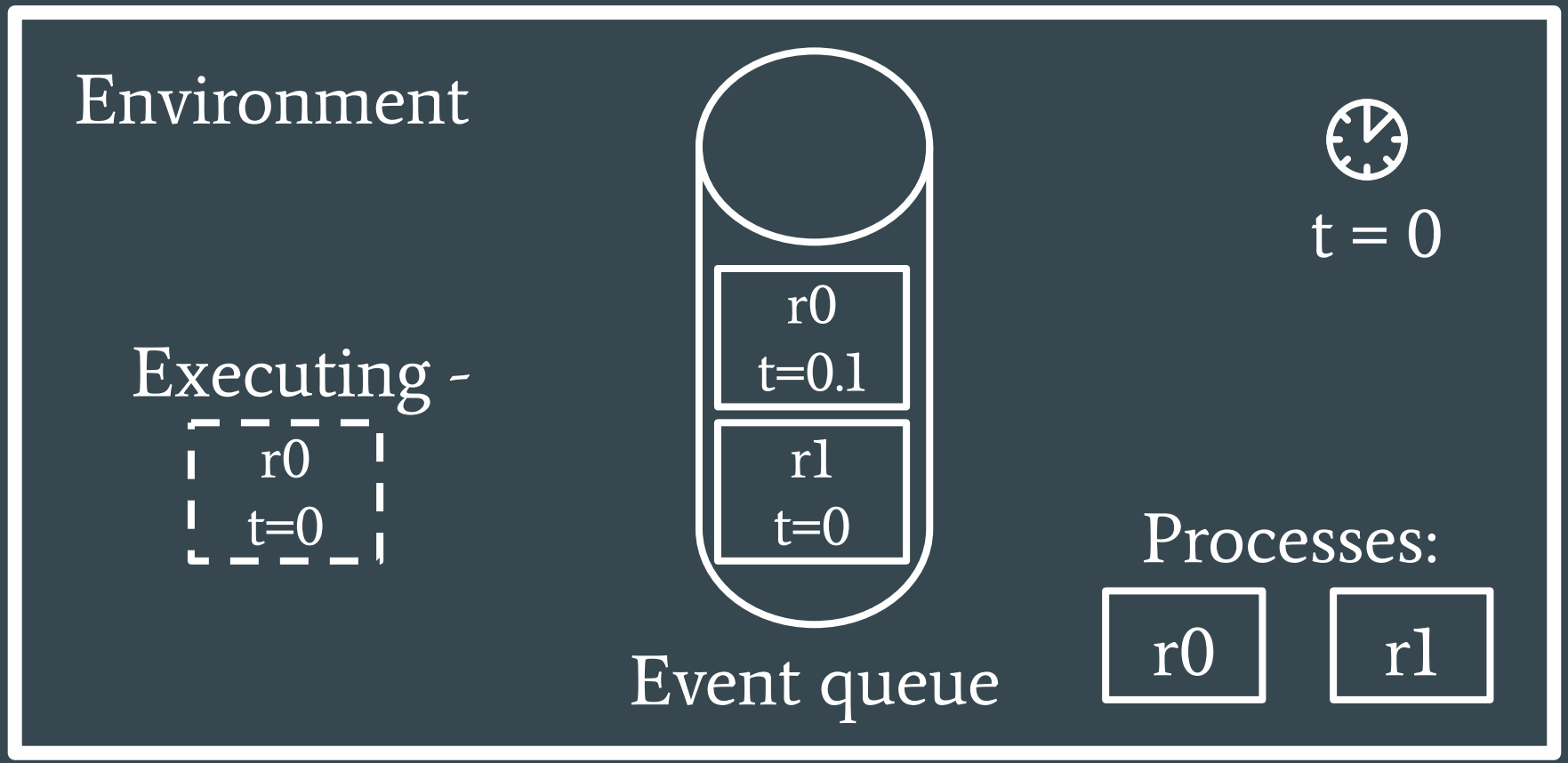
SimPy Overview



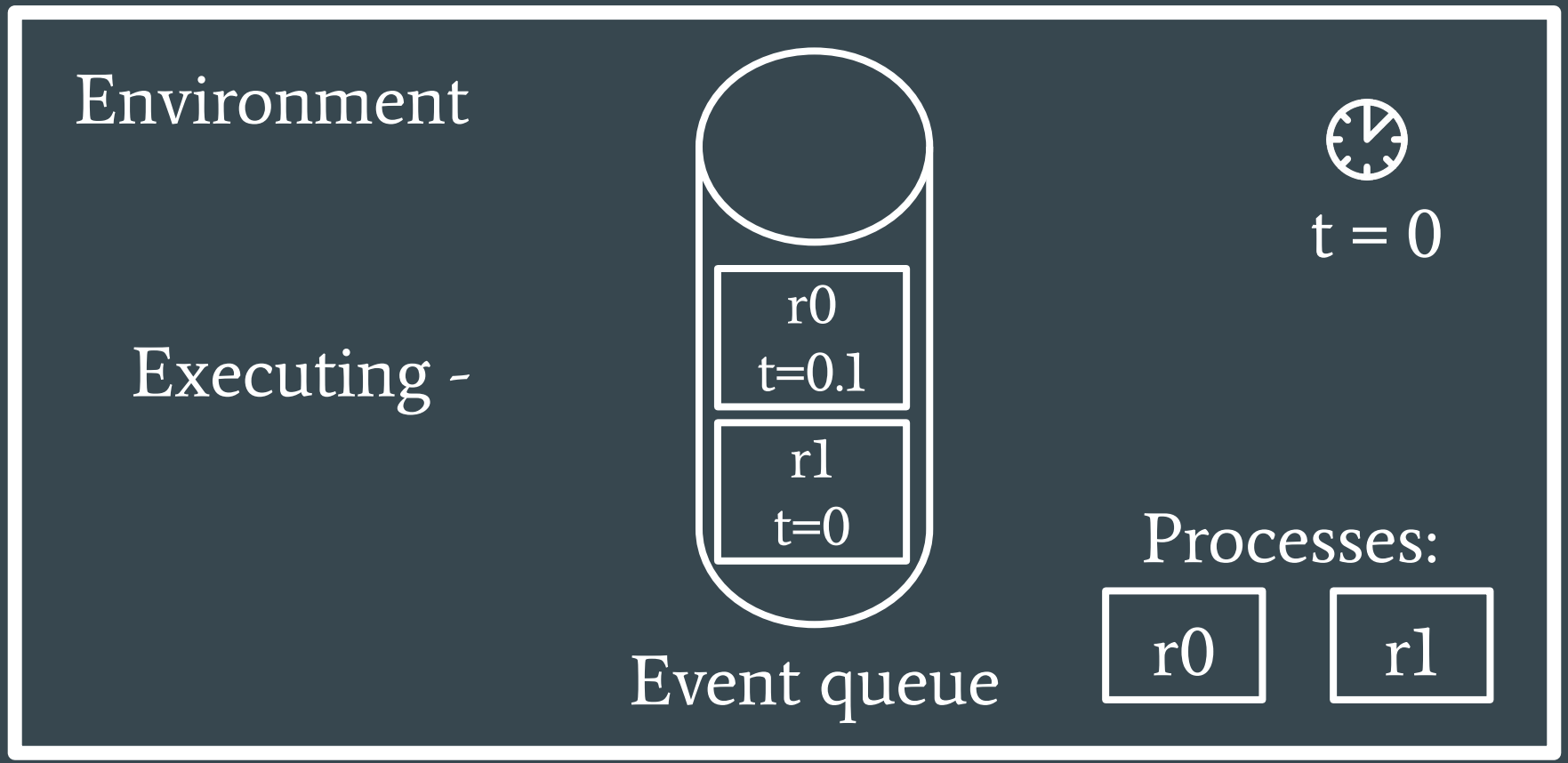
SimPy Overview



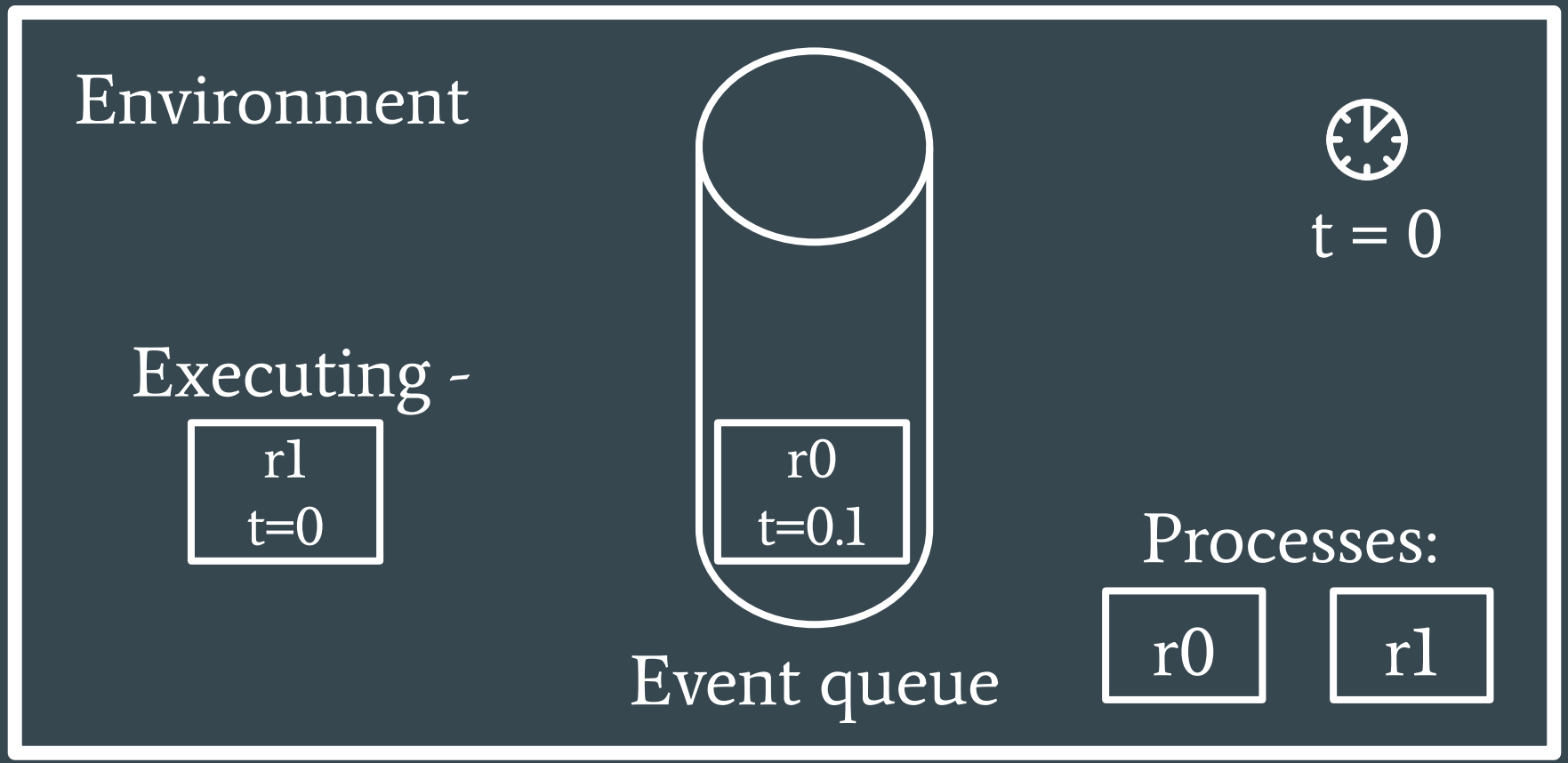
SimPy Overview



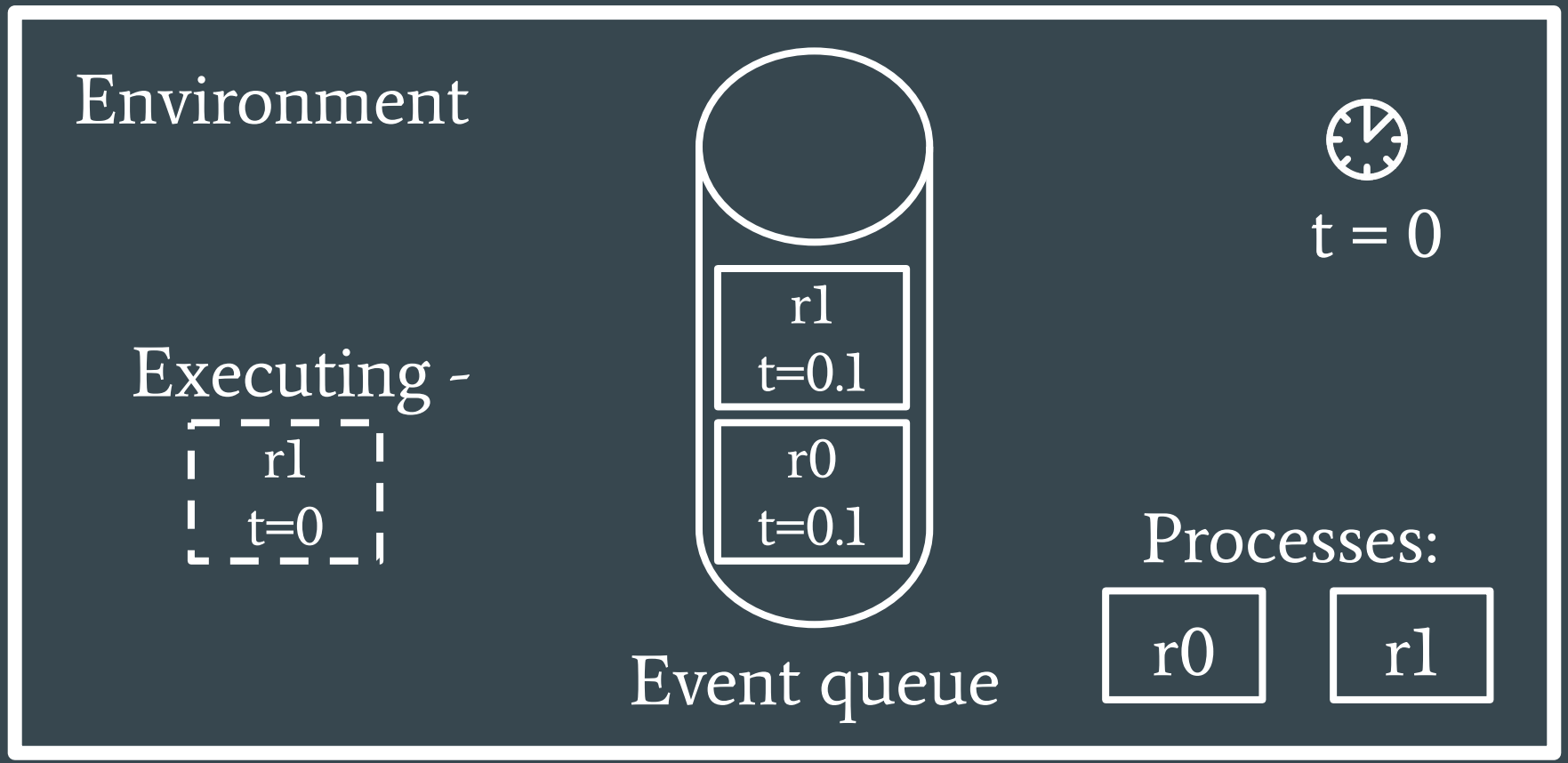
SimPy Overview



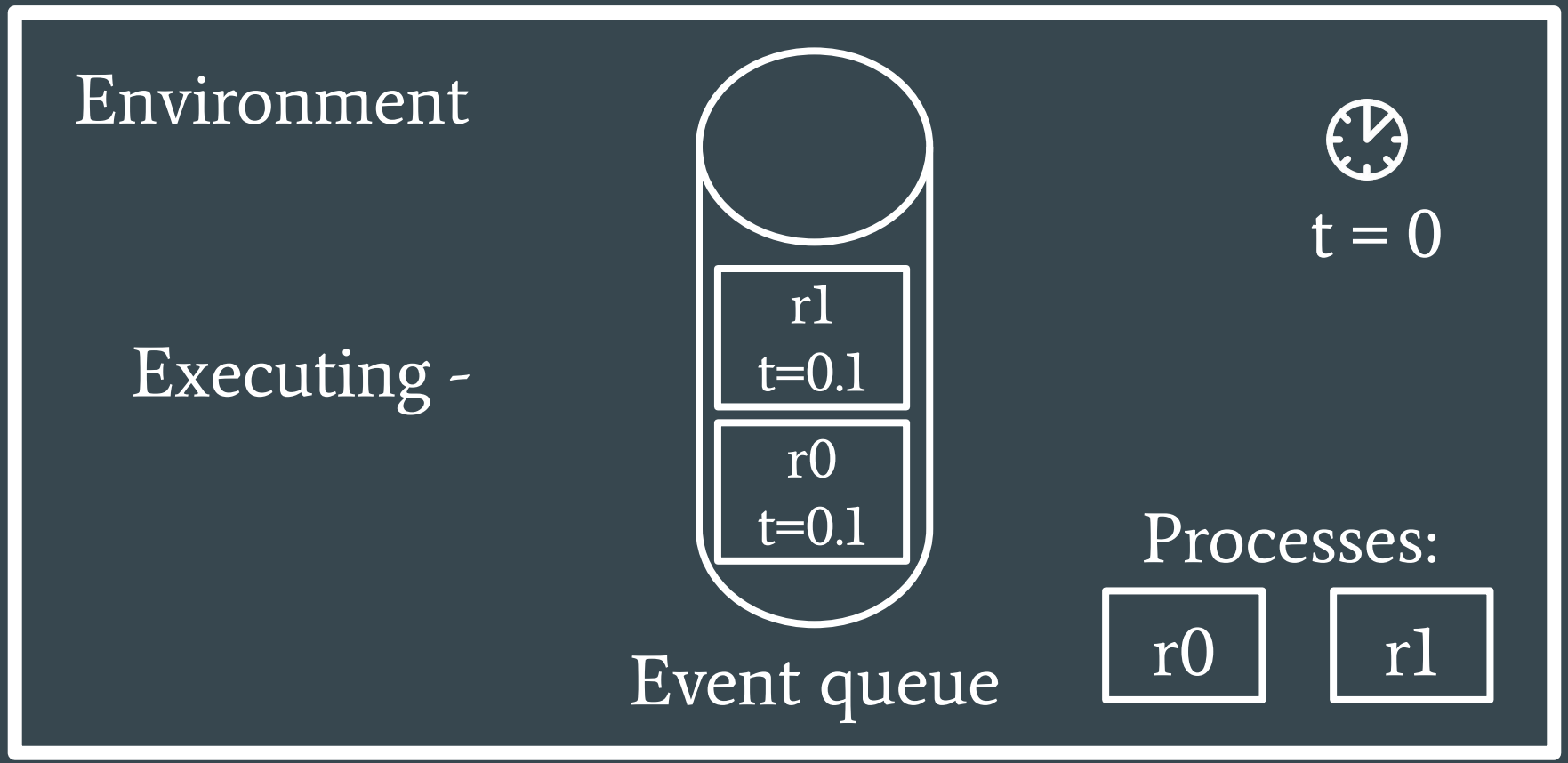
SimPy Overview



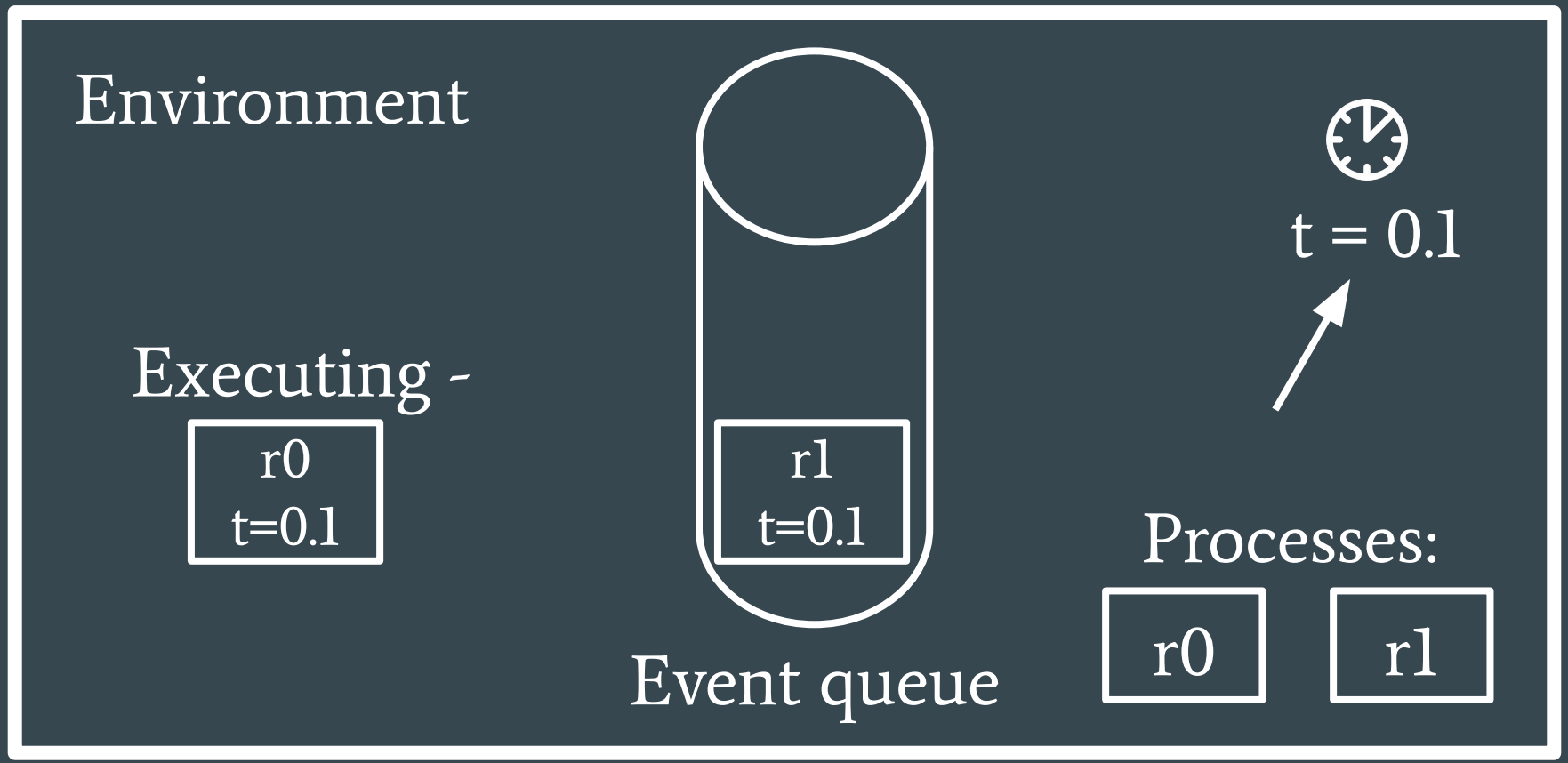
SimPy Overview



SimPy Overview



SimPy Overview

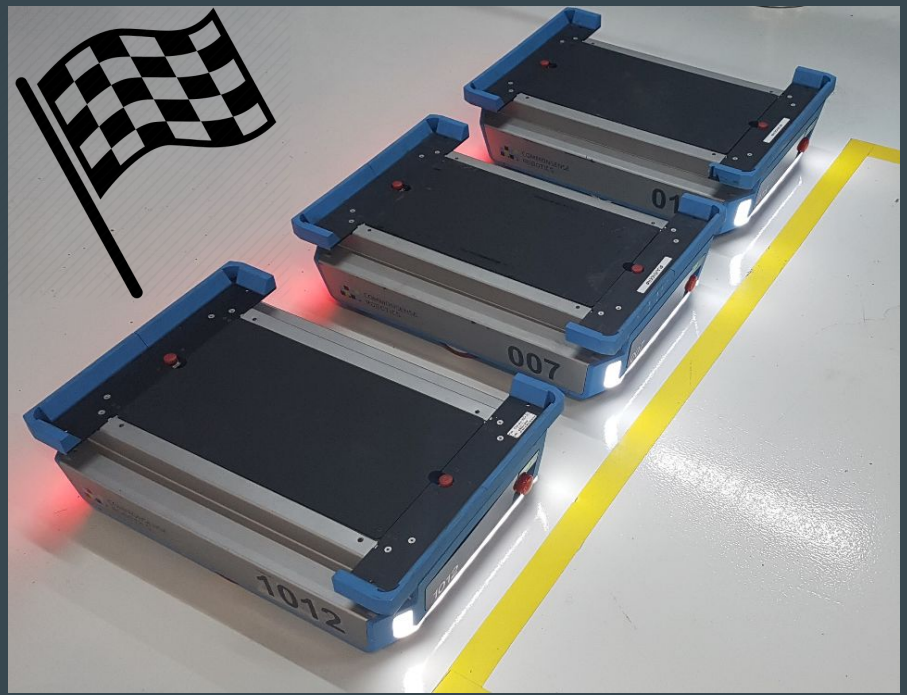


SimPy Overview

- Processes
  - Modeled by Python generators
  - All processes run in a single thread
- Environment
  - Can run in ‘real-time’ mode
  - Receives *initial_time* as parameter

SimPy Example - Robot Race

- A robot's speed is about 2-4 meters/second



```python
from random import randint
import simpy

num_robots = 3
sim_time = 5  # seconds
time_tick = 0.5

class Robot:
    def move(self, env, id):
        pos = 0
        while True:
            pos += randint(1,2)
            print(f"{env.now} r_{id} moved to {pos}")
            yield env.timeout(time_tick)

env = simpy.Environment()

for i in range(num_robots):
    r = Robot()
    env.process(r.move(env, id=i))

env.run(until=sim_time)
```

SimPy Example - Robot Race

- SimPy code is simulative only
- Parameters that affect performance:
  - Number of simulated components
  - Time tick granularity

Implementation

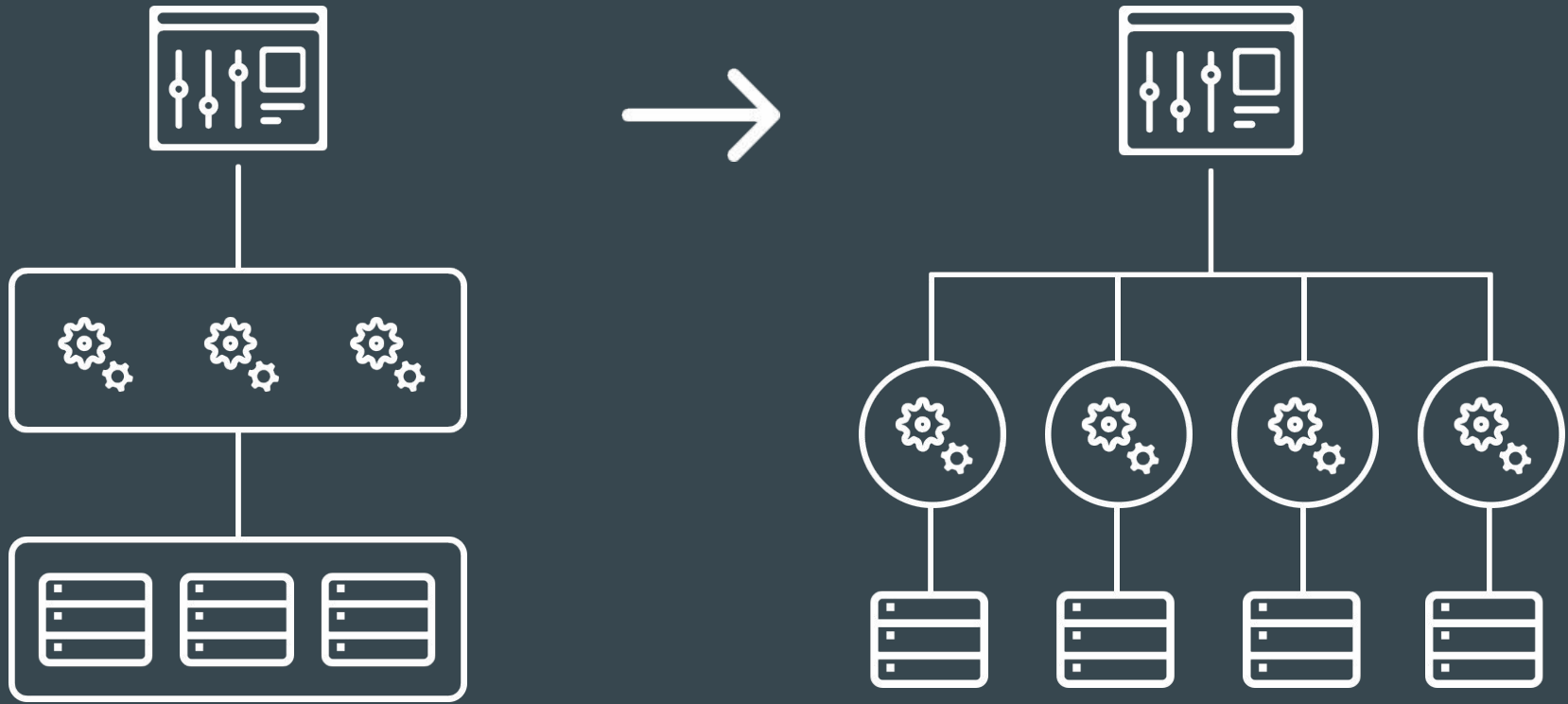
- Wrapping time-related functionality in our own module
  - *datetime.now()*
  - *time.time()*
  - *time.sleep()*
  - …
- Debugging - simulation timestamp in log

Time Leak - Event-Driven Component

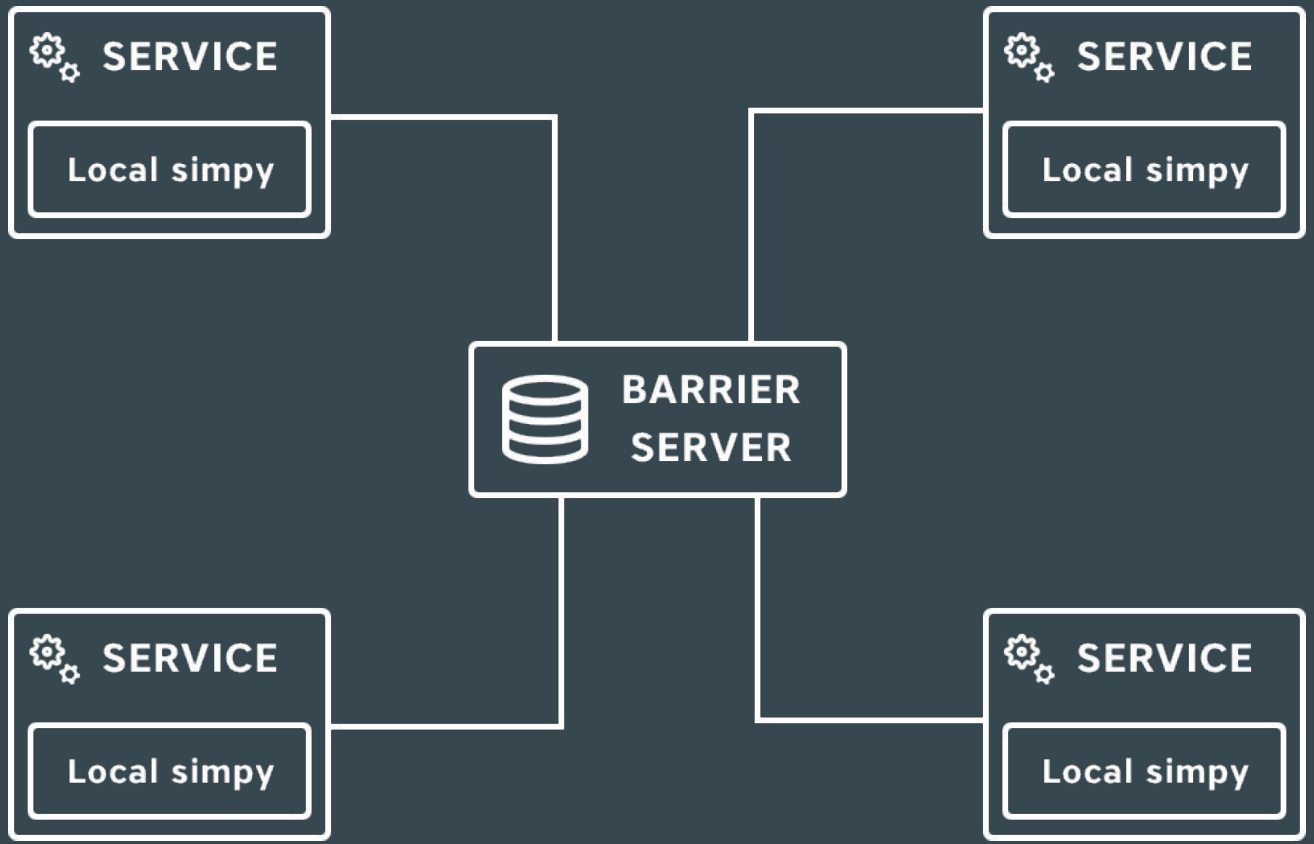
- Event-driven components are not naturally tied to time
- SimPy supports event-driven processes
- Not suitable for multi-threaded systems
- Solution: inherit from *Queue* and create a SimPy process that *joins* on itself in each time tick

```python
from threading import Thread
from queue import Queue
import simpy

time_tick = 1
sim = True

class EventDrivenQueue(Queue):
    def __init__(self, env, *args, **kwargs):
        super().__init__(*args, **kwargs)
        if sim:
            env.process(self._sim_join(env))

    def _sim_join(self, env):
        while True:
            self.join()
            yield env.timeout(time_tick)

class EventDrivenComponent:
    def run(self):
        while True:
            msg = q.get()
            print(f"Got {msg}")
            q.task_done()

class SimRobot:
    def work(self, env):
        i = 1
        while True:
            q.put(f"msg {i}")
            i += 1
            yield env.timeout(time_tick)

env = simpy.Environment()
# q = EventDrivenQueue(env)
q = Queue()
Thread(target=EventDrivenComponent().run, daemon=True).start()
env.process(SimRobot().work(env))
env.run(until=50)
```

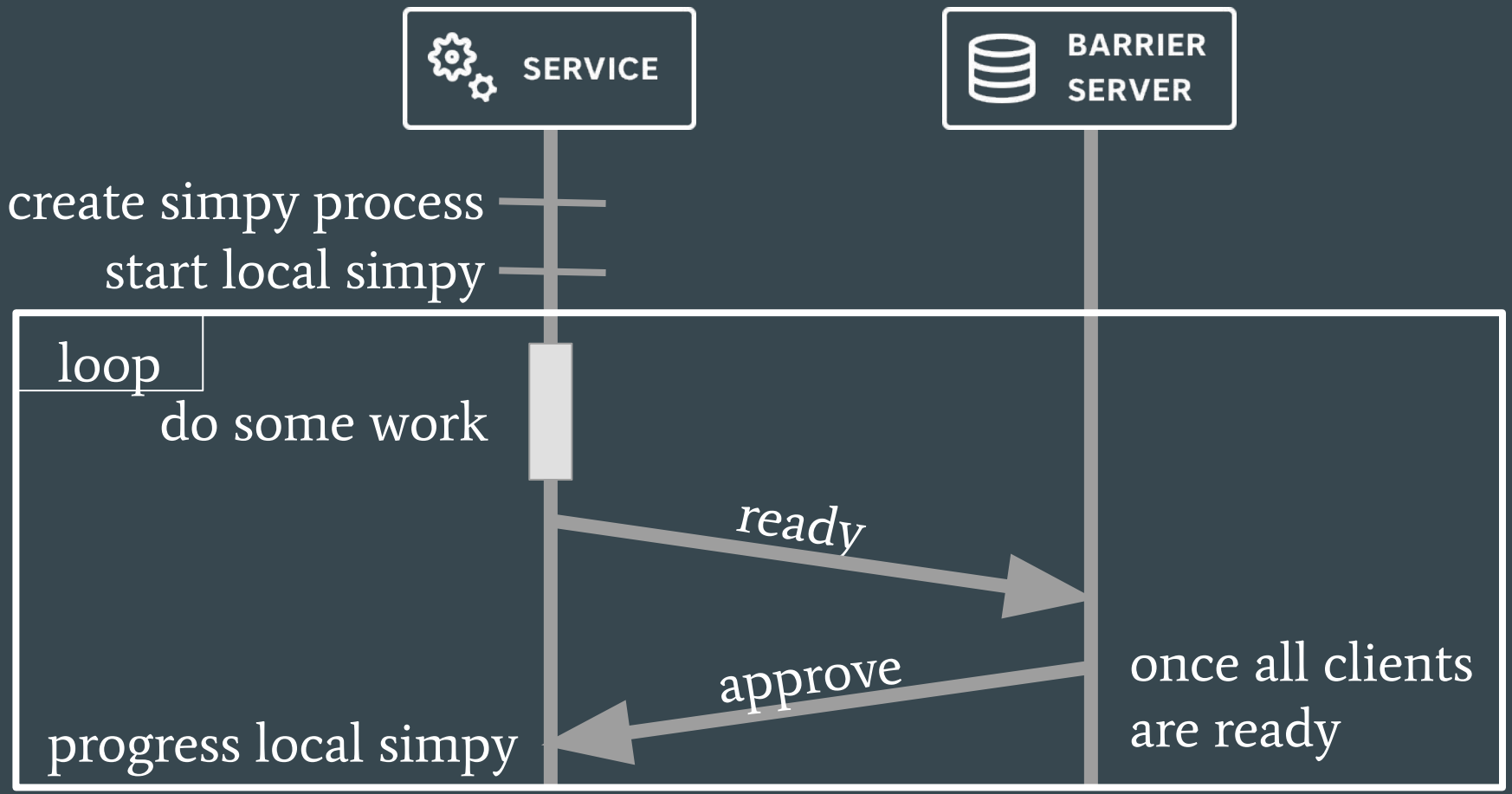
Distributed Simulation



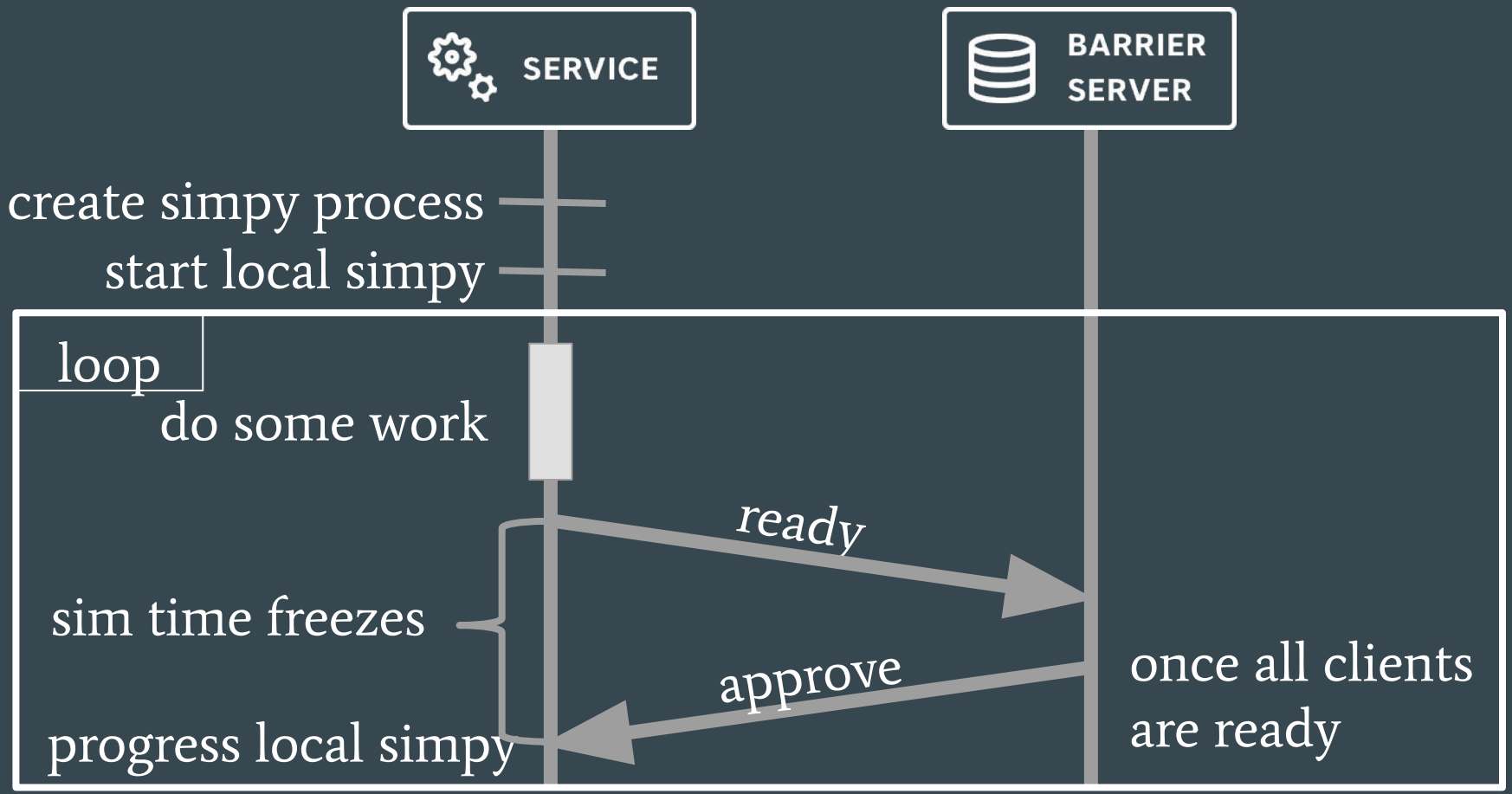
Distributed Simulation



Distributed Simulation



Distributed Simulation



Summary

- Simulation is a powerful tool
- DES makes it more powerful
- SimPy is SimPle
- Time leak - synchronize all components' time
- Easy to extend to a distributed simulation