Simulating Hyperledger Networks with Shadow

Learning Objectives of the project:

- Put into practice writing real software in the programming languages they have learned.
- Gain a deep understanding of how the Hyperledger blockchains are organized at the network level.
- Learn about consensus networks and how the nodes must collaborate and corroborate to always come to the same conclusion.
- Experience working closely with an open source working group to implement a tool they can use to run experiments with.

**(Note: Learning objectives taken by official project website of Hyperledger community)*

Ahmet Turkmen
110510016
Veysel Karani Pehlivan
110510007

Introduction

Shadow is a discrete event simulator which means that we can make experiments on it by writing integrated plug-ins. It runs in parallel, so creating multiple threads for handy operation is needed. Otherwise %100 percentage of the CPU consumed for one core of the computer. Since its core functionalities originally written in C and C++, it is very fast. The communication of apps and simulation environment of shadow done by defined plug-in callback functions. The application which needs to be simulated causes events, then those events intercepted by Shadow after several system calls. In Shadow, we are creating virtual nodes and networks to for different environment effects such as path loss, bandwidth, CPU frequency, geo location, latency, routing and further preferences. When the nodes created, and scenario planned to the XML files, interactions in or out of the network logged. The logs are used to extract some statistical information about the scenario. Shadow simulation engine can be summarized given figure below.

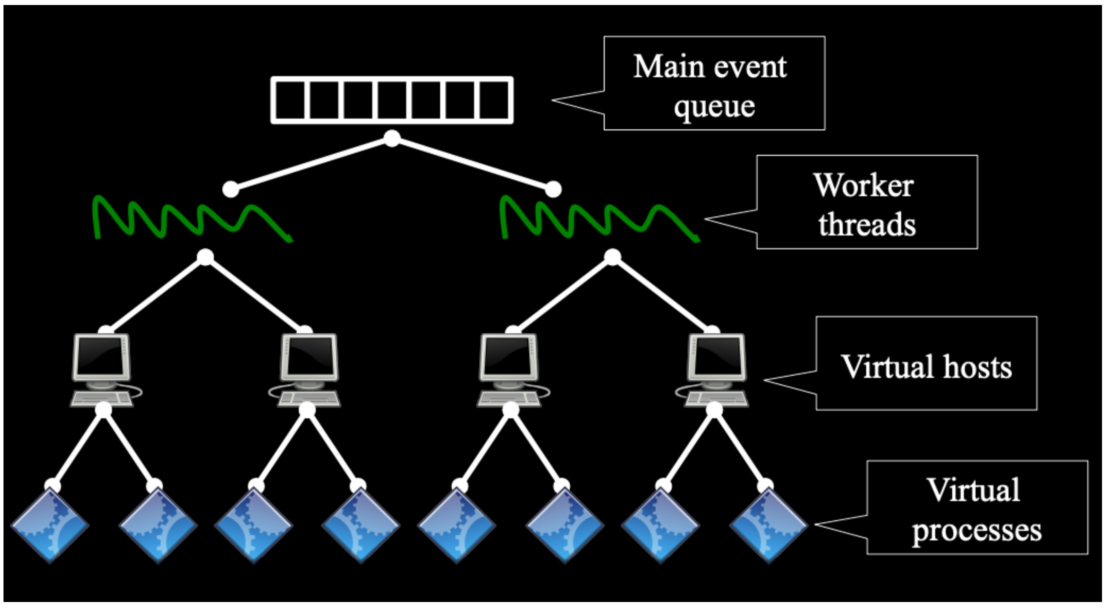
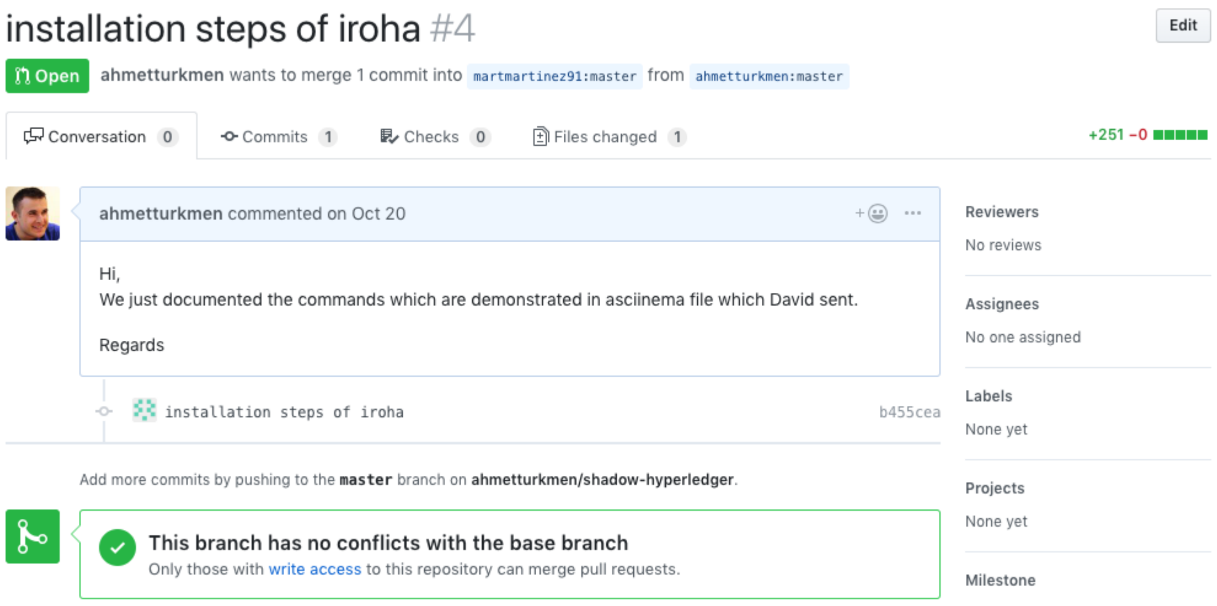


Figure 1 Simulation Engine of Shadow

In similar approach, Shadow can be applied for IROHA network, which is an Hyperledger framework for creating assets, transactions and validating documents. In other words, before implementing IROHA network in real life, we can plan the scenario on Shadow and analysis the results of the situation that we have.
Since Shadow is still under development, there is lack of resources for implementing it to various areas. In this perspective, we have started to contribute this Open Source project to make it better.

Experimental techniques and methods

We have started this project by contacting David Houseby, who is Senior Security Maven at Hyperledger. At the first stage of the project, as David requested to prepare documentation for installation of Iroha to Debian 9.0 from saved terminal session which is attached by David to mail. We started to build required environment for the Shadow and Iroha, besides preparing installation steps of Iroha. It actually took longer time than we have estimated because of missing libraries of C and changed library download links. When we done with it, we have made pull request to repository of Shadow, as you can observe from the image below.



installation steps of iroha #4

ahmetturkmen wants to merge 1 commit into martmartinez91:master from ahmetturkmen:master

ahmetturkmen commented on Oct 20

Hi,
We just documented the commands which are demonstrated in asciinema file which David sent.

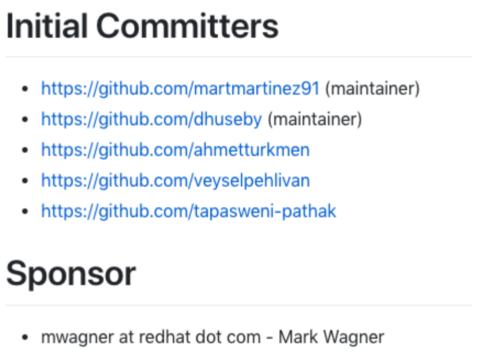
Regards

installation steps of iroha b455cea

Add more commits by pushing to the master branch on ahmetturkmen/shadow-hyperledger.

This branch has no conflicts with the base branch
Only those with write access to this repository can merge pull requests.

After a while, the repository of the project officially opened under Hyperledger community and we have added as initial committers of the project. *(https://github.com/orgs/hyperledger-labs/teams/umbra-committers)



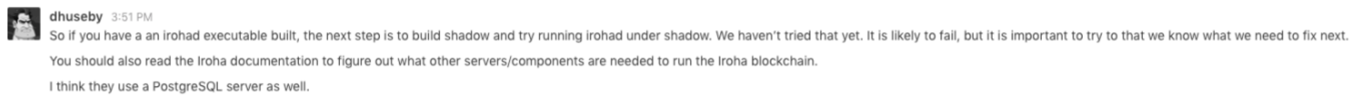
Initial Committers

- https://github.com/martmartinez91 (maintainer)
- https://github.com/dhuseby (maintainer)
- https://github.com/ahmetturkmen
- https://github.com/veyselpehlivan
- https://github.com/tapasweni-pathak

Sponsor

- mwagner at redhat dot com - Mark Wagner

Along the way, we have prepared the required environment to run shadow, after this point, we requested our main goal from David and his response was :



dhuseby 3:51 PM
So if you have a an irohad executable built, the next step is to build shadow and try running irohad under shadow. We haven't tried that yet. It is likely to fail, but it is important to try to that we know what we need to fix next.
You should also read the Iroha documentation to figure out what other servers/components are needed to run the Iroha blockchain.
I think they use a PostgreSQL server as well.

Well, we started to search existing plug-ins for shadow to investigate and implement in order to understand what's going on under the hood. Of course, before it, we also dealt with database and its configuration for Debian 9, which was Postgres because it is required to run Iroha.   As we mentioned before, in order to understand the concept of Shadow under the hood, we choose to implement Tor Network on our local Shadow.

In this stage, to gather as much as information from this implementation, first we have to understand what is it and how a Tor network works, let's dive in to it.

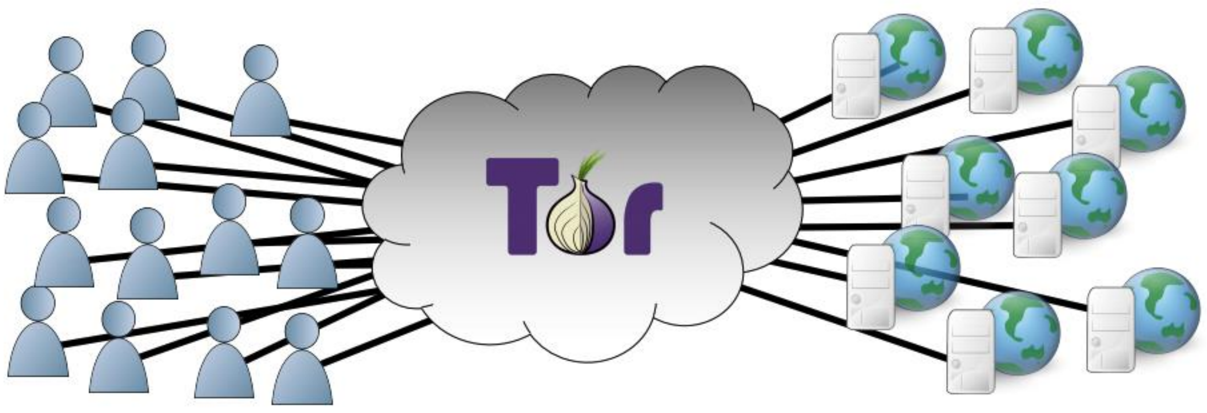


Figure 2 Tor Network Representation

Tor network provides anonymous communication over internet by routing internet traffic to overlay network which contains thousands of relays.

How an anonymous communication work inside Tor; following figure explains very well.

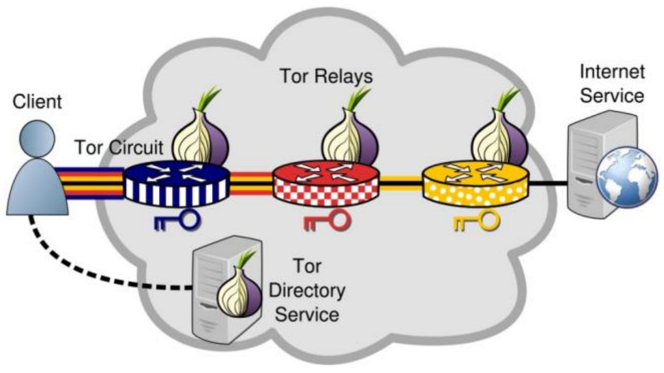
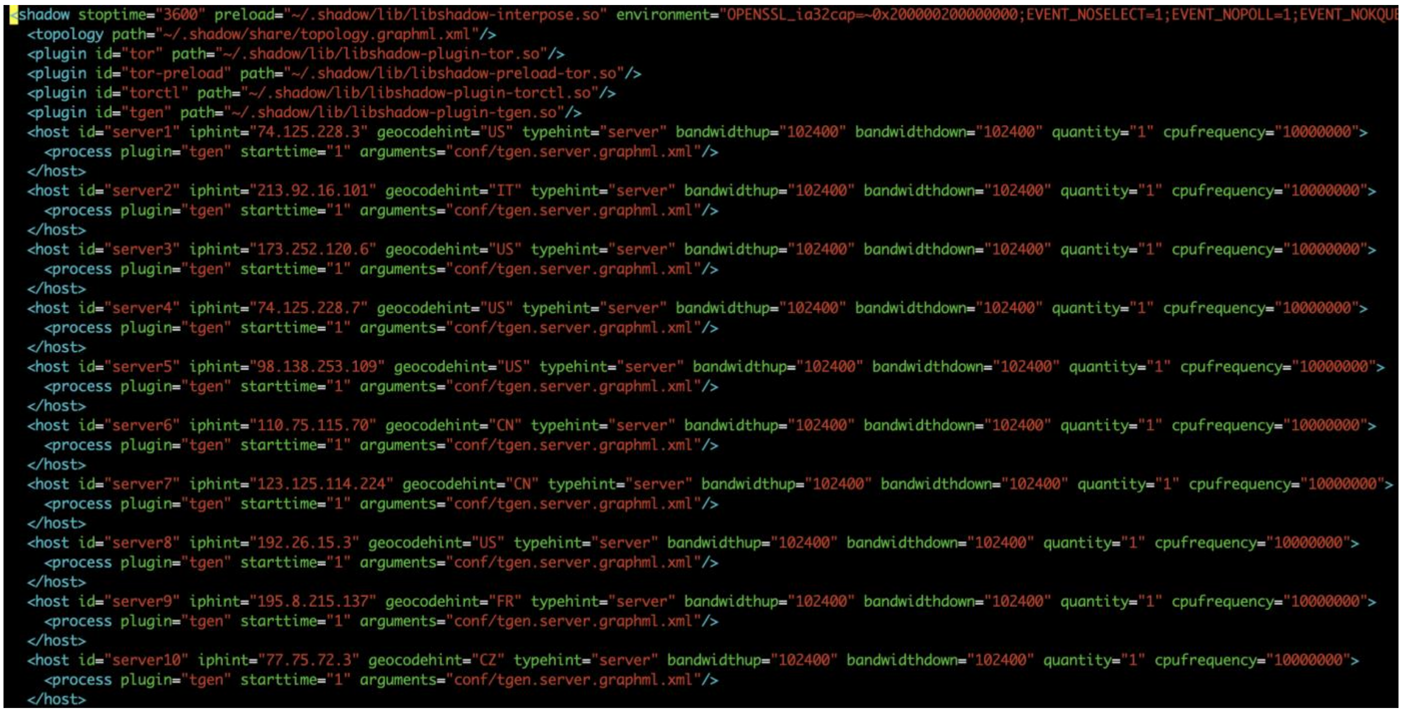


Figure 3 Anonymous Communication

Since Tor Network consists of multi hops, it becomes slow, to prevent delays and unwanted acts such as attacks. We can first fictionalize the scenario then run it on Shadow, then see the results which cannot harm any environment or anyone, since it is all simulated and virtualized. In this manner, an implementation is done on Debian 9, with Vanilla and Priority each of them contains 598 hosts. Settings and parameters of the hosts set to XML configuration files, example snippet is given below.



```xml
<shadow stoptime="3600" preload="~/.shadow/lib/libshadow-interpose.so" environment="OPENSSL_ia32cap=~0x200000200000000;EVENT_NOSELECT=1;EVENT_NOPOLL=1;EVENT_NOKQUE
  <topology path="~/.shadow/share/topology.graphml.xml"/>
  <plugin id="tor" path="~/.shadow/lib/libshadow-plugin-tor.so"/>
  <plugin id="tor-preload" path="~/.shadow/lib/libshadow-preload-tor.so"/>
  <plugin id="torctl" path="~/.shadow/lib/libshadow-plugin-torctl.so"/>
  <plugin id="tgen" path="~/.shadow/lib/libshadow-plugin-tgen.so"/>
  <host id="server1" iphint="74.125.228.3" geocodehint="US" typehint="server" bandwidthup="102400" bandwidthdown="102400" quantity="1" cpufrequency="10000000">
    <process plugin="tgen" starttime="1" arguments="conf/tgen.server.graphml.xml"/>
  </host>
  <host id="server2" iphint="213.92.16.101" geocodehint="IT" typehint="server" bandwidthup="102400" bandwidthdown="102400" quantity="1" cpufrequency="10000000">
    <process plugin="tgen" starttime="1" arguments="conf/tgen.server.graphml.xml"/>
  </host>
  <host id="server3" iphint="173.252.120.6" geocodehint="US" typehint="server" bandwidthup="102400" bandwidthdown="102400" quantity="1" cpufrequency="10000000">
    <process plugin="tgen" starttime="1" arguments="conf/tgen.server.graphml.xml"/>
  </host>
  <host id="server4" iphint="74.125.228.7" geocodehint="US" typehint="server" bandwidthup="102400" bandwidthdown="102400" quantity="1" cpufrequency="10000000">
    <process plugin="tgen" starttime="1" arguments="conf/tgen.server.graphml.xml"/>
  </host>
  <host id="server5" iphint="98.138.253.109" geocodehint="US" typehint="server" bandwidthup="102400" bandwidthdown="102400" quantity="1" cpufrequency="10000000">
    <process plugin="tgen" starttime="1" arguments="conf/tgen.server.graphml.xml"/>
  </host>
  <host id="server6" iphint="110.75.115.70" geocodehint="CN" typehint="server" bandwidthup="102400" bandwidthdown="102400" quantity="1" cpufrequency="10000000">
    <process plugin="tgen" starttime="1" arguments="conf/tgen.server.graphml.xml"/>
  </host>
  <host id="server7" iphint="123.125.114.224" geocodehint="CN" typehint="server" bandwidthup="102400" bandwidthdown="102400" quantity="1" cpufrequency="10000000">
    <process plugin="tgen" starttime="1" arguments="conf/tgen.server.graphml.xml"/>
  </host>
  <host id="server8" iphint="192.26.15.3" geocodehint="US" typehint="server" bandwidthup="102400" bandwidthdown="102400" quantity="1" cpufrequency="10000000">
    <process plugin="tgen" starttime="1" arguments="conf/tgen.server.graphml.xml"/>
  </host>
  <host id="server9" iphint="195.8.215.137" geocodehint="FR" typehint="server" bandwidthup="102400" bandwidthdown="102400" quantity="1" cpufrequency="10000000">
    <process plugin="tgen" starttime="1" arguments="conf/tgen.server.graphml.xml"/>
  </host>
  <host id="server10" iphint="77.75.72.3" geocodehint="CZ" typehint="server" bandwidthup="102400" bandwidthdown="102400" quantity="1" cpufrequency="10000000">
    <process plugin="tgen" starttime="1" arguments="conf/tgen.server.graphml.xml"/>
  </host>
```

Figure 4 Configuration of Shadow XML

As observed from the configuration file above, there are plenty of hosts , precisely 598 hosts , each of them has geo location , IP, id, bandwidth up/down, CPU frequency and tgen XML config file. Before running it plugin ids, paths and topology file path to raise up shadow network and creating defined hosts with specified features.

To be able to run the code and get some logs which will be created for seeing transactions and actions between each other, in order to run it, supplied code below used.

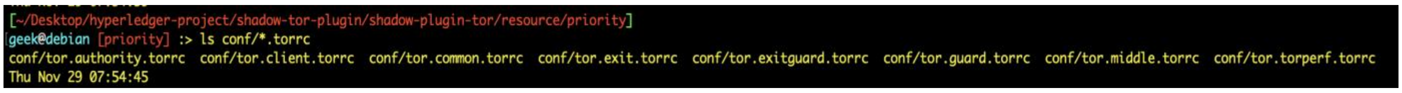
```
cd vanilla
shadow -w 3 shadow.config.xml > shadow.log
```

The code basically says that raise up three threads in computer with given configuration file XML and write the output of conversation between nodes to shadow.log file which will be investigated by parsing and investigating for different metrics such as throughput, recv, goodput and etc. In same manner, similar produce process is done for priority also.

```
cd ../priority
shadow -w 3 shadow.config.xml > shadow.log
```

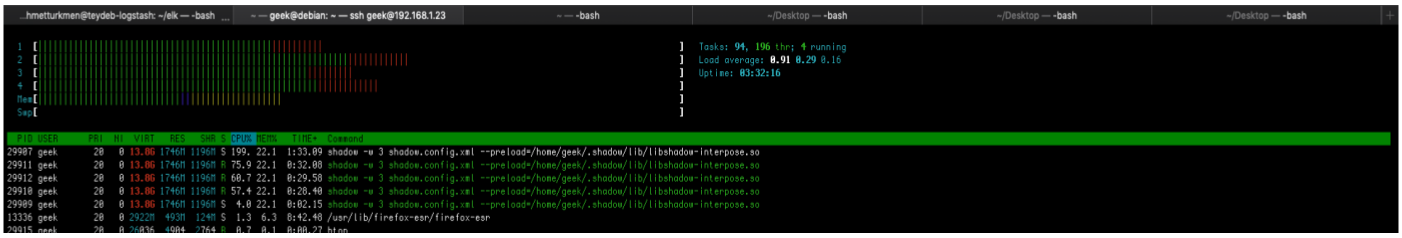
If we have to create our own nodes for a special case, then we can change the parameters and settings in *torrc files which can be found under

```
../priority/conf/*.torcc   #  (* corresponds all files which have torcc extension end of it)
```



```
[~/Desktop/hyperledger-project/shadow-tor-plugin/shadow-plugin-tor/resource/priority]
[geek@debian [priority] :> ls conf/*.torrc
conf/tor.authority.torrc  conf/tor.client.torrc  conf/tor.common.torrc  conf/tor.exit.torrc  conf/tor.exitguard.torrc  conf/tor.guard.torrc  conf/tor.middle.torrc  conf/tor.torperf.torrc
Thu Nov 29 07:54:45
```

If we want to check that the threads are running we can type `htop` to terminal which will display the running processors and threads on computer. Example output of it;



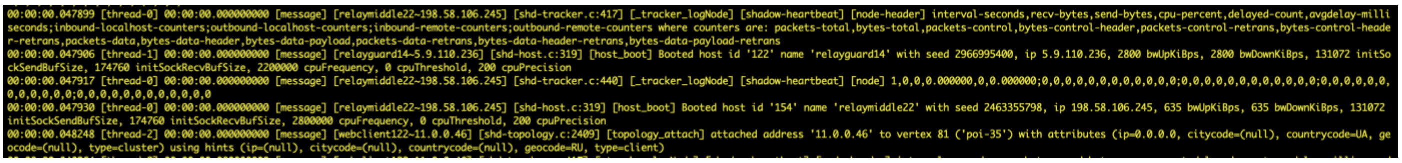
```
...hmetturkmen@teydeb-logstash: ~/elk — -bash   ~ — geek@debian: ~ — ssh geek@192.168.1.23   ~ — -bash   ~/Desktop — -bash   ~/Desktop — -bash   ~/Desktop — -bash

  1  [||||||||||||||||||||||||||||||||||||||||||||||||        ]   Tasks: 94, 196 thr; 4 running
  2  [||||||||||||||||||||||||||||||||||||||||||||||||||||||||]   Load average: 0.91 0.29 0.16
  3  [|||||||||||||||||||||||||||||||||||||||||||||||||||     ]   Uptime: 03:32:16
  4  [||||||||||||||||||||||||||||||||||||||||||||||||||||||  ]
  Mem[||||||||||||||||||||||||||||||||||||||||||              ]
  Swp[                                                        ]

  PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
29907 geek       20   0 13.8G 1746M 1196M S 199. 22.1  1:33.09 shadow -w 3 shadow.config.xml --preload=/home/geek/.shadow/lib/libshadow-interpose.so
29911 geek       20   0 13.8G 1746M 1196M R 75.9 22.1  0:32.08 shadow -w 3 shadow.config.xml --preload=/home/geek/.shadow/lib/libshadow-interpose.so
29912 geek       20   0 13.8G 1746M 1196M R 60.7 22.1  0:29.58 shadow -w 3 shadow.config.xml --preload=/home/geek/.shadow/lib/libshadow-interpose.so
29910 geek       20   0 13.8G 1746M 1196M R 57.4 22.1  0:28.40 shadow -w 3 shadow.config.xml --preload=/home/geek/.shadow/lib/libshadow-interpose.so
29909 geek       20   0 13.8G 1746M 1196M S  4.0 22.1  0:02.15 shadow -w 3 shadow.config.xml --preload=/home/geek/.shadow/lib/libshadow-interpose.so
13336 geek       20   0 2922M  493M  124M S  1.3  6.3  8:42.48 /usr/lib/firefox-esr/firefox-esr
29915 geek       20   0 26036  4904  2764 R  0.7  0.1  0:00.27 htop
```

Figure 5 Running threads

After a while, running threads for priority and vanilla is finished and it created shadow.log files for each of them. A snippet from shadow.log file can be seen like below:



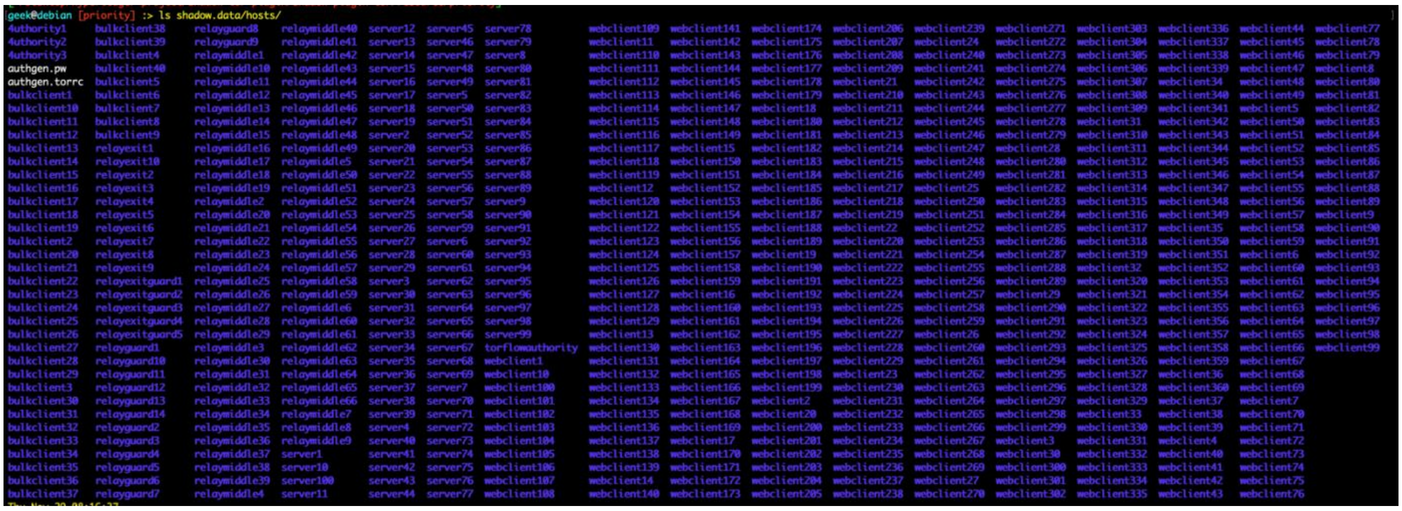
```
00:00:00.047899 [thread-0] 00:00:00.000000000 [message] [relaymiddle22~198.58.106.245] [shd-tracker.c:417] [_tracker_logNode] [shadow-heartbeat] [node-header] interval-seconds,recv-bytes,send-bytes,cpu-percent,delayed-count,avgdelay-milli
seconds;inbound-localhost-counters;outbound-localhost-counters;inbound-remote-counters;outbound-remote-counters where counters are: packets-total,bytes-total,packets-control,bytes-control-header,packets-control-retrans,bytes-control-heade
r-retrans,packets-data,bytes-data-header,bytes-data-payload,packets-data-retrans,bytes-data-header-retrans,bytes-data-payload-retrans
00:00:00.047906 [thread-1] 00:00:00.000000000 [message] [relayguard14~5.9.110.236] [shd-host.c:319] [host_boot] Booted host id '122' name 'relayguard14' with seed 2966995400, ip 5.9.110.236, 2800 bwUpKiBps, 2800 bwDownKiBps, 131072 initSo
ckSendBufSize, 174760 initSockRecvBufSize, 2200000 cpuFrequency, 0 cpuThreshold, 200 cpuPrecision
00:00:00.047917 [thread-0] 00:00:00.000000000 [message] [relaymiddle22~198.58.106.245] [shd-tracker.c:440] [_tracker_logNode] [shadow-heartbeat] [node] 1,0,0,0.000000,0,0.000000;0,0,0,0,0,0,0,0,0,0,0,0;0,0,0,0,0,0,0,0,0,0,0,0;0,0,0,0,0,0,
0,0,0,0,0,0;0,0,0,0,0,0,0,0,0,0,0,0
00:00:00.047930 [thread-0] 00:00:00.000000000 [message] [relaymiddle22~198.58.106.245] [shd-host.c:319] [host_boot] Booted host id '154' name 'relaymiddle22' with seed 2463355798, ip 198.58.106.245, 635 bwUpKiBps, 635 bwDownKiBps, 131072
initSockSendBufSize, 174760 initSockRecvBufSize, 2800000 cpuFrequency, 0 cpuThreshold, 200 cpuPrecision
00:00:00.048248 [thread-2] 00:00:00.000000000 [message] [webclient122~11.0.0.46] [shd-topology.c:2409] [topology_attach] attached address '11.0.0.46' to vertex 81 ('poi-35') with attributes (ip=0.0.0.0, citycode=(null), countrycode=UA, ge
ocode=(null), type=cluster) using hints (ip=(null), citycode=(null), countrycode=(null), geocode=RU, type=client)
```

Figure 6 Transactions/Actions inside defined network

When we done by creating two separated networks by defining its nodes and configs in their subfolders, we may have slightly different results. After that step we should compare performances of each separate network by testing them from different hosts which is given at :

```
priority/shadow.data/hosts
```

The following figure corresponds the hosts for tested network.



```
geek@debian [priority] :> ls shadow.data/hosts/
4uthority1     bulkclient38     relayguard8    relaymiddle40  server12  server45  server78          webclient109  webclient141  webclient174  webclient206  webclient239  webclient271  webclient303  webclient336  webclient44   webclient77
4uthority2     bulkclient39     relayguard9    relaymiddle41  server13  server46  server79          webclient11   webclient142  webclient175  webclient207  webclient24   webclient272  webclient304  webclient337  webclient45   webclient78
4uthority3     bulkclient4      relaymiddle1   relaymiddle42  server14  server47  server8           webclient110  webclient143  webclient176  webclient208  webclient240  webclient273  webclient305  webclient338  webclient46   webclient79
authgen.pw     bulkclient40     relaymiddle10  relaymiddle43  server15  server48  server80          webclient111  webclient144  webclient177  webclient209  webclient241  webclient274  webclient306  webclient339  webclient47   webclient8
authgen.torrc  bulkclient5      relaymiddle11  relaymiddle44  server16  server49  server81          webclient112  webclient145  webclient178  webclient21   webclient242  webclient275  webclient307  webclient34   webclient48   webclient80
bulkclient1    bulkclient6      relaymiddle12  relaymiddle45  server17  server5   server82          webclient113  webclient146  webclient179  webclient210  webclient243  webclient276  webclient308  webclient340  webclient49   webclient81
bulkclient10   bulkclient7      relaymiddle13  relaymiddle46  server18  server50  server83          webclient114  webclient147  webclient18   webclient211  webclient244  webclient277  webclient309  webclient341  webclient5    webclient82
bulkclient11   bulkclient8      relaymiddle14  relaymiddle47  server19  server51  server84          webclient115  webclient148  webclient180  webclient212  webclient245  webclient278  webclient31   webclient342  webclient50   webclient83
bulkclient12   bulkclient9      relaymiddle15  relaymiddle48  server2   server52  server85          webclient116  webclient149  webclient181  webclient213  webclient246  webclient279  webclient310  webclient343  webclient51   webclient84
bulkclient13   relayexit1       relaymiddle16  relaymiddle49  server20  server53  server86          webclient117  webclient15   webclient182  webclient214  webclient247  webclient28   webclient311  webclient344  webclient52   webclient85
bulkclient14   relayexit10      relaymiddle17  relaymiddle5   server21  server54  server87          webclient118  webclient150  webclient183  webclient215  webclient248  webclient280  webclient312  webclient345  webclient53   webclient86
bulkclient15   relayexit2       relaymiddle18  relaymiddle50  server22  server55  server88          webclient119  webclient151  webclient184  webclient216  webclient249  webclient281  webclient313  webclient346  webclient54   webclient87
bulkclient16   relayexit3       relaymiddle19  relaymiddle51  server23  server56  server89          webclient12   webclient152  webclient185  webclient217  webclient25   webclient282  webclient314  webclient347  webclient55   webclient88
bulkclient17   relayexit4       relaymiddle2   relaymiddle52  server24  server57  server9           webclient120  webclient153  webclient186  webclient218  webclient250  webclient283  webclient315  webclient348  webclient56   webclient89
bulkclient18   relayexit5       relaymiddle20  relaymiddle53  server25  server58  server90          webclient121  webclient154  webclient187  webclient219  webclient251  webclient284  webclient316  webclient349  webclient57   webclient9
bulkclient19   relayexit6       relaymiddle21  relaymiddle54  server26  server59  server91          webclient122  webclient155  webclient188  webclient22   webclient252  webclient285  webclient317  webclient35   webclient58   webclient90
bulkclient2    relayexit7       relaymiddle22  relaymiddle55  server27  server6   server92          webclient123  webclient156  webclient189  webclient220  webclient253  webclient286  webclient318  webclient350  webclient59   webclient91
bulkclient20   relayexit8       relaymiddle23  relaymiddle56  server28  server60  server93          webclient124  webclient157  webclient19   webclient221  webclient254  webclient287  webclient319  webclient351  webclient6    webclient92
bulkclient21   relayexit9       relaymiddle24  relaymiddle57  server29  server61  server94          webclient125  webclient158  webclient190  webclient222  webclient255  webclient288  webclient32   webclient352  webclient60   webclient93
bulkclient22   relayexitguard1  relaymiddle25  relaymiddle58  server3   server62  server95          webclient126  webclient159  webclient191  webclient223  webclient256  webclient289  webclient320  webclient353  webclient61   webclient94
bulkclient23   relayexitguard2  relaymiddle26  relaymiddle59  server30  server63  server96          webclient127  webclient16   webclient192  webclient224  webclient257  webclient29   webclient321  webclient354  webclient62   webclient95
bulkclient24   relayexitguard3  relaymiddle27  relaymiddle6   server31  server64  server97          webclient128  webclient160  webclient193  webclient225  webclient258  webclient290  webclient322  webclient355  webclient63   webclient96
bulkclient25   relayexitguard4  relaymiddle28  relaymiddle60  server32  server65  server98          webclient129  webclient161  webclient194  webclient226  webclient259  webclient291  webclient323  webclient356  webclient64   webclient97
bulkclient26   relayexitguard5  relaymiddle29  relaymiddle61  server33  server66  server99          webclient13   webclient162  webclient195  webclient227  webclient26   webclient292  webclient324  webclient357  webclient65   webclient98
bulkclient27   relayguard1      relaymiddle3   relaymiddle62  server34  server67  torflowauthority  webclient130  webclient163  webclient196  webclient228  webclient260  webclient293  webclient325  webclient358  webclient66   webclient99
bulkclient28   relayguard10     relaymiddle30  relaymiddle63  server35  server68  webclient1        webclient131  webclient164  webclient197  webclient229  webclient261  webclient294  webclient326  webclient359  webclient67
bulkclient29   relayguard11     relaymiddle31  relaymiddle64  server36  server69  webclient10       webclient132  webclient165  webclient198  webclient23   webclient262  webclient295  webclient327  webclient36   webclient68
bulkclient3    relayguard12     relaymiddle32  relaymiddle65  server37  server7   webclient100      webclient133  webclient166  webclient199  webclient230  webclient263  webclient296  webclient328  webclient360  webclient69
bulkclient30   relayguard13     relaymiddle33  relaymiddle66  server38  server70  webclient101      webclient134  webclient167  webclient2    webclient231  webclient264  webclient297  webclient329  webclient37   webclient7
bulkclient31   relayguard14     relaymiddle34  relaymiddle7   server39  server71  webclient102      webclient135  webclient168  webclient20   webclient232  webclient265  webclient298  webclient33   webclient38   webclient70
bulkclient32   relayguard2      relaymiddle35  relaymiddle8   server4   server72  webclient103      webclient136  webclient169  webclient200  webclient233  webclient266  webclient299  webclient330  webclient39   webclient71
bulkclient33   relayguard3      relaymiddle36  relaymiddle9   server40  server73  webclient104      webclient137  webclient17   webclient201  webclient234  webclient267  webclient3    webclient331  webclient4    webclient72
bulkclient34   relayguard4      relaymiddle37  server1        server41  server74  webclient105      webclient138  webclient170  webclient202  webclient235  webclient268  webclient30   webclient332  webclient40   webclient73
bulkclient35   relayguard5      relaymiddle38  server10       server42  server75  webclient106      webclient139  webclient171  webclient203  webclient236  webclient269  webclient300  webclient333  webclient41   webclient74
bulkclient36   relayguard6      relaymiddle39  server100      server43  server76  webclient107      webclient14   webclient172  webclient204  webclient237  webclient27   webclient301  webclient334  webclient42   webclient75
bulkclient37   relayguard7      relaymiddle4   server11       server44  server77  webclient108      webclient140  webclient173  webclient205  webclient238  webclient270  webclient302  webclient335  webclient43   webclient76
```

Figure 7 Hosts to test metrics of configured network under Shadow

To extract traffic behaviors from given data, Tgen is used which provides traffic behavior of network with help of action-dependency graph, which is stored under graphml.xml. Action dependency graph basically describes how a network should be designed, it has **vertices** , **edges** and **vertex** attributes and these are corresponds **actions**, **dependencies** and **parameters** of the network in sequence. TGen is an module which is written in C. So in order to observe traffic behaviors of the supplied data we have tried following code in order. In vanilla directory following codes implied.

```
cd vanilla
python ~/shadow/src/tools/parse-shadow.py --prefix vanilla-results shadow.log
python ~/shadow/src/tools/parse-tgen.py --prefix vanilla-results shadow.data/hosts
```

In priority directory following command implied.

```
cd ../priority
python ~/shadow/src/tools/parse-shadow.py --prefix priority-results shadow.log
python ~/shadow/src/tools/parse-tgen.py --prefix priority-results shadow.data/hosts
```

After given commands created results from given data, we ran compare and plot code in order to compare results.

```
python ~/shadow/src/tools/plot-shadow.py --prefix "scheduler" --data vanilla-results/ "vanilla" --data priority-results/ "priority"
```

The plotted graphs and results given in ***Appendices.***

Outcomes of this experiment can be listed as follow:

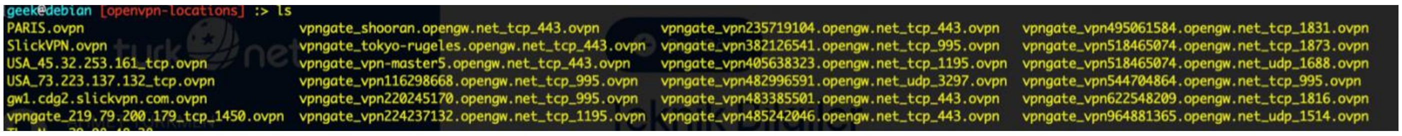
- Learnt how to place nodes configuration in XML files which are given as parameter to Shadow
- Learnt how to deal with C shared libraries in order to bypass errors
- Learnt how to get experimental simulation and its results.

- Learnt how is the plugin characteristics for integrating Shadow.

Challenges

There is no doubt that having first researchers for a new started project is compeller. Along the way, we have faced many failures and bugs which sourced by either operating system or not up to date OS or network failures such as ICA*(BTK) blocks tor repositories to access. These issues created problems during the development and research, but we had to find solutions to them.

Beginning of the implementation of Tor network under shadow, we should install required dependencies and libraries from torproject.org website however since it is blocked in Turkey. We had to build an OpenVPN on Debian 9 and wrote a script which scans provided files who has ***.ovpn** extension and try to connect them. Used and tried ovpn files given in figure below, note that these are used to have access to torproject.org repositories in order to download its required libraries and shared libs.



```
geek@debian [openvpn-locations] :> ls
PARIS.ovpn                          vpngate_shooran.opengw.net_tcp_443.ovpn       vpngate_vpn235719104.opengw.net_tcp_443.ovpn    vpngate_vpn495061584.opengw.net_tcp_1831.ovpn
SlickVPN.ovpn                       vpngate_tokyo-rugeles.opengw.net_tcp_443.ovpn vpngate_vpn382126541.opengw.net_tcp_995.ovpn    vpngate_vpn518465074.opengw.net_tcp_1873.ovpn
USA_45.32.253.161_tcp.ovpn          vpngate_vpn-master5.opengw.net_tcp_443.ovpn   vpngate_vpn405638323.opengw.net_tcp_1195.ovpn   vpngate_vpn518465074.opengw.net_udp_1688.ovpn
USA_73.223.137.132_tcp.ovpn         vpngate_vpn116298668.opengw.net_tcp_995.ovpn  vpngate_vpn482996591.opengw.net_udp_3297.ovpn   vpngate_vpn544704864.opengw.net_tcp_995.ovpn
gw1.cdg2.slickvpn.com.ovpn          vpngate_vpn220245170.opengw.net_tcp_995.ovpn  vpngate_vpn483385501.opengw.net_tcp_443.ovpn    vpngate_vpn622548209.opengw.net_tcp_1816.ovpn
vpngate_219.79.200.179_tcp_1450.ovpn vpngate_vpn224237132.opengw.net_tcp_1195.ovpn vpngate_vpn485242046.opengw.net_tcp_443.ovpn   vpngate_vpn964881365.opengw.net_udp_1514.ovpn
```

Figure 8 OPENVPN accces files

After we have access to tor project repository, we have downloaded required ones and uploaded them to our Gitlab account.

Another one on challenge part was modifying existing Python scripts because some of them has out-dated packages and links which are blocked in Turkey. We have re-configured Python scripts in order to run them correctly and it also took some time.

Since the project is under beta version and there is lack of documentation about Shadow and its plugins, it is hard to find a way to implement good approaches. Nevertheless, we are investigating Python scripts to make further development on them. Some other challenges can be summarized as seen below.

- Proficient C/C++ knowledge required
- Debian configuration should be completed according to needs.
- Parallel programming skills need to be at good level.

Technologies

In further steps of this project, we will try to implement Iroha framework under Shadow, Iroha is a blockchain platform which uses unique chain-based Byzantine Fault Tolerant consensus algorithm and BFT ordering service. Now, we have installed required libraries and database to Debian 9.0, it might be easy think to do however if you do not have any C programming language experience then it may come handy to you. Iroha's core functionalities mainly written in C++ and some features written in different programming languages such as Pyhon, Javascript and Java. It uses Postgresql database as its main database.   Our main tech is basically Hyperledger tech which is kind of umbrella for blockchain based open source projects. The validations of nodes on Iroha distributed half-signed transactions over the Gossip protocol. Used tools and programming languages for this project up to now, can be summarized as following:

- Python programming language to interact with nodes in client side
- C programming language is used to build shared libraries on Debian 9.0 in order to have running platform.
- XML config files and its integration with running Python script; writing nodes and their specs to the XML files.
- Linux skills are used during building C files and adjusting paths of the libraries.

We have just started to implement Iroha on local Debian 9.0 machine, by that we will create some transactions and validations, if required a Python script will be written to automate the process. So, in this point we will try to catch very-well understanding of Iroha under the hood, we will try to write appropriate plugin in order to run it under Shadow.

Since it is hard to keep local computers alive 7/24, we just rented a server on Google Cloud, in further steps, all studies will be moved to Cloud.

The advantages of having studies on Cloud will be ;

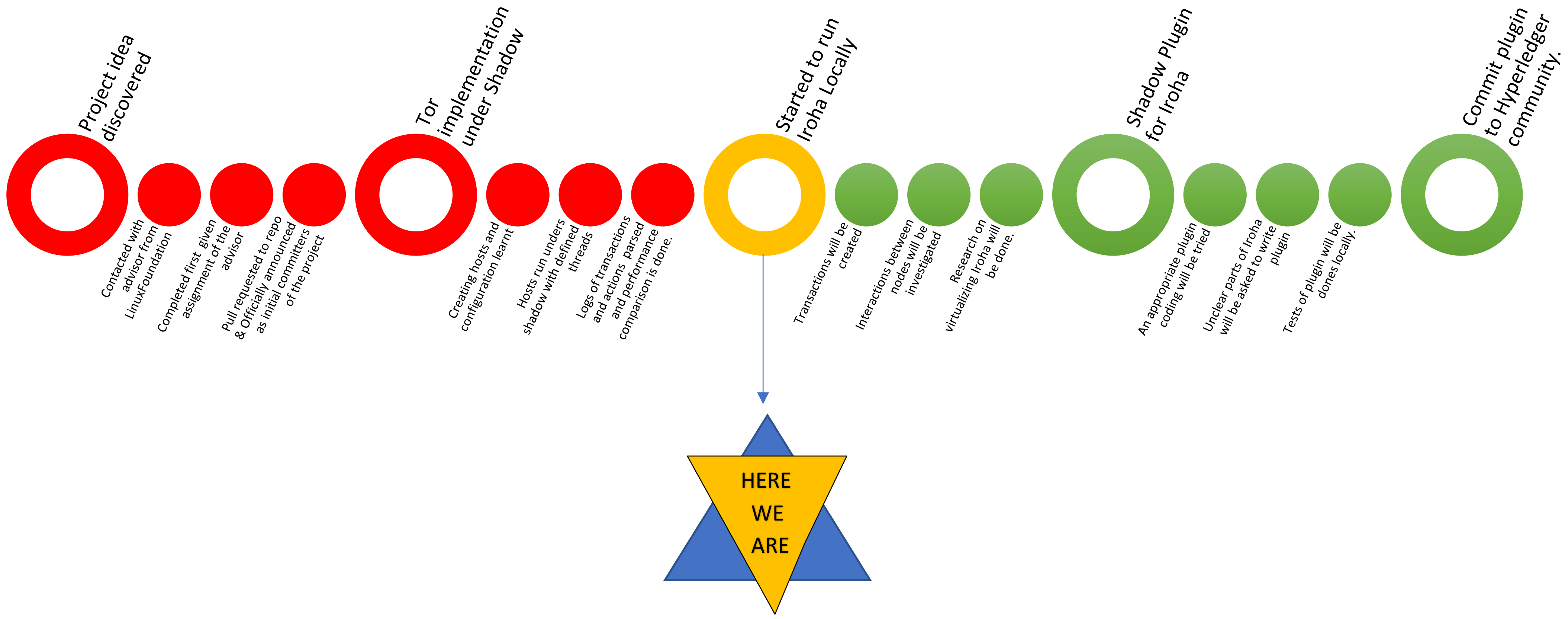
- 7/24 Operational
- No data loss
- Easy access with static IP
- Do not need to rebuild libraries for every other computers

Roadmap

The estimated road map of this project is given in next page, the planned studies are estimated so they may change time to time.

THE LINUX FOUNDATION

HYPERLEDGER



Project idea discovered
Contacted with advisor from LinuxFoundation
Completed first given assignment of the advisor
Pull requested to repo & Officially announced as initial committers of the project
Tor implementation under Shadow
Creating hosts and configuration learnt
Hosts run unders shadow with defined threads
Logs of transactions and actions parsed and performance comparison is done.
Started to run Iroha Locally
Transactions will be created
Interactions between nodes will be investigated
Research on virtualizing Iroha will be done.
Shadow Plugin for Iroha
An appropriate plugin coding will be tried
Unclear parts of Iroha will be asked to write plugin
Tests of plugin will be dones locally.
Commit plugin to Hyperledger community.
HERE WE ARE

Conclusion

There are a lot of open source projects to contribute and create your own project by drawing inspiration from them. Our idea was, when we decided to start to this project, being visionary and not limited to simple projects to grow up our minds and skills by having conversation with developers and advisors from all around the world who have different backgrounds.

Up to now, we have made several experiments and documentations for open source project in order to contribute. Completed tasks can be summarized as following:

- Installation steps of Iroha re-wrote based on David requests and created pull request to Martin's repo as mentioned before.
- Created proper environment to run Shadow and its dependencies, Debian 9.0 configured with C/C++ libraries
- In order to understand how Shadow works, Tor Network is implemented under Shadow, along the way, connection blocked for Tor project, so we have downloaded OpenVPN configuration files and wrote auto assign Python script from defined path, which basically scans OpenVPN configuration files and connect it, if connection failed, then it tries next one.
- Downloaded Tor dependencies added to FTP server located in home and some of them published to Gitlab account to easily download it from anywhere.
- Some Python scripts of Shadow needed to be modified because of Internet blocks in Turkey.
- The results of implemented Tor network printed out as logs, which contains transactions between nodes, then it is parsed to plot performance analysis of different Tor networks. *(Vanilla & Priority)
- Iroha is installed to Debian 9.0 in order to explore its features and tech specifications.
- While dealing with implementation Tor network under shadow, we have learned content structure of XML configuration files and how to communicate shadow with Python.
- Server rented from Google Cloud to create Debian 9.0 on cloud and accessing it from anywhere, now we are working on configuring it, besides Iroha.

APPENDIX

- Performance analysis results of Vanilla & Priority under Shadow.

*(*Note : Since the graphs of it kept over 20 pages, we prefer to provide download link of the file, you can download it from this link:* [*https://www.dropbox.com/s/8o3wx5oiqzifx4a/shadow_experiment_results.pdf?dl=0*](https://www.dropbox.com/s/8o3wx5oiqzifx4a/shadow_experiment_results.pdf?dl=0) *)*

THE LINUX FOUNDATION

HYPERLEDGER