



ARISTOTLE UNIVERSITY OF THESSALONIKI

FACULTY OF SCIENCES
SCHOOL OF INFORMATICS
DATA & WEB SCIENCE M.SC. PROGRAMME

Automating massive deployment of bitcoin private network nodes for network simulation

Student:

Napoleon-Christos Oikonomou

Advisors:

Professor, Athena Vakali

PhD Candidate, Georgios Vlahavas

Thessaloniki, March 2020

Motive

- In recent years, Bitcoin has become one of the most quickly evolving technologies.
- Lots of research is being conducted not only by technology scientists, but also by researchers in different fields.
- The nature of the technology demands the continuous creation and deployment of networks in order to test various hypotheses.
- There is a need for tools that:
 - Automate the process of deploying multiple networks of multiple nodes.
 - Make the creation and editing of networks as easy as possible.

Purpose of Master's Thesis

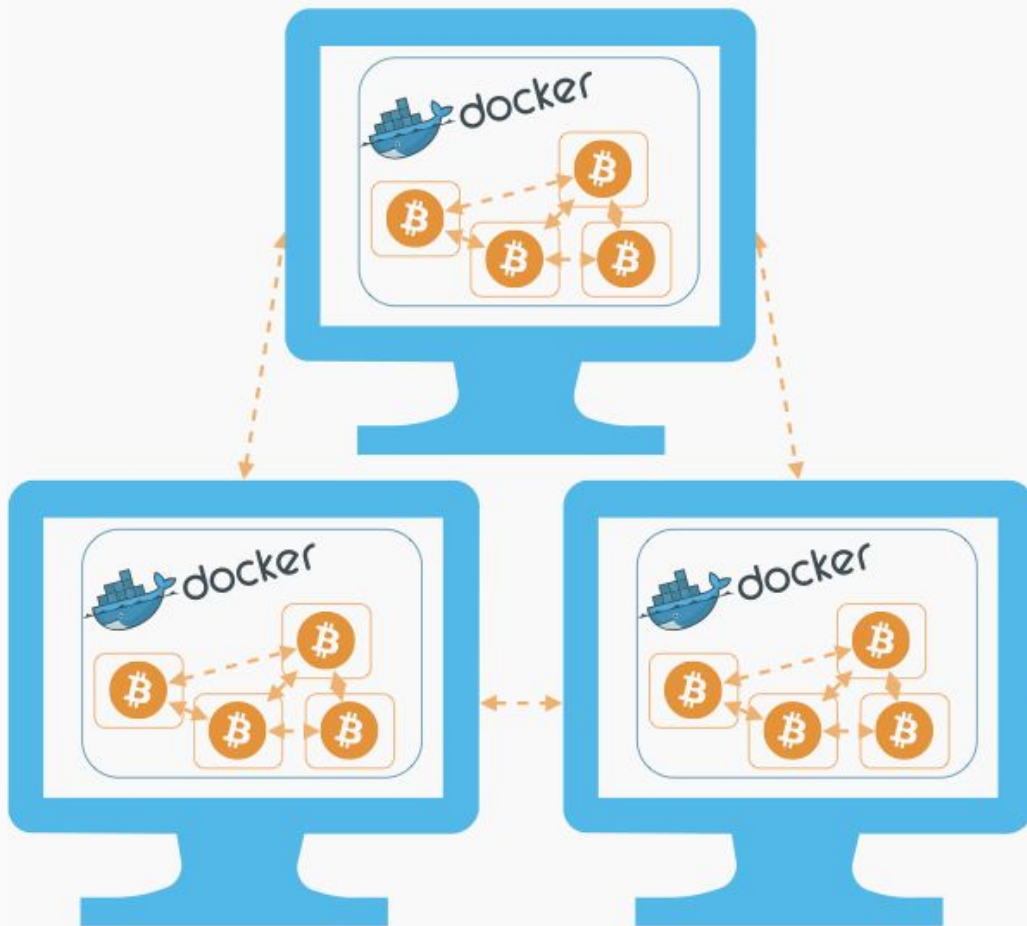
- Create such a tool that “hides” as much of the complexity of deploying a Bitcoin Network as possible, while making the process of creating and editing one easy to use, regardless of the background of the user.

Basic Idea :

- Use virtualization for deploying a node to get rid of hosting-system-specific architectures, dependencies and incompatibilities.
- Use smart mechanisms towards code reusability in order to lower compiling and execution times.
- Create a Command Line Interface where by answering simple questions and inputting configurations with a structured, predefined way, a user can easily deploy a custom network.

- *Virtualization* refers to the coexistence of multiple, isolated from each other userspace instances under the kernel of an operating system.
- Two mainstream ways to achieve this: Full-blown Virtual Machines & Containerization.
- In this work Docker Containers were used, because they require less resources, provide better results and are generally easier to use and build upon.¹
- Docker is a tool that allows bundling applications and their dependencies into containers, in order to run them in an isolated fashion.
- First, an image of a container with Bitcoin's Core Software precompiled was created.
- Using this image as base, alongside specialized compiling software, the system is able to create new, custom containers by editing only the necessary source files.

¹Chung et al., "Using Docker in High Performance Computing Applications" in IEEE Sixth International Conference on Communications and Electronics, 2016



- Multiple machines per network.
- Where each machine hosts multiple nodes.
- Each node runs in its isolated environment using specified resources.
- Node communication with peers both inside each host and in other hosts.

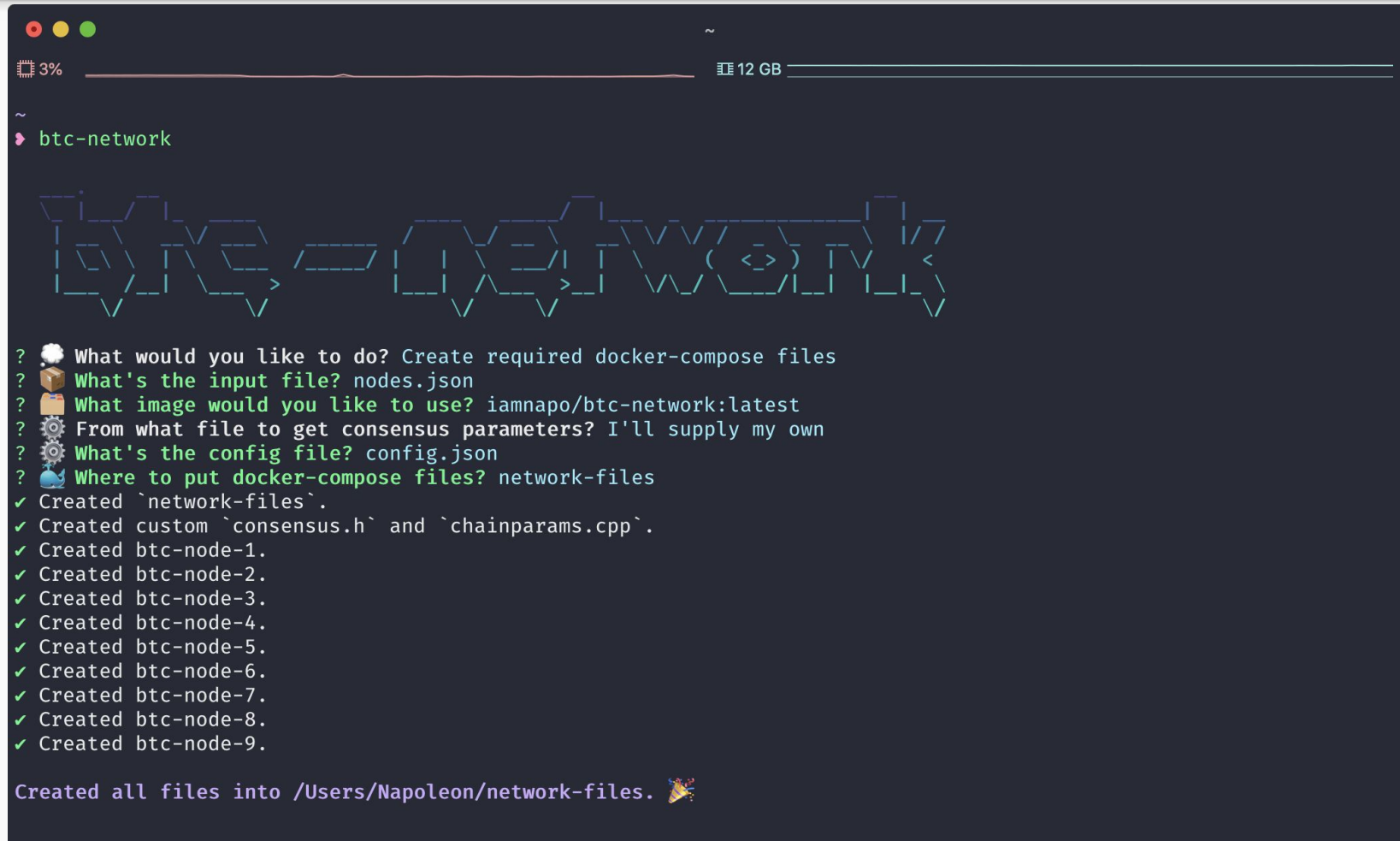
- Manually creating containers and deploying the network is still a complex and cumbersome process. As such, a Command Line Interface was created.
- By answering simple questions and inputting configurations with a structured, predefined way, a user can easily deploy a custom network.
- This CLI:
 - Gathers the required input.
 - Parses configuration files and edits Bitcoin's source code appropriately.
 - Recompiles modified files.
 - Creates Docker containers using this custom configuration.
 - Connects them as peers.
 - Deploys them to specified hosts.

```
[
  { "ip": "host.docker.internal", "p2p_port": 18501, "rpc_port": 18401 },
  { "ip": "host.docker.internal", "p2p_port": 18502, "rpc_port": 18402 },
  { "ip": "host.docker.internal", "p2p_port": 18503, "rpc_port": 18403 },
  { "ip": "host.docker.internal", "p2p_port": 18504, "rpc_port": 18404 },
  { "ip": "host.docker.internal", "p2p_port": 18505, "rpc_port": 18405 },
  { "ip": "host.docker.internal", "p2p_port": 18506, "rpc_port": 18406 },
  { "ip": "host.docker.internal", "p2p_port": 18507, "rpc_port": 18407 },
  { "ip": "host.docker.internal", "p2p_port": 18508, "rpc_port": 18408 },
  { "ip": "host.docker.internal", "p2p_port": 18509, "rpc_port": 18409 }
]
```

- Specifying IPs and ports of the host of each node.

```
{
  "chainparamsCPP": {
    "nSubsidyHalvingInterval": 1000
  },
  "consensusH": {
    "COINBASE_MATURITY": 50
  }
}
```

- Specifying custom network parameters.



```
~
3% 12 GB

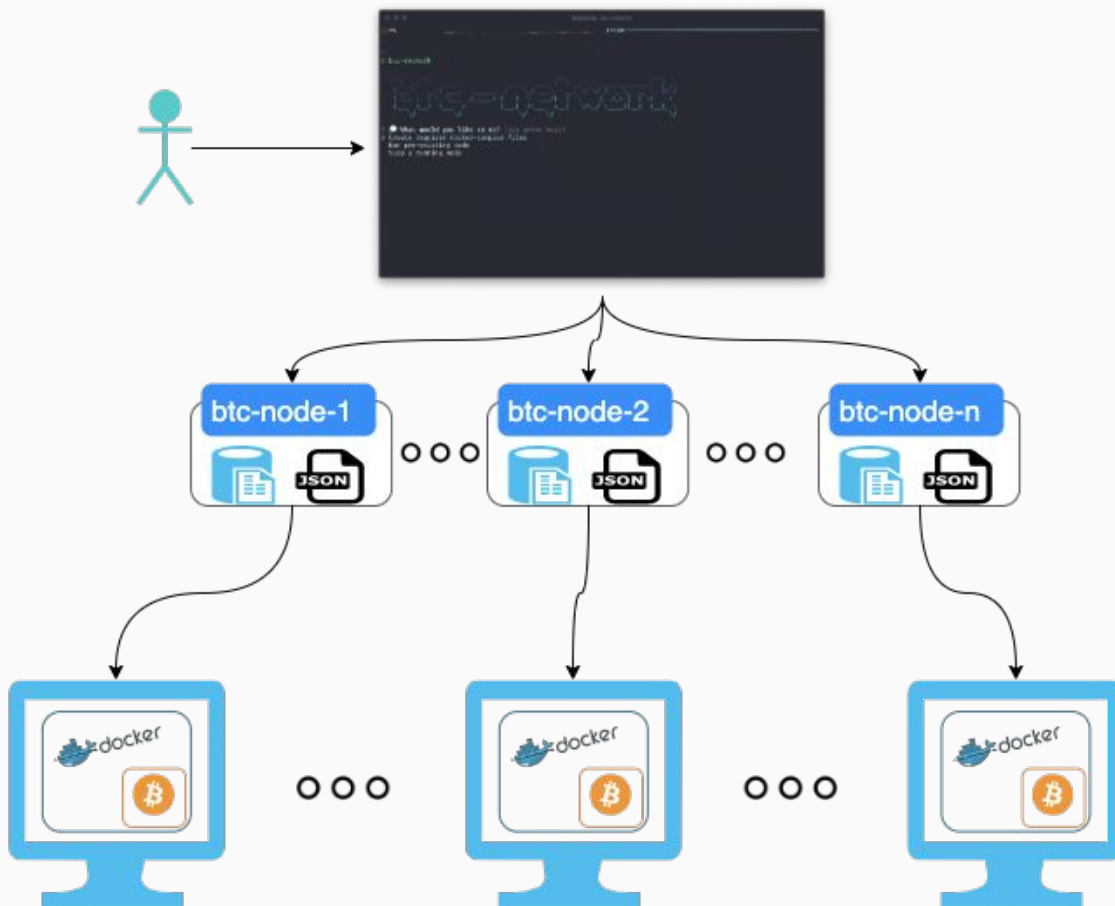
~
btc-network

btc-network

? 🧠 What would you like to do? Create required docker-compose files
? 📁 What's the input file? nodes.json
? 🖼️ What image would you like to use? iamnapo/btc-network:latest
? ⚙️ From what file to get consensus parameters? I'll supply my own
? ⚙️ What's the config file? config.json
? 📁 Where to put docker-compose files? network-files
✓ Created `network-files`.
✓ Created custom `consensus.h` and `chainparams.cpp`.
✓ Created btc-node-1.
✓ Created btc-node-2.
✓ Created btc-node-3.
✓ Created btc-node-4.
✓ Created btc-node-5.
✓ Created btc-node-6.
✓ Created btc-node-7.
✓ Created btc-node-8.
✓ Created btc-node-9.

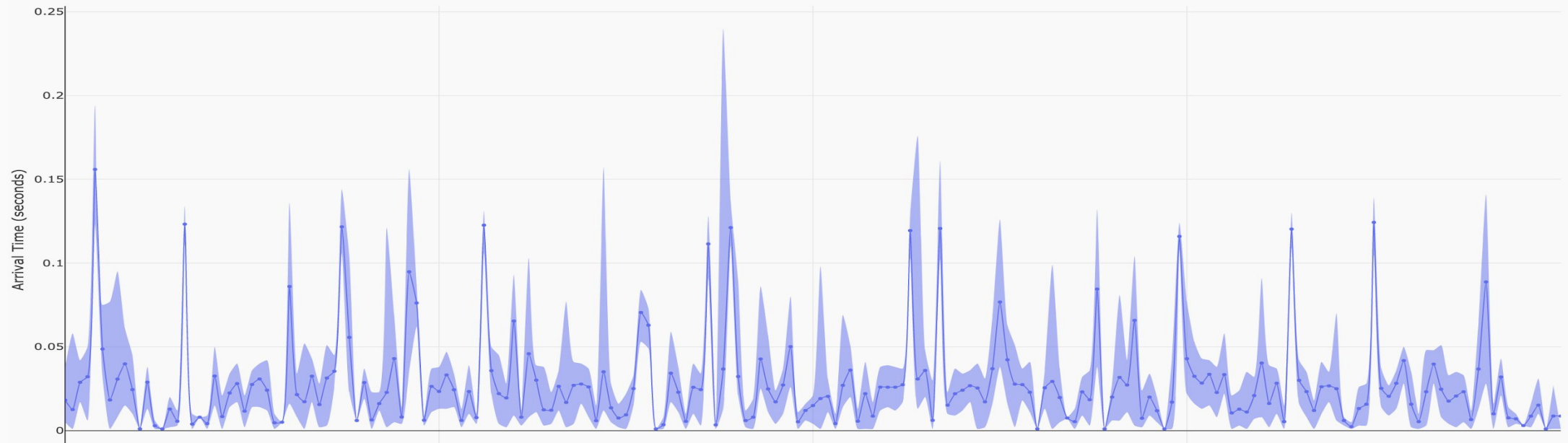
Created all files into /Users/Napoleon/network-files. 💣
```


btc-network - Final System Structure

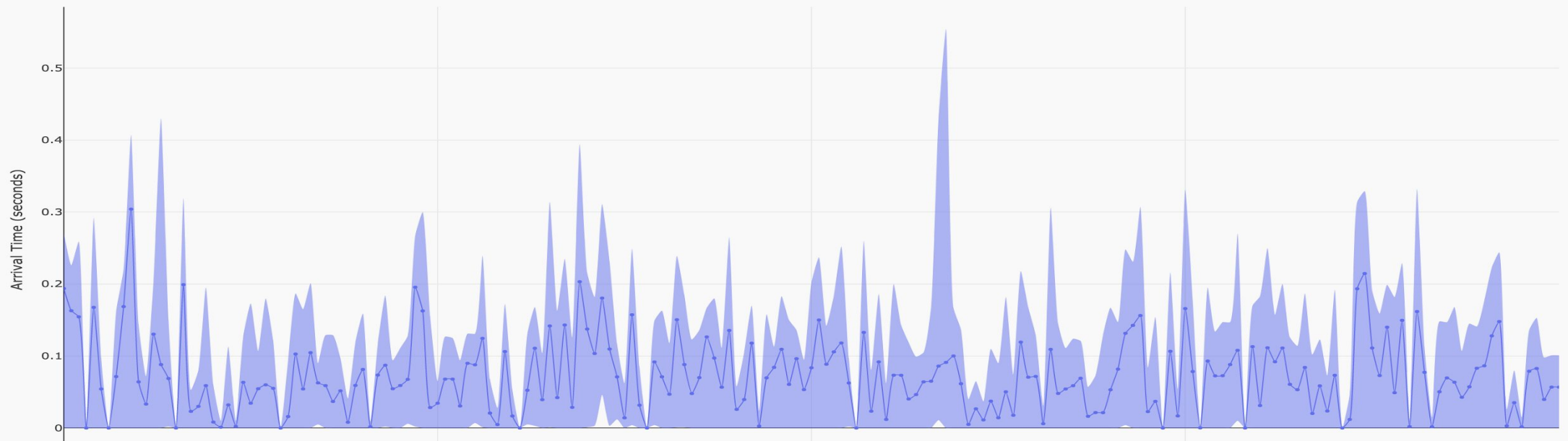


- User specifies their preferences.
- Multiple containers are automatically created.
- Containers are sent to hosts for the network to be deployed.

- In an effort to both test the usability of the tool that was developed as well as showcasing a way to use it, two experiments were conducted.
- In the first one a network was deployed using multiple hosts, everyone acting as a different node, in order to resemble the real Bitcoin network.
- In the second one Docker's ability to deploy multiple containers per host was leveraged, in order to deploy a multi-node network in a single host.
 - This is the most beneficial use case in small-scale experimentation.



- 20 machines with 4-core CPUs clocked at 2.1Ghz and 4GB of RAM.
- Mean Block arrival time: 34.4 ms.
- Maximum arrival time: 240 ms.
- Speed: 350 Mbps



- Same machine as before.
- Mean Block arrival time: 99 ms.
- Maximum arrival time: 554 ms.
- Speed: 0.5 Mbps

⚠ : Given that this is a local network, a special image was used in order to apply network throttling, to better simulate the real world network.

Conclusions

- The initial goal was achieved.
- Through containerization it is possible to create a full node in almost any hosting operating system, with very little to no needed configuration at all.
- The CLI that was created bundles this complexity into an easy to use tool.
- **BONUS:** By running each node inside a container, a security breach of the software will not harm the hosting machine.

Suggestions for Future Work

- Graphical User Interface
 - Easier to use
 - Addition of analytics, graphs etc
- Support for Network-wide commands
 - For example, collect all log files to a single place
- Experiment Continuation
 - Multi-host & Multi-node
 - Different throttling per host
- Support for Different Blockchains

Acknowledgements

- Special thanks go to:
 - Athena Vakali, Professor
 - Georgios Vlahavas, PhD candidate

Thank you for your attention!

Questions?



- CLI module:
npmjs.com/package/@iamnapo/btc-network
- Docker images created:
hub.docker.com/r/iamnapo/btc-network
- Source code of the project (& binaries):
github.com/iamnapo/btc-network