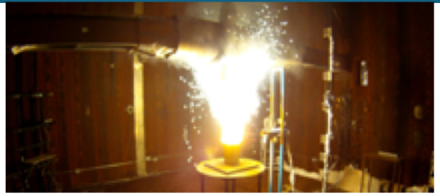
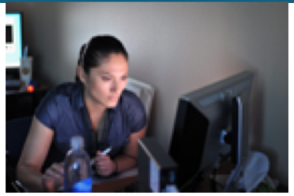
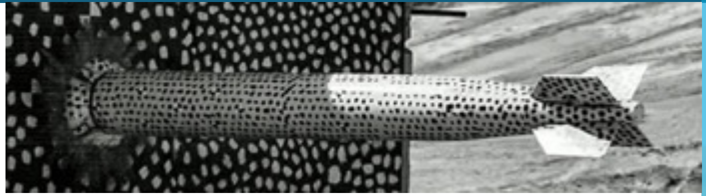
Sandia National Laboratories

Proteus: A DLT-Agnostic Emulation and Analysis Framework







Russell Van Dam
Thien-Nam Dinh
Christopher Cordi
Gregory Jacobus
Nicholas Pattengale
Steven Elliott

U.S. DEPARTMENT OF ENERGY NNSA

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

SAND2019-9324 C

Background

Distributed Ledger Technologies (DLT) have seen exponential growth

- 2,400 listed cryptocurrencies
- Varied trust models: permissionless, permissioned, federated
- Varied data structures: blockchain, DAG, HashGraph

Existing analysis tools

- Simulation – simbit, VIBES
- Emulation – SherlockFog
- Hybrid – Shadow Bitcoin plugin
- Physical Testbed – Bitcoin-NG, TrustChain, BLOCKBENCH, Grid’5000

Contributions

DLT-Agnostic emulation framework

- Abstract away infrastructure common to all DLTs
- Minimize user effort for implementing new DLTs
- Provide standardized tools for analysis

Extend capabilities of underlying FIREWHEEL orchestration platform

Demonstrated use case: Ethereum 51% attack

FIREWHEEL

Experiment orchestration tool for managing emulated networks

Focus on supporting well-structured experiments at a large scale

Specific capabilities:

- Programmatic definition of experiment topologies
- VM deployment across compute clusters
- Management and execution of in-experiment events
- Centralized collection, analysis, and display of experimental data
- Repeatability

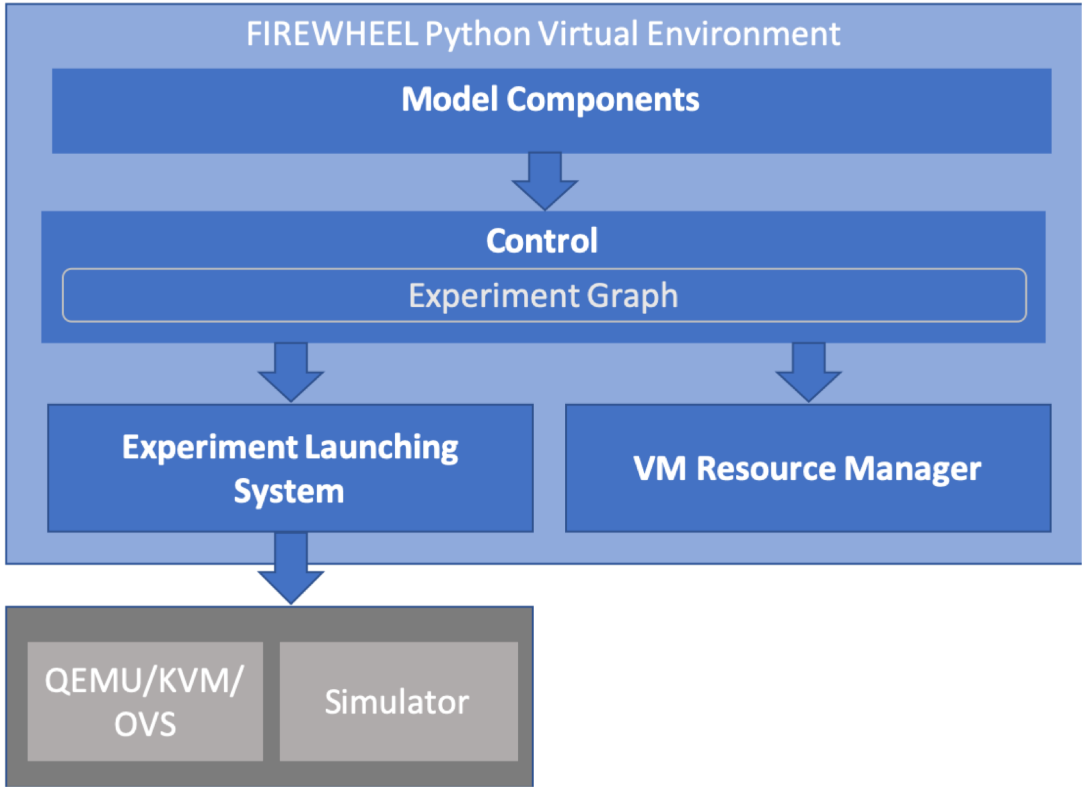
FIREWHEEL Architecture

Model components
- Collection of files for modeling experiment components
- Primary interface for building applications like Proteus

Control
- Translate model components into graph representation
- Launch experiment using configurable emulation tools

VM Resource Handler
- Manage scheduled experiment events
- Redirect output for logging and analysis

Networking
- Provide emulated switches and routers
- Configure parameters such as bandwidth and latency



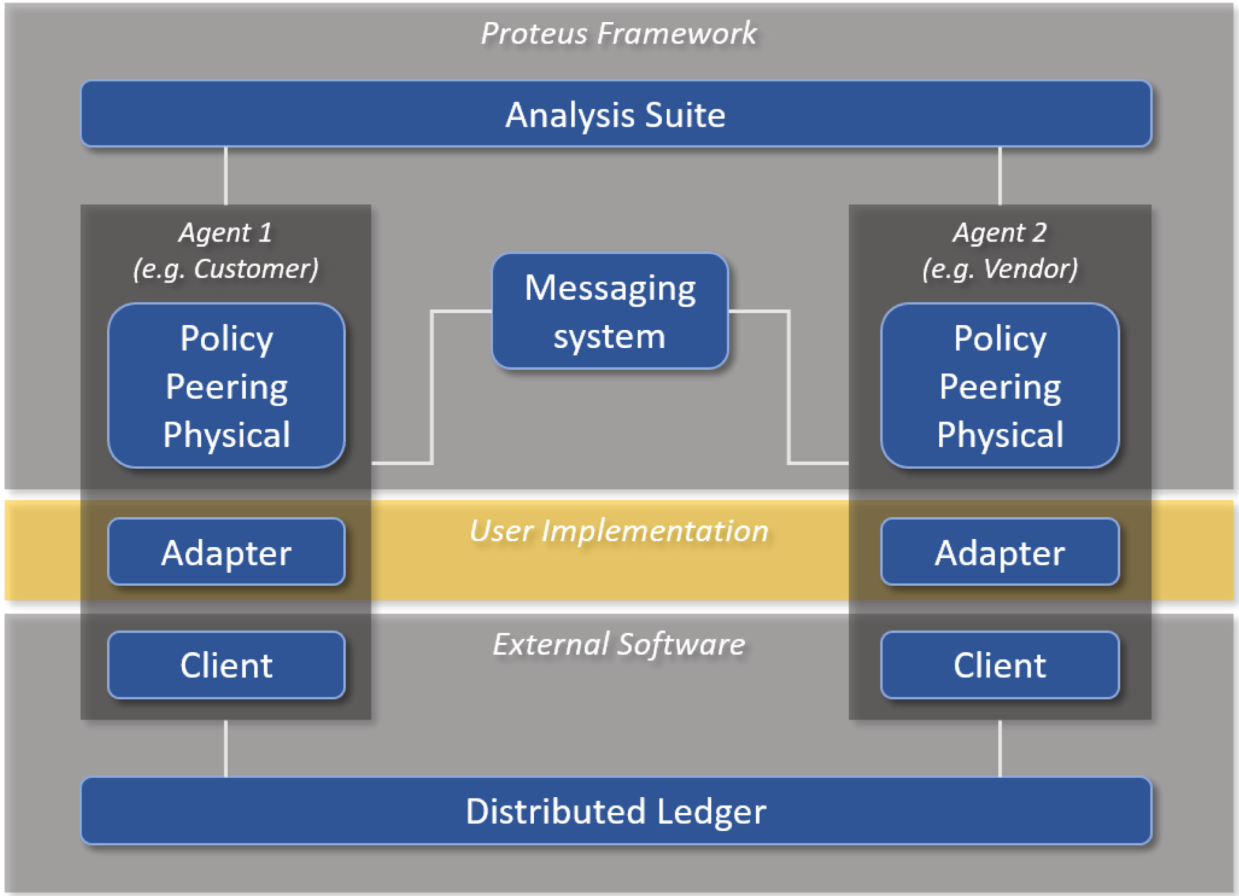
Proteus Architecture

Agent-based modeling paradigm

- Compose simple behaviors to create complex topologies
- Intuitive mapping to real-world agent types

Extensible by design

- Easy access to common model components
- Well-defined process to add new DLTs



What Your Emulation Can Do For You

Model components

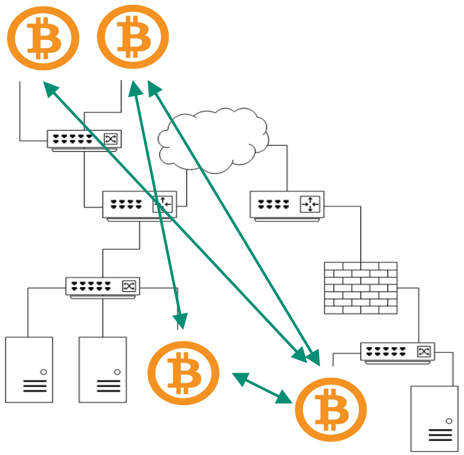
- Physical – physical, data-link, and networking layer topologies
- Peering – peer-to-peer overlay topology
- Policy – DLT-level actions (e.g. send tx or propose a new block)

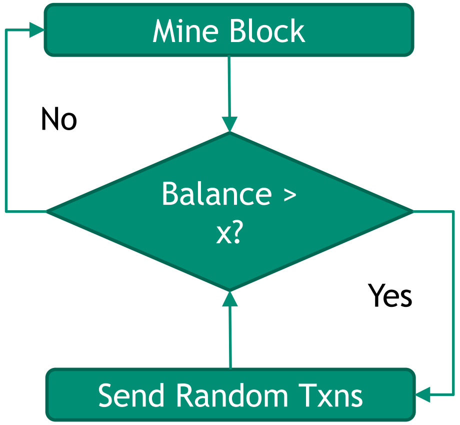
Messaging system

- Coordinated launch and teardown
- Information transfer between Agents
- Global synchronization of Agents

Analysis

- Generic system metrics
- DLT-level metrics





Mine Block

No

Balance > x?

Yes

Send Random Txns

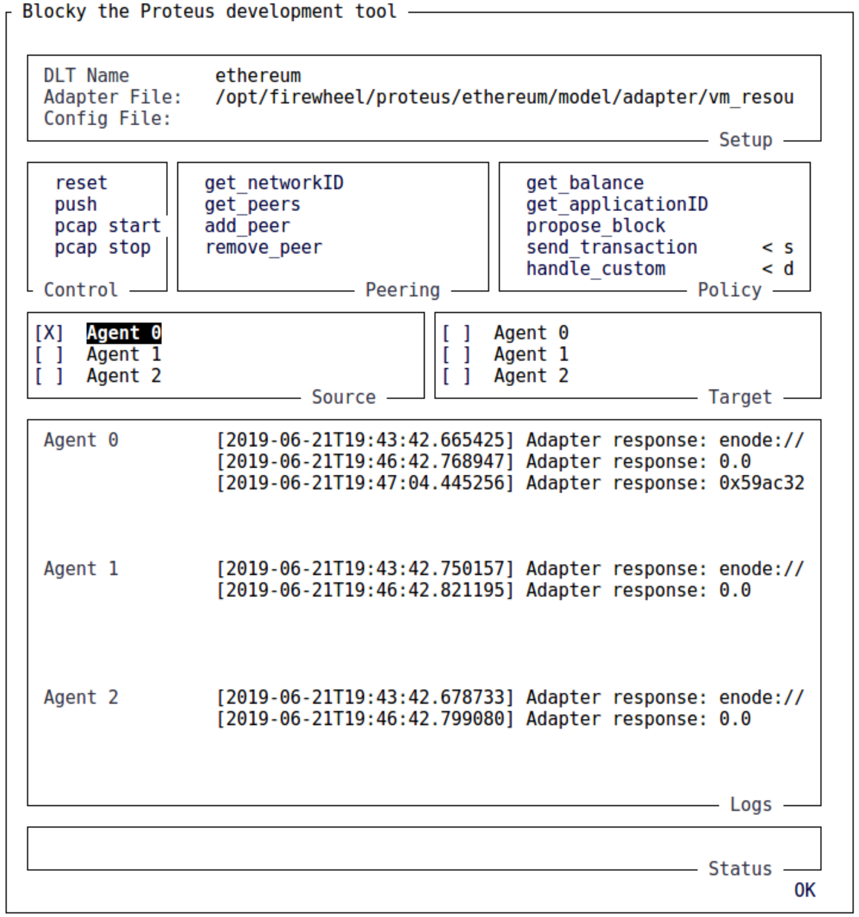
What You Can Do For Your Emulation

Model components

- Client – process for installing and configuring DLT software
- Adapter – hooks for translating abstract Proteus instructions into DLT actions

Blocky development tool

- Terminal application to aid adapter development
- Launch small-scale FIREWHEEL test environment
- Incrementally develop and test adapter implementation



```
Blocky the Proteus development tool

DLT Name         ethereum
Adapter File:    /opt/firewheel/proteus/ethereum/model/adapter/vm_resou
Config File:
                                                                 Setup

 reset          get_networkID            get_balance
 push           get_peers                get_applicationID
 pcap start     add_peer                 propose_block
 pcap stop      remove_peer              send_transaction      < s
                                         handle_custom         < d
Control                         Peering                     Policy

[X]  Agent 0                           [ ]  Agent 0
[ ]  Agent 1                           [ ]  Agent 1
[ ]  Agent 2                           [ ]  Agent 2
                              Source                           Target

Agent 0          [2019-06-21T19:43:42.665425] Adapter response: enode://
                 [2019-06-21T19:46:42.768947] Adapter response: 0.0
                 [2019-06-21T19:47:04.445256] Adapter response: 0x59ac32


Agent 1          [2019-06-21T19:43:42.750157] Adapter response: enode://
                 [2019-06-21T19:46:42.821195] Adapter response: 0.0


Agent 2          [2019-06-21T19:43:42.678733] Adapter response: enode://
                 [2019-06-21T19:46:42.799080] Adapter response: 0.0

                                                                  Logs

                                                                Status
                                                                    OK
```

Case Study: 51% Attack

We emulated a 51% attack on a private Ethereum network

We split the network into two partitions: a malicious partition with 60% of the network hash power, and an honest partition with the other 40%.

This network topology facilitates a double spend and allows the 51% attack to occur.

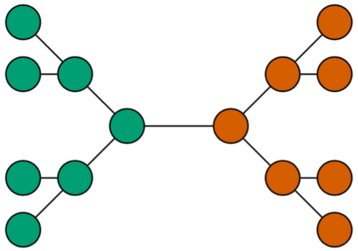


Figure 3: Example initial P2P topology of the 51% attack.

Experiment Setup

2,000 Ubuntu 16.04 Server VMs running go-etherum

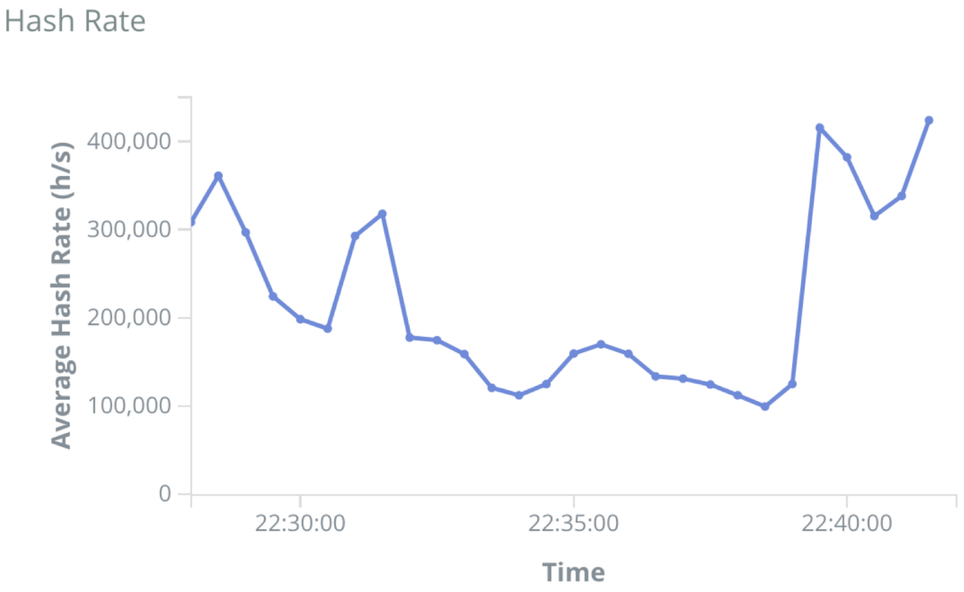
20 FIREWHEEL nodes

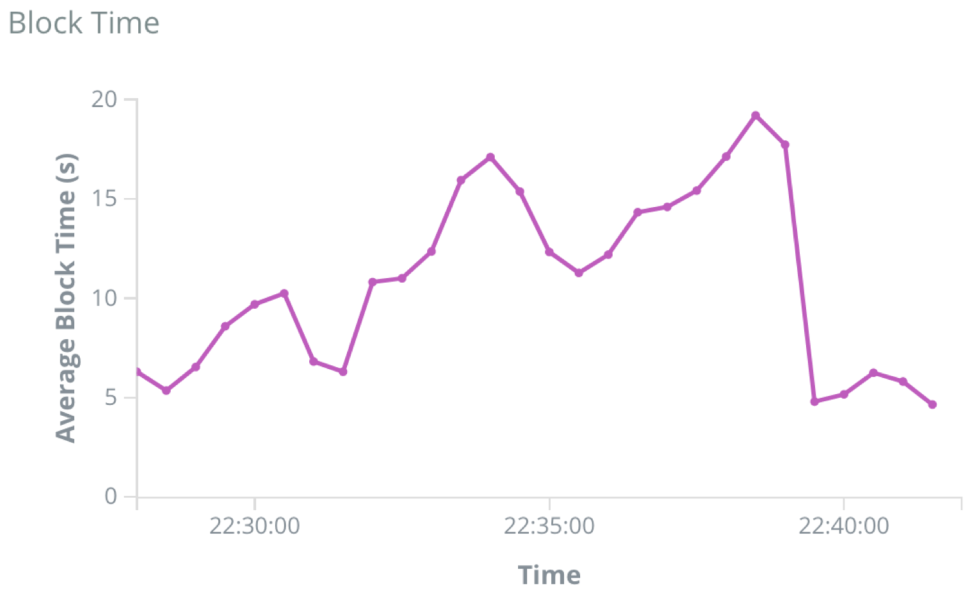
- Dual socket Intel® Xeon® E5-v4 2.10GHz CPUs
- 512GB Memory
- Local solid-state drives
- 100 Gigabit Ethernet

Results

During the 51% attack, we collected metrics about network hash power and block times as seen by the honest partition.

A large drop in hash power can be seen between the start of the attack (22:28) and the end (22:39). Block time is approximately the inverse.

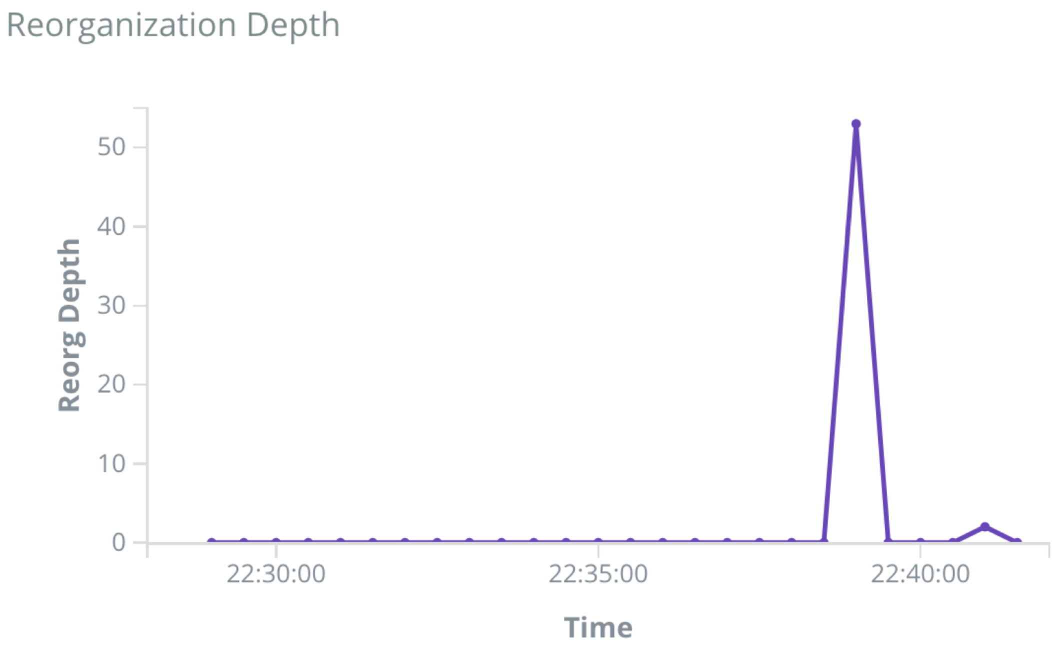




Results

We also collected metrics about reorganization depth, as seen by the honest partition.

There is a clear spike in reorganization depth when the 51% attack completes, indicating success.



Future Work

Cross-DLT analytics to detect common attacks

Developing adapters for non-cryptocurrency DLTs and assessing their compatibility.

Incorporating real-world P2P topologies in Proteus and evaluating how that impacts the indicators of 51% attacks.

Conclusion

Proteus is an agent-based framework for conducting rapid, emulated analysis of DLTs.

Proteus allows for quick development of DLT clients and adapters.

The results of the 51% attack case study validate Proteus' utility in assessing the security guarantees of DLTs.

Questions?