

Simulation of Distributed Applications Based on Containerization Technology

Daniil Malevanniy(✉), Oleg Iakushkin, and Vladimir Korkhov

Saint-Petersburg State University,
7/9 Universitetskaya nab., St. Petersburg 199034, Russia
rukmarr@gmail.com, o.yakushkin@spbu.ru

Abstract. The logical structure of the underlying network is often neglected in the design and deployment of new distributed systems and sometimes can come into play only after the system reach the production state. This can lead to the vulnerability of the system against intruders controlling the properties of the network. This paper describes a simulation system designed to analyze the influence of the logical structure and quality of the network service on a distributed systems workflow and their general testing during unstable network operation. This system allows developers to detect network vulnerabilities of distributed systems in advance and simplifies further development with regard to new findings.

Keywords: Modeling · Distributed computing · Containerization

1 Introduction

The slowdown in the rate of increase in the performance of computing resources makes developers look for new ways to improve the performance of their products [1]. One of the ways that can provide the desired effect, is the distribution of the workload on several machines. Systems based on this principle are called distributed. As the performance and reliability of the global network increases, distributed systems are becoming more common [2,9,14].

Regardless of how the load is partitioned in such systems, individual machines, or "nodes", of which the system consists, will use network connections to organize work. However, often the development of such systems does not take into account the logical structure of the network on top of which they operate: all nodes are assumed to be directly connected. This approach simplifies development process but can guarantee the correct operation of the application only under normal network conditions and does not allow assessing the impact of networking on the application.

For decentralized services built on the Ethereum platform, it was shown that changing networks between nodes can affect the ratio of their performances in the Ethereum network [3]. And for another distributed system, Bitcoin cryptocurrency, there is already evidence of the successful attacks based on manipulations with the logical structure of the global network in which the system operates [4].

© Springer Nature Switzerland AG 2019
S. Misra et al. (Eds.): ICCSA 2019, LNCS 11620, pp. 587–595, 2019.
https://doi.org/10.1007/978-3-030-24296-1_47

In order to test the operation of distributed systems in case of violations in the functioning of the network and to analyze the influence of the network structure on such systems, we developed a simulation system based on containerization technology. It simulates the operation of the system nodes and network, over which it is deployed, and allows you to dynamically manipulate the quality of the network. To allow access to information obtained during testing, we implemented a monitoring system that visualizes the data generated by the model in real time, taking into account the network structure and allowing to correlate changes in the network with changes in system behavior and immediately plan through the future direction of testing.

As a demonstration of the work of the created simulation system, we analyzed the behavior of DHT Kademlia nodes during network disruption, during which we discovered a temporary slowdown of the entire network operation after one node disconnection and the inability to reconnect the node after a brief interruption of its connection with the rest of the network.

2 Related Works

There are a large number of specialized simulation frameworks for p2p networks, such as NS-3 [5] and PeerSim [6], providing users with ample opportunities for modeling distributed systems, but requiring users to carefully study the documentation and conduct complex adaptation of the tested systems for use in the model. This significantly complicates the comparative testing of several technologies.

Other p2p network simulation systems, such as PeerfactSim.KOM [7] and BLOCKBENCH [8], are focused on benchmarking different solutions and do not allow the model parameters to be dynamically changed, and therefore not suitable for testing the stability of distributed systems.

The VIBES project is the closest in terms of methods and objectives [10], but it is aimed solely at simulating the operation of systems based on blockchain technology and cannot be used to study the operation of other distributed systems, for example, DHT or computing cluster managers.

3 Approach

The purpose of this work is to implement a system that provides users with the opportunity to study the behavior of realistic simulation of a distributed system nodes operation and the network interactions between them.

To build realistic simulations of distributed systems, it may be necessary to simultaneously simulate the operation of a large number of nodes. The simulation system must take this into account and minimize the resources overhead required to simulate the operation of individual node.

Virtual networks connecting the individual nodes of the system in their behavior and logical structure should be close to real networks. For this, system should simulate the operation of dynamic routing protocols that ensure the

operation of a real global network [11] and provide tools for describing the logical structure of virtual networks.

In addition to observing the stable operation of simulated distributed systems, it is of interest to study the influence of networks outages on them. The system should provide this capability, allowing users to dynamically change the quality of individual network parts.

However, dynamic changes in the state of network make little sense if the user does not have the opportunity to inspect their impact on the operation of the simulated system. Therefore, a monitoring system is needed that displays the state of the system in real time.

Besides that, modeling the behavior of real large-scale networks may require the creation of highly complex virtual networks. In order to ensure the visibility of the influence of the structure and state of the network at the same time, the monitoring system should take into account the structure of the network when visualizing the state of the system.

Finally, to demonstrate the capabilities of the created modeling system, an example of its work is needed, which uses a widespread and easy-to-understand technology as a simulated application.

4 Results

4.1 Simulation

Network Simulation. For a complete simulation of a distributed application, it is crucial to simultaneously simulate all its components and their network interactions. To build such a model, Kathara framework and Docker containerization platform were used. Based on the description of the network structure, Kathara creates its logical structure, launches docker containers corresponding to the network nodes, creates virtual networks and performs their initial configuration. Thus, based on the description of its structure, a minimally functional network model is built.

In this form, however, this model has few applications, since the networks created by Kathara represent only the physical structure of the network and do not define the logic of packet routing. For the final configuration of the network model, standard Linux utilities, such as ifconfig and route, was used in combination with Quagga suite, which provides a set of implementations of various dynamic routing protocols. Configuring routing logic creates a single logical network model.

To complete the functionality of the model, it is necessary to include nodes with the simulated application in the overall network structure. The only Kathara requirements for using containers as such nodes are the availability of the bash command shell and ifconfig and route commands. These requirements are not always met in basic docker images, but this is easily fixed since all the necessary packages are widely available in package managers.

Network Management. For a more in-depth study of simulated applications, the ability to flexibly adjust the quality of the simulated network is required. For this, the tcconfig package was used, which controls the quality of operation of virtual network interfaces using the Linux tc (transport control) utility. To provide a convenient user interface to console commands from this package, a web form was developed for setting network quality parameters. It allows users to limit the bandwidth and latency of the network interfaces of the container and the percentage of damage, loss, rearrangement or duplication of packets passing through them.

Additional Instruments. Despite the possible usefulness of such models of distributed applications, the format for describing the network structure accepted by Kathara has a drawback—to describe networks even with the simplest dynamic routing, the user must manually configure all the details of the dynamic routing protocol and distribute the IP addresses of subnets and nodes. This makes the change of the network topology of the model extremely time-consuming, which is unacceptable when studying the influence of topology on the operation of the simulated application. To solve this problem, a software interface was created that allows you to describe the network in terms of nodes, connections between groups of nodes and zones of dynamic routing, and then save it in the Kathara lab format. At the moment, the interface supports RIP, OSPF, and BGP protocols, and allows you to select a docker image running on a host, open access to the outside world, and publish ports to the host machine. In this way, access is provided to the main docker container options necessary for the operation of the model and its monitoring.

The selected combination of software allows you to create detailed network topologies from abstract descriptions, model the networks built with these topologies, deploy distributed applications on top of them and change the quality of the network service.

4.2 Monitoring

Monitoring System API. A running model, isolated from the outside world, can hardly be of any benefit when developing or testing an application. Obviously, to obtain useful conclusions from the model, you need to somehow monitor its condition. For this purpose, a web-based model monitoring interface with a software interface for the nodes of the tested application to report their state was developed.

The programming interface is based on the ZMQ messaging library. Each host creates a ZMQ PUSH socket and assigns it to the address “ipc: /// hostsockets / $ {HOSTNAME}”, where $ {HOSTNAME} is the value of the environment variable HOSTNAME, which is set by Kathara when the container is started. The message format is defined as a JSON object consisting of three fields:

– status—operation status, valid values - an integer from 0 to 3

– traffic—the amount of data processed, any positive integer is valid
– log—output when performing an operation, any string is valid.

It is assumed that the components of the application under test will send messages after each completed logical iteration of their work with the rest of the network. An example of such an iteration is the attempt of the DHT Kademlia client to save and then load a file. Each iteration must check the performance of the application in some way.

Monitoring System User Interface. The messages transmitted through the above-described API contain a large amount of useful information, but they can hardly be called visual. To facilitate the task of analyzing the data obtained, a web interface was implemented to display the structure of the simulated application and visualize the messages received from its components. It is based on the jgraph, a minimalistic JavaScript library designed to demonstrate three-dimensional graphs in a browser. Jgraph was adapted to the needs of the monitoring system, support various shapes of the graph nodes and change in the color or size of the nodes. The structure of the displayed graph is based on the Kathara network configuration. Each message received from the network node is

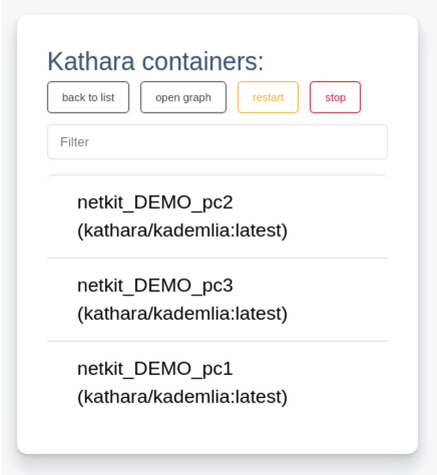

Fig. 1. The initial screen of the monitoring system lists the containers operating within the model and allows you to start and stop the simulation.

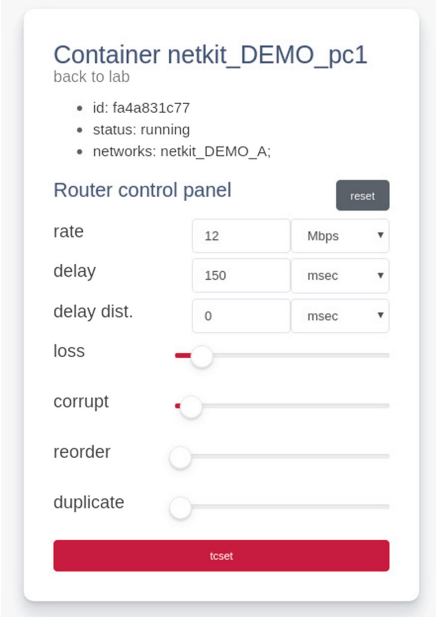

Fig. 2. The container network connection status control panel displays the status of the container and the virtual networks to which it is connected.

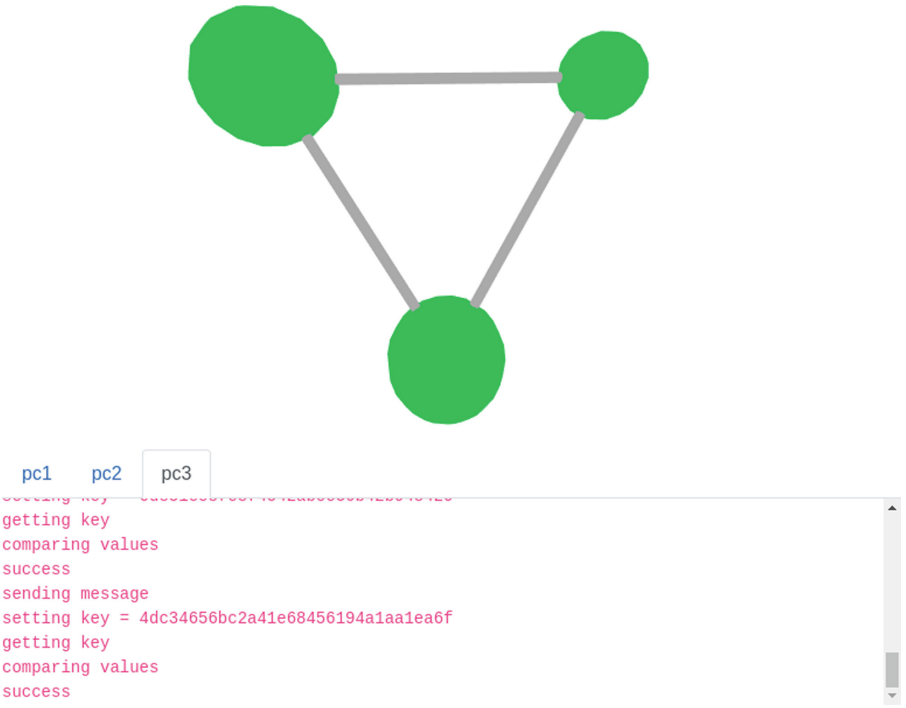

Fig. 3. Monitoring system interface, visualising simulated underlay network structure as a three-dimensional graph. Each node corresponds to a network node, and the edges between them indicate the presence of a physical connection. The green color of the nodes indicates normal operation, and the size corresponds to the proportion of system traffic passing through it. The depicted network consists of three nodes connected with a common collision domain. (Color figure online)

immediately visualized on the graph: the node color corresponds to the status, and the size corresponds to the amount of data transferred. The text part of the message falls into the log window of the corresponding node. Thus, the entire completeness of the information received from the application components finds one or another visualization in the web-interface of the model monitoring.

The web-based monitoring of the model operation allows users to simultaneously monitor the general state of the network and examine in detail the work process of its individual nodes. The use of the ZMQ library, sharpened for high performance, allows you to quickly visualize network changes, even with a large number of nodes in it.

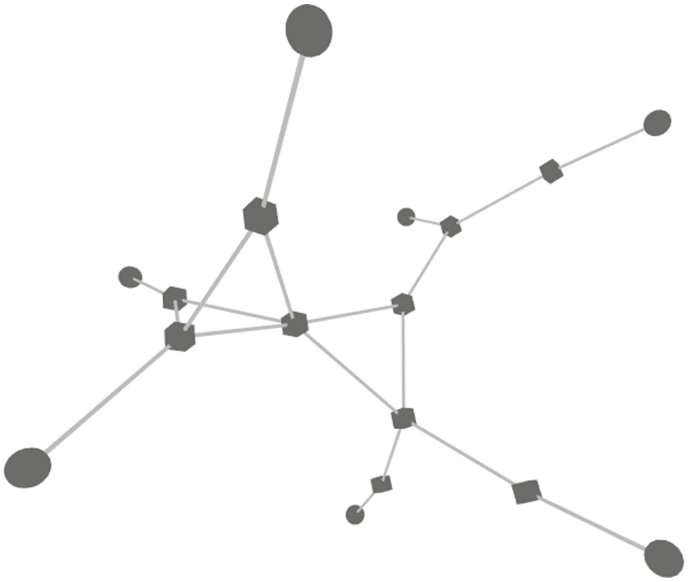

Fig. 4. Visualization of the structure of a more complex network. The round nodes of the graph correspond to the nodes of the distributed system, the square ones - to the routers of the virtual network, which support the work of dynamic routing protocols.

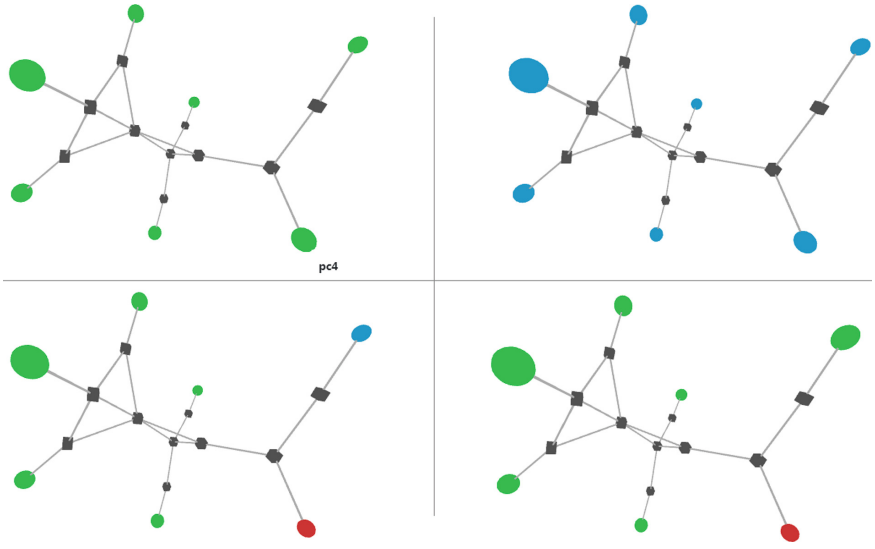

Fig. 5. Visualization of the reaction of Kademlia DHT nodes to the loss of communication with one of them (pc4) in the monitoring system. Consistently: normal network operation, short outage of the whole network, recovery of most nodes, whole network recovery. It should be noted that after the connection of the node to the network was restored, the picture did not change and pc4 could not reconnect to the Kademlia DHT network.

5 Analysis

To demonstrate the capabilities of the created simulation system, an application based on the Python implementation of the Kademlia distributed hash table protocol [13] was deployed. The demo application at the same time provides the network operability and conducts tests of its functionality, storing data in the DHT and then downloading it for comparison. It was wrapped in a docker image and launched on model nodes with different network topologies. During the observation of the model, it was discovered that the server side of this implementation of the Kademlia protocol cannot independently reconnect to the rest of the network after a short of network outage. This example demonstrates how a distributed application simulation system can detect unexpected behavior in the operation of low-level libraries (Figs. 1, 2, 3, 4 and 5).

6 Conclusions

The described modeling system allows a simulation of the operation of distributed applications on different network topologies and with different quality of network operation. This allows developers of such systems to test their work in different conditions, in particular under high loads and poor or unstable network operation. The ability to use dynamic routing increases the realism of the models regarding the operation of distributed applications on top of the real Internet.

The use of containerization technology increases the efficiency of resource utilization relative to virtual machines [12] and allows flexible control over the quality of work of simulated networks using Linux kernel tools. However, the Kathara framework, which is used to configure these networks, was developed at an early stage of technology development and may not fully utilize its potential. It may be wise to consider replacing it with Docker native capabilities.

In addition, despite the completeness of the data presented in the web interface, it is not suitable for in-depth analysis or benchmarking of applications. For the possible implementation of such functionality, it is necessary to think about the organization of storage and access to the data transmitted by the application to the monitoring system.

Simulating the work of a distributed application in a variety of conditions simplifies the life of its developers, and on the other hand, testing different solutions on a real network model can simplify the choice for end users. In addition, the possibility of dynamic changes in the model allows us to study the stability of distributed applications with respect to network attacks.

References

1. Etiemble, D.: 45-year CPU Evolution: one law and two equations. arXiv preprint arXiv:1803.00254 (2018)
2. Skala, K., Davidović, D., Lipić, T., Sović, I.: G-Phenomena as a base of scalable distributed computing —G-Phenomena in Moore's Law. Int. J. Internet Distrib. Syst. **2**(1), 1–4 (2014). https://doi.org/10.4236/ijids.2014.21001
3. Shurov, A., Malevanniy, D., Iakushkin, O., Korkhov, V.: Blockchain network threats: the case of PoW and ethereum. In: Misra, S., et al. (eds.) ICCSA 2019. LNCS, vol. 11620, pp. 606–617. Springer, Cham (2019)
4. Apostolaki, M., Zohar, A., Vanbever, L.: Hijacking bitcoin: routing attacks on cryptocurrencies. In: 2017 IEEE Symposium on Security and Privacy (SP). IEEE (2017)
5. Henderson, T.R., et al.: Network simulations with the ns-3 simulator. SIGCOMM Demonstration **14**(14), 527 (2008)
6. Montresor, A., Jelasity, M.: PeerSim: a scalable P2P simulator. In: Proceedings of the 9th International Conference on Peer-to-Peer (P2P 2009), Seattle, WA, September 2009, pp. 99–100 (2009)
7. Feldotto, M., Graffi, K.: Comparative evaluation of peer-to-peer systems using PeerfactSim.KOM. In: Proceedings of IEEE International Conference on High Performance Computing and Simulation (IEEE HPCS 2013) (2013)
8. Dinh, T.T.A., et al.: Blockbench: a framework for analyzing private blockchains. In: Proceedings of the 2017 ACM International Conference on Management of Data. ACM (2017)
9. Iakushkin, O., Shichkina, Y., Sedova, O.: Petri Nets for modelling of message passing middleware in cloud computing environments. In: Gervasi, O., et al. (eds.) ICCSA 2016. LNCS, vol. 9787, pp. 390–402. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-42108-7_30
10. Stoykov, L., Zhang, K., Jacobsen, H.-A.: VIBES: fast blockchain simulations for large-scale peer-to-peer networks: demo. In: Proceedings of the 18th ACM/IFIP/USENIX Middleware Conference: Posters and Demos (Middleware 2017), pp. 19–20. ACM, New York (2017). https://doi.org/10.1145/3155016.3155020
11. Rekhter, Y., et al.: Application of the border gateway protocol in the internet (1990)
12. Felter, W., et al.: An updated performance comparison of virtual machines and linux containers. In: 2015 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS). IEEE (2015)
13. Maymounkov, P., Mazières, D.: Kademlia: a peer-to-peer information system based on the XOR metric. In: Druschel, P., Kaashoek, F., Rowstron, A. (eds.) IPTPS 2002. LNCS, vol. 2429, pp. 53–65. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45748-8_5
14. Iakushkin, O., Malevanniy, D., Bogdanov, A., Sedova, O.: Adaptation and deployment of PanDA task management system for a private cloud infrastructure. In: Gervasi, O., et al. (eds.) ICCSA 2017. LNCS, vol. 10408, pp. 438–447. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-62404-4_32