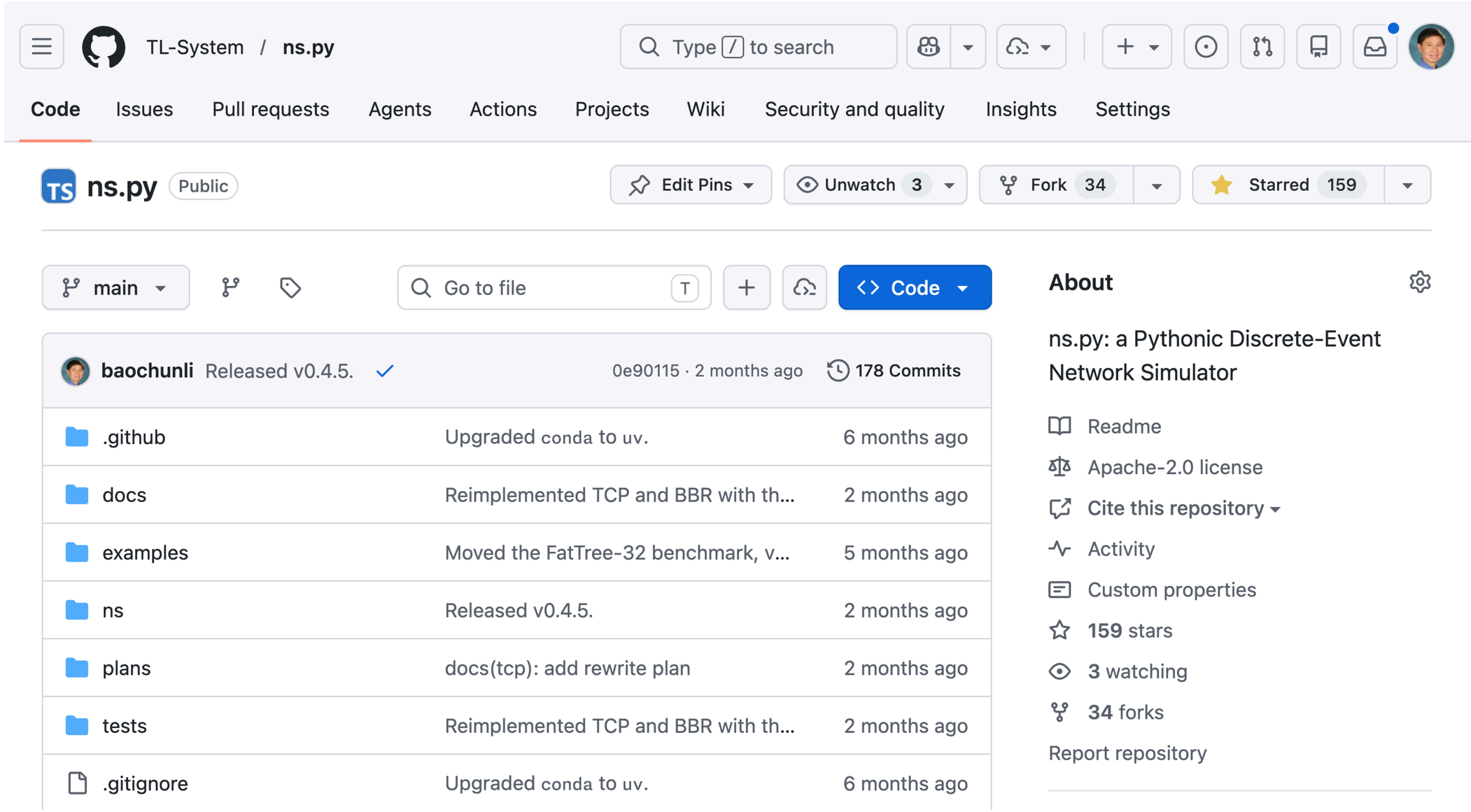
Days: Discrete-Event Network Simulation on Steroids

Baochun Li
Department of Electrical and Computer Engineering
University of Toronto

III. A Tale of Two Simulators

A. *Openings*

Complexity. In his article [28] titled "A Plea for Lean Software," Niklaus Wirth made the claim that "software's girth has surpassed its functionality," and that such complexity was due to software's monolithic design, in that all features are available all the time. We are of the opinion that current DES frameworks, such as ns-3 [9] and OmNeT++ [11], are too complex to use, to extend, and more importantly, to become performant at scale. If we use lines of code (LOC)



https://github.com/TL-System/ns.py

Commits on Nov 19, 2020

Updated .gitignore to exclude .vscode/.

 baochunli committed on Nov 19, 2020

53f4781

Added a basic design with examples.

 baochunli committed on Nov 19, 2020

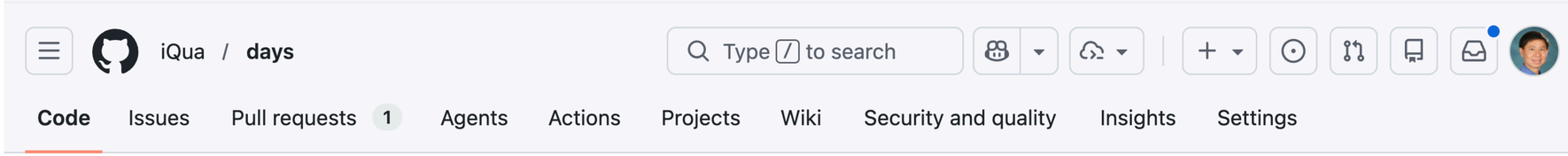
ce04a6f

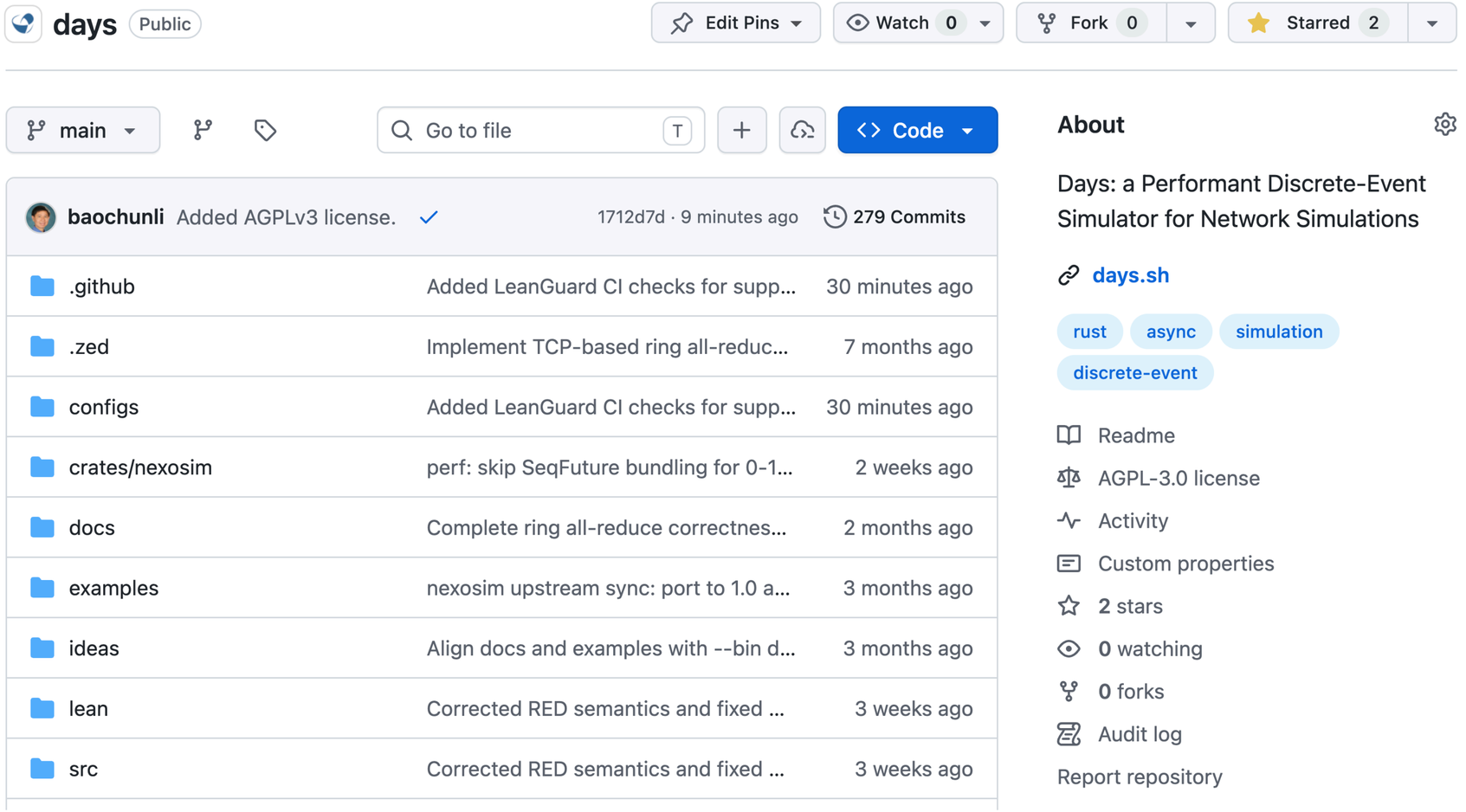
Initial commit

 baochunli authored on Nov 19, 2020

Verified

89012d0





https://github.com/iqua/days

Commits on Nov 14, 2023

Added the GitHub templates for issues and PRs.

 baochunli committed on Nov 14, 2023

6007092

A basic example on multiple packet generators sending packets to one packet sink.

 baochunli committed on Nov 14, 2023

c74f8e0

ns.py and Days

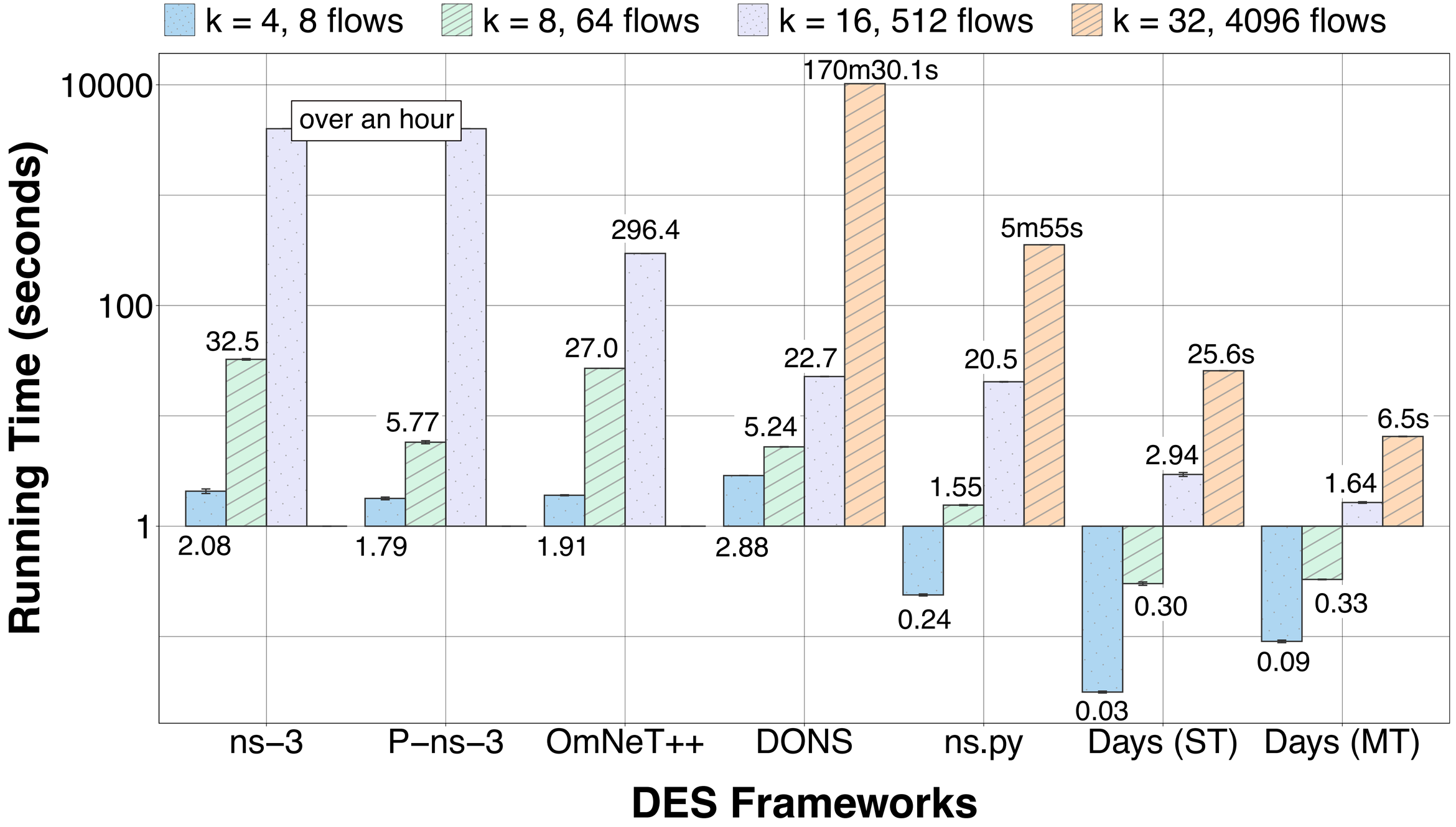
ns.py is written in Python, single-threaded, and uses generators — which uses **coroutines**

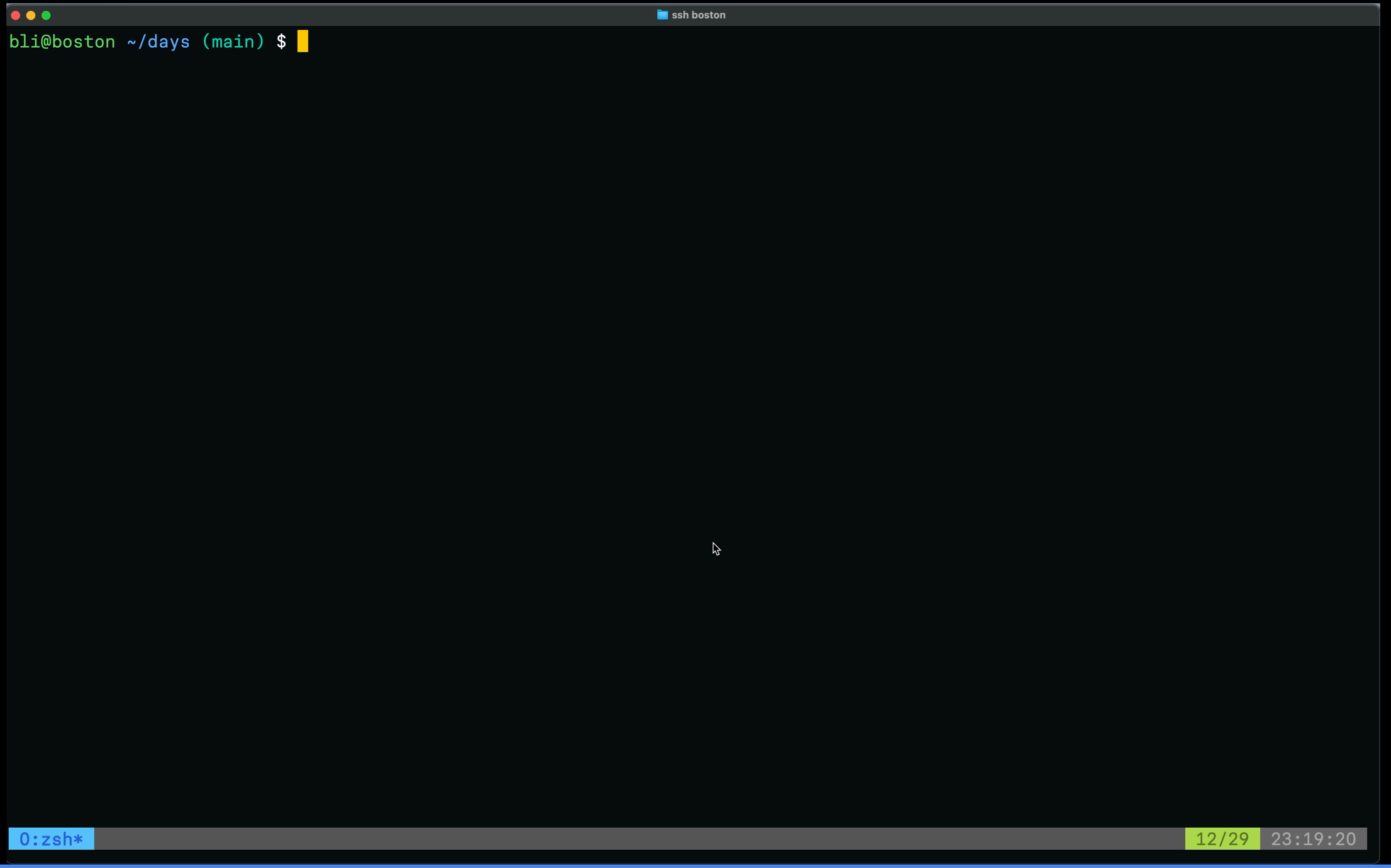
Days uses async Rust and **stackless coroutines** — for the **best possible** performance

1

Let’s look at some results first

Comparing ns.py and Days with ns-3, OMNET++, and DONS (SIGCOMM 2023, co-authors from Tsinghua University)


k = 4, 8 flows
k = 8, 64 flows
k = 16, 512 flows
k = 32, 4096 flows
Running Time (seconds)
10000
100
1
over an hour
170m30.1s
296.4
5m55s
32.5
27.0
22.7
20.5
25.6s
5.77
5.24
6.5s
2.94
1.55
1.64
2.08
1.79
1.91
2.88
0.24
0.30
0.33
0.03
0.09
ns–3
P–ns–3
OmNeT++
DONS
ns.py
Days (ST)
Days (MT)
DES Frameworks


ssh boston
bli@boston ~/days (main) $
0:zsh*
12/29
23:19:20

Performance figures are questionable. Network simulator is a widely discussed field with a long history. Even though careful design and substantial work were conducted by authors, it's hard to believe that their design deliver a speedup of more than 1000 times. Further evaluation with open-source code in the future is needed to back their claims on system performance.

Performance figures are questionable. Network simulator is a widely discussed field with a long history. Even though careful design and substantial work were conducted by authors, it's hard to believe that their design deliver a speedup of more than 1000 times. Further evaluation with open-source code in the future is needed to back their claims on system performance.

Performance figures are questionable. Network simulator is a widely discussed field with a long history. Even though careful design and substantial work were conducted by authors, it's hard to believe that their design deliver a speedup of more than 1000 times. Further evaluation with open-source code in the future is needed to back their claims on system performance.

Performance figures are questionable. Network simulator is a widely discussed field with a long history. Even though careful design and substantial work were conducted by authors, it's hard to believe that their design deliver a speedup of more than 1000 times. Further evaluation with open-source code in the future is needed to back their claims on system performance.

Overall Rating: Your overall rating (based on strengths, weaknesses, and quality of writing).

reject - I am strongly in favor of rejecting this paper and will try to convince others. (1)

Commits

main ▾ | All users ▾ | All time ▾

Commits on May 18, 2026

Added AGPLv3 license.
baochunli committed 52 minutes ago · ✓ 8 / 8 — 1712d7d

Added LeanGuard CI checks for supported protocols. (#96)
baochunli authored 1 hour ago · ✓ 8 / 8 — Verified — e003249

https://github.com/iqua/days

2

Why are ns.py and Days so much faster than, say, ns-3?

days

```
>_ OpenAI Codex (v0.131.0)

model:       gpt-5.5 xhigh   /model to change
directory:   ~/Playground/days
permissions: YOLO mode
```

Tip: *New* Use **/fast** to enable our fastest inference with increased plan usage.

› Compared to ~/Playground/ns-3.47, why is Days so much faster?

• I'll compare the simulator architectures from the local sources first, then sanity-check any benchmark/config assumptions before answering. Dependency graph: T1 inspect Days runtime and event loop depends_on: []; T2 inspect ns-3 core/event

2a

Simplicity

Popular discrete-event simulation frameworks

OMNET++ — 257K LOC

ns-3 — 551K LOC

ns.py has around 3K LOC, and is designed to be **easy to use**

Days has around 8K LOC (excluding the custom executor), and is trying to be as **fast** as possible

“Software’s girth has surpassed its functionality.”

— Niklaus Wirth, “**A Plea for Lean Software**,” 1995

“Ideally, only a basic system with essential facilities would be offered, a system that would lend itself to various extensions.”

— Niklaus Wirth, “**A Plea for Lean Software**,” 1995

Similar to recent simulators such as DONS, both ns-3 and Days are purpose-built simulators for datacenter networking

And optional features are feature-gated

```
#[cfg(feature = "l2_pfc")]
pub mod pfc;
```

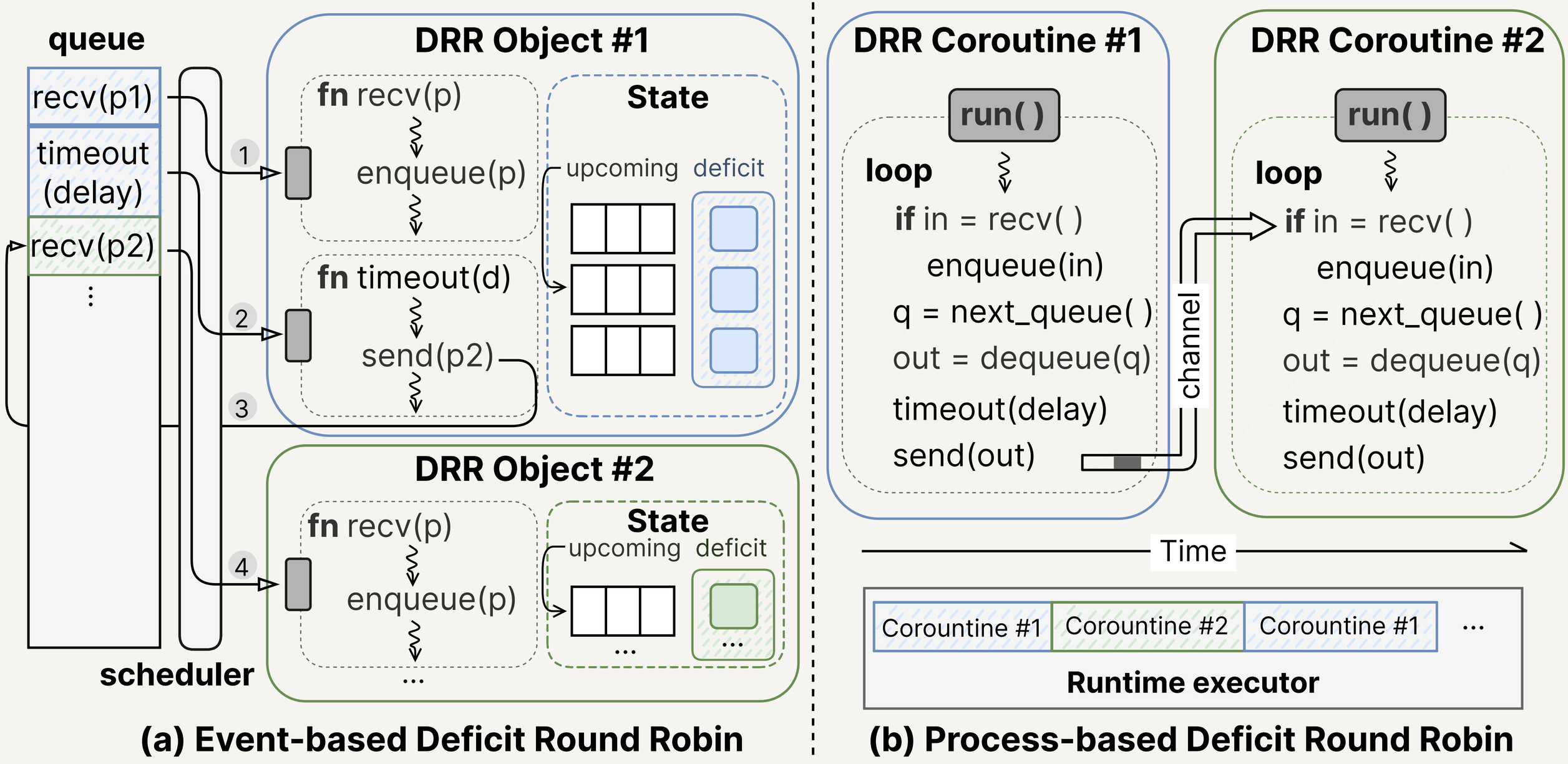
2b

Process-Based Design

Process-based design was proposed over six decades ago in 1963

But since ns-1 (1994), network simulators used event-based design

So what's the difference between the two designs?



(a) Event-based Deficit Round Robin

(b) Process-based Deficit Round Robin

Threads → Stackless Coroutines

Concurrency happens **entirely** within your code

An **executor** manages async tasks

Your code yields control from time to time, using the **`await`** keyword

Rust is one of **only** two languages that support **stackless** coroutines

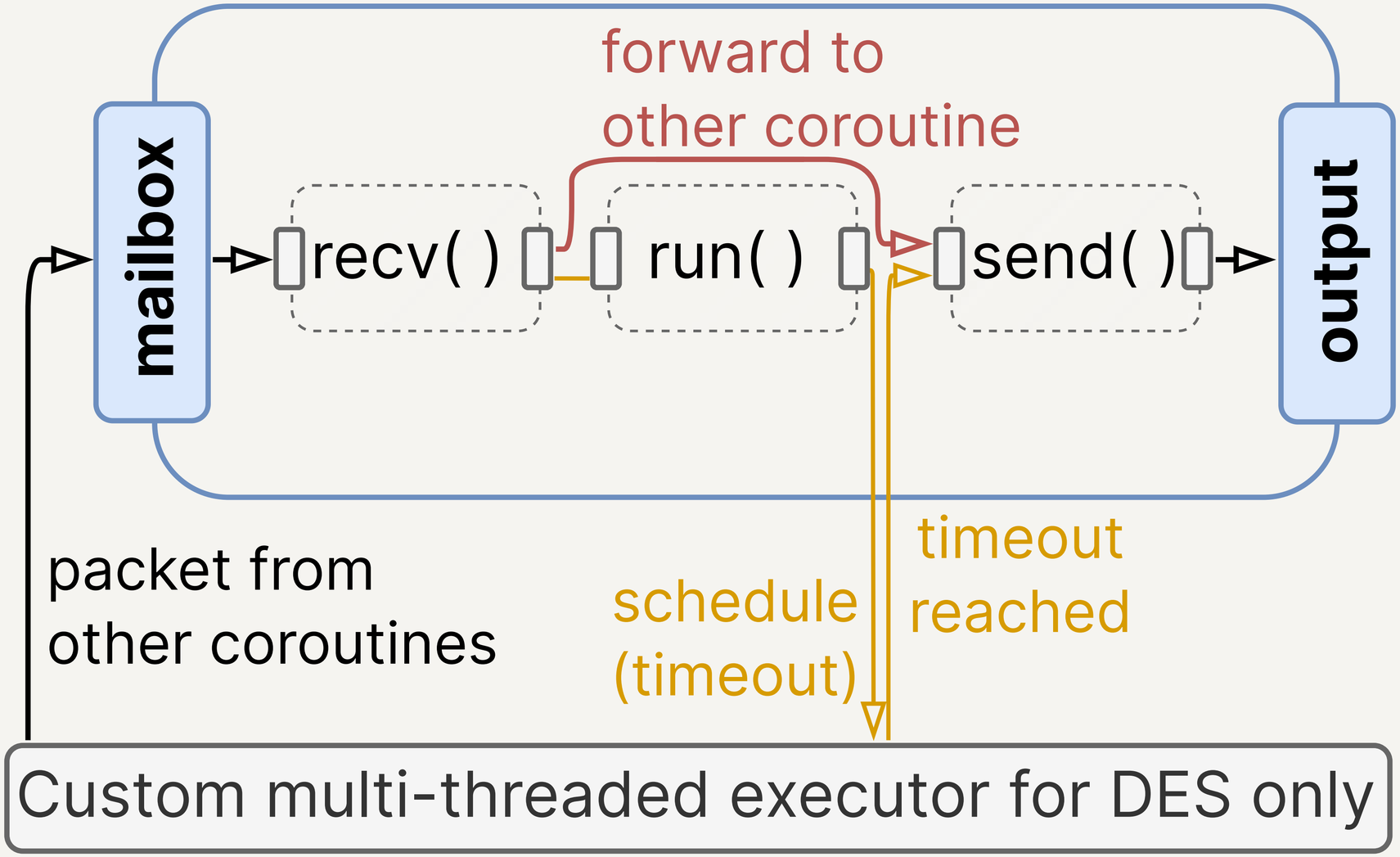
with **`Future`** and the **`async`** / **`await`** syntax

Go also supports coroutines, but the slower **stack-based** variant

2c

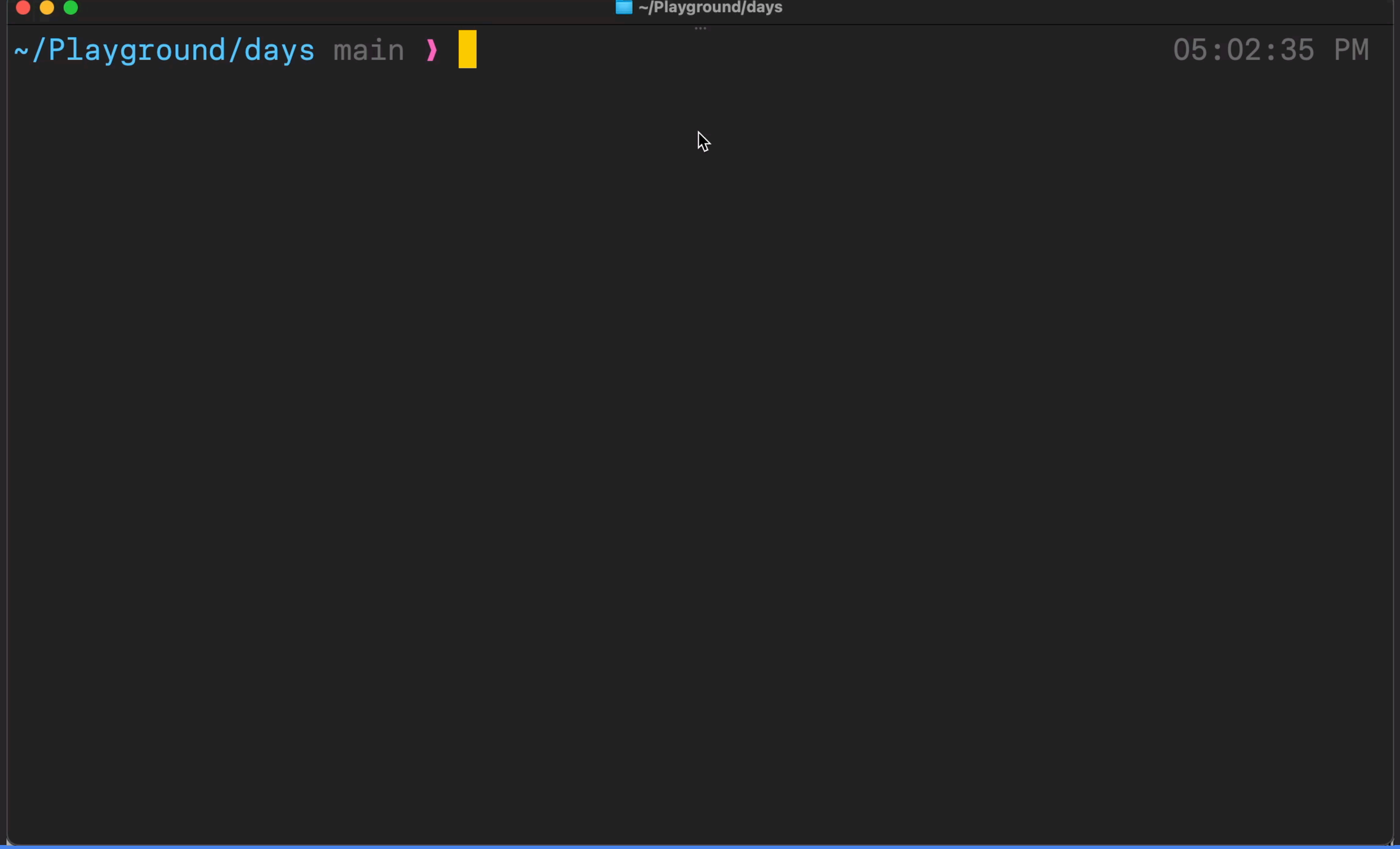
The Actor Model
with Mailboxes and Outputs

Coroutine with the Mailbox Model



2d

A custom, heavily optimized multi-threaded executor for discrete-event simulations only


~/Playground/days
~/Playground/days main ❯
05:02:35 PM

A heavily customized executor

Contention for globally shared states is minimized

Per-worker local event buffers with deterministic flushing

Time quantization

Keeping threads warm across barriers (step boundaries)

Let’s look at what the agents have to say

But wait! How do you verify that your results are faithful to protocol specs?

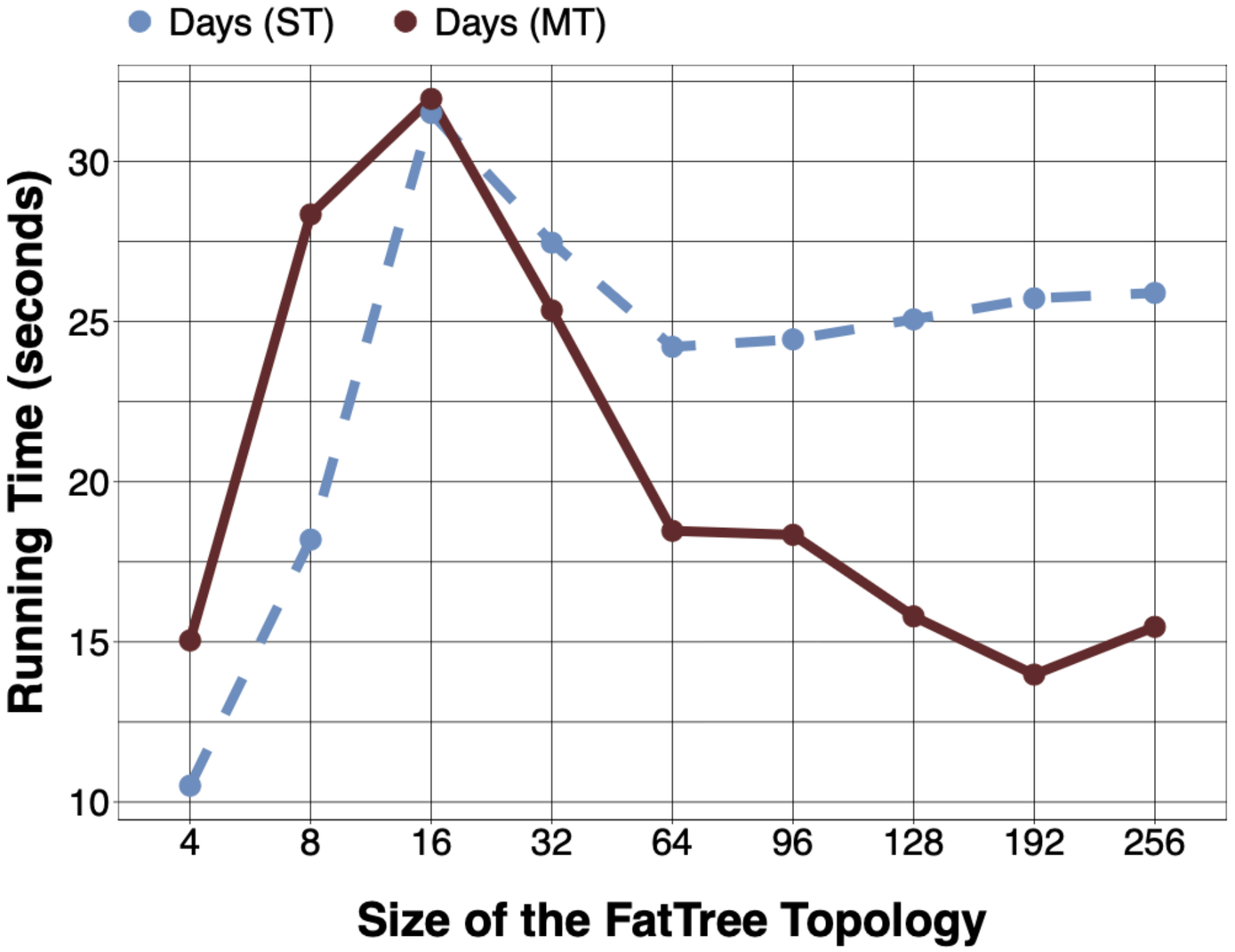
We have solved this problem in a rather unique way

And will submit a new paper about this approach

As a widely discussed field with a long history, we need a new, modern redesign for network simulation

And I hope ns.py and Days are the first, agent-friendly steps towards the right direction

https://days.sh



(b) Scaling up the topologies with 128 flows.