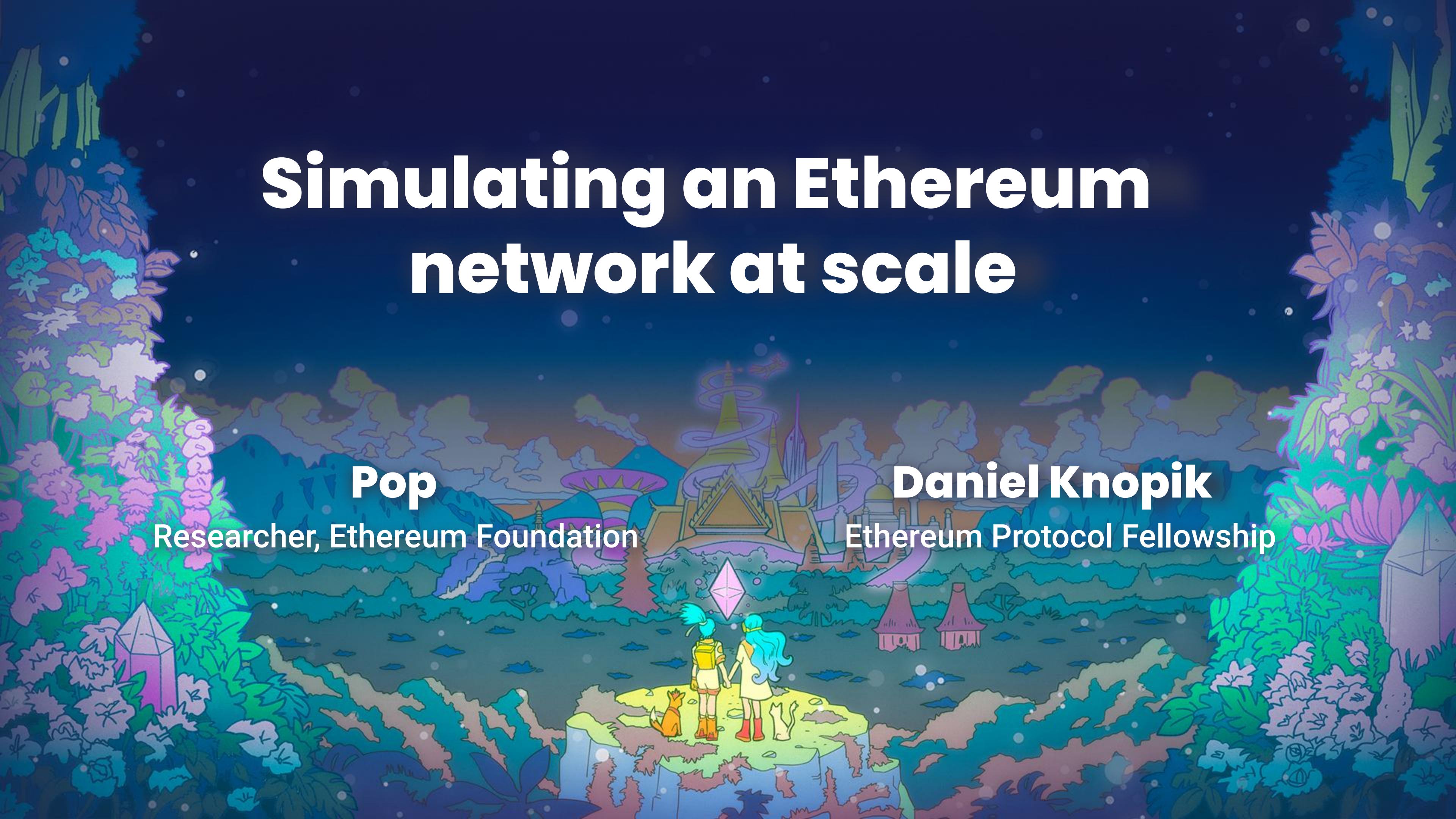
Simulating an Ethereum network at scale



Pop
Researcher, Ethereum Foundation

Daniel Knopik
Ethereum Protocol Fellowship

Why do network simulations?

Why? From our experience, there are at least two reasons.

- Get insights of new protocol ideas that cannot be easily derived from pure reasoning.
- Confirm insights/hypotheses that you have.

Inputs:

1. Protocol implementation.
2. Network topology (bandwidths, latencies, loss rates of links between nodes).

Outputs: The metrics you are interested in, such as, number of reorgs, bandwidth usage, message propagation time, etc. Then analyse the metrics to get or confirm the insights.

How we do simulations previously

Do network emulations instead of simulations

- Run clients natively on machines or containers and use “netem” to emulate link characteristics.

Drawbacks: Doesn’t scale well. To scale, you need a cluster of computers.

Moving to discrete-event simulators

Discrete-event simulators are simulators that create and process events inside simulated nodes and can freely control and jump to the time of the next event.

The clocks in simulated nodes are independent and different from the real clock.

Good side: the simulated clocks are not tied to the real clock, so we can scale a lot more.

Moving to discrete-event simulators

Issues with existing discrete-event simulators (like ns-3, OMNeT++, etc):

- You need to implement the protocol in a supported language and use their libraries to signal the events.

But it's not an issue with **Shadow**.

Shadow

- Instead of forcing you to use the simulator's libraries, you can implement the protocol in any language.
- Shadow will intercept the system calls made by your implementation and treat that as a signal of an event. For example, `gettimeofday`, `socket`, `send`, `recv`

In short, Shadow can simulate a lot of nodes with your real client softwares written in any language.

We implemented Ethshadow to let you simulate an Ethereum network easily with Shadow

Introducing Ethshadow

github.com/ethereum/ethshadow

Ethshadow is a configuration generator that makes it easy to simulate Ethereum networks in Shadow

Introducing Ethshadow

- Simple YAML configuration format
- Simulate realistic geolocation and network reliability
- Supports Geth, Lighthouse, and Teku, more planned
- Capture client metrics
- Run your own programs (e.g. to simulate attackers)
- Simulate >100 nodes on your laptop!

Geolocation and reliability

- Ethereum is a globally distributed network of p2p nodes. Communication latency matters!
- Home Stakers are important to Ethereum. Easily simulate bandwidth restrictions, added latency and packet loss.
- Use predefined locations and reliabilities or define your own.

```yaml
ethereum:
  validators: 110
  nodes:
    # First node:
    - location: east_asia
      reliability: reliable
    # Ten more nodes:
    - locations:
        - europe
        - na_east
      reliability: home
      count:
        total: 10
```

Supported Clients

- Supported Clients: Geth and Lighthouse
- Support for Reth and Teku in testing

- Easily run mixed client stacks

- Other supported software:
    - Prometheus for automatically capturing metrics
    - A purpose-built blob spammer
    - Your program?

```
ethereum:
  nodes:
    - location: east_asia
      reliability: reliable
      clients:
        el: reth
        cl: teku
    - locations: na_east
      reliability: home
      clients:
        el: geth
        cl: lighthouse
        vc: lighthouse_vc
    - locations: europe
      reliability: reliable
      clients:
        metrics: prometheus
```

Supporting MORE Clients

- Unsupported clients either
  - use an unsupported Linux system call
  - get stuck in a spinlock
  - use a supported Linux system call, but Shadow behaves differently from Linux

- Shadow supports 186 of ~375 system calls and 4 of ~80 socket options

- We can modify clients to avoid unsupported calls and options - but we want to run unmodified clients!
- **Implement missing features in Shadow instead**

Simulation Size

- You can run simulations on your Linux laptop!

- Small simulations with few nodes (<10) run quickly - faster than real time!
- Large simulations may run slower, depending on your system

- Example on Daniel’s system: 38 nodes (Reth, Lighthouse + Validator Client), ~28GB memory, taking ~45 minutes for 30 minutes of simulated time
- Example on Pop’s system: 100 nodes (Geth, Lighthouse + Validator Client), ~16GB memory, ~32GB swap, taking ~11 hours for 10 minutes of simulated time

→ Using swap is slow, but enables large simulations!

Large simulations

- Simulations with **over 1000 nodes** have been tested!
- 1TB of memory, 45 minutes simulated time takes ~4,5h
- Memory speed becomes the bottleneck

Ethshadow in practice

- Experiments testing PeerDAS and IDONTWANT
- 10 minute presentation

Thank you!

EF Research is hiring a p2p network expert!
Go to https://jobs.lever.co/ethereumfoundation

Play with Ethshadow at
https://github.com/ethereum/ethshadow

Pop
Researcher, Ethereum Foundation
pop@ethereum.org

Daniel Knopik
Ethereum Protocol Fellowship
daniel@dknopik.de
@daniel_knopik (X, Telegram)