Multi-agent tasks scheduling system in software defined networks

2014 J. Phys.: Conf. Ser. 510 012006

(http://iopscience.iop.org/1742-6596/510/1/012006)

Multi-agent tasks scheduling system in software defined networks

P.O. Skobelev[1], O.N. Granichin[2], D.S. Budaev[1], V.B. Laryukhin[1], I.V. Mayorov[1]
[1]SEC "Smart Solutions", Ltd, Samara, Russia
[2]Saint Petersburg State University, Saint Petersburg, Russia

Key words: multi-agent system, real-time scheduling, adaptability, software defined networks.

Abstract: In this paper a multi-agent tasks scheduling system in software defined networks is considered. This system is designed for distribution simulation and tasks implementation on computational resources including network dynamic characteristics and topology.

1 Introduction

Until now, computer networks were generally developing in an extensive way: more and more devices were joined into local and global networks, network equipment was being improved, data communication channels capacity was increasing, and response time of servers and network components was increasing.

However, conceptually new solutions were not widely integrated in practice and gradually, the existing networks could not cope with the increasing number of devices and data flows. The structure of the network schedule changed because each inquiry creates a big amount of references to application servers, databases, other storage and information processing systems. Complexity of the modern network infrastructure and the scheduled growth lead to the increasing number of the rejections and in the course of the current tendencies can cause a deterioration of the speed characteristics of data transfer.

At the same time the existing network architecture is still oriented towards the group of old enough protocols to provide the interaction of complex network devices, which makes the scaling difficult. Adding new devices takes high effort, so does the resetting of commutators, routers and firewalls. The network structure remains static and therefore significant means for its support are needed.

Such innovations, as cloud computing and virtualizing, demand new approaches to the network organization.

In this regard in the last few years there has grown an interest to the new architecture of Software Defined Networks (SDN) that should enable to overcome the outlined crisis [1].

SDN concept defines the paradigm shift in the network architecture when the network management layer is programmed directly and is separated from the direct packet routing. This management transfer provides the abstraction of the basic network for the top layers applications, which allows them to see the network as a logical or virtual entity. The developed protocol OpenFlow (OF) that is now applied in many network devices, allows transferring the functions of network routers management into the central module that is called controller or network operational system [2].

Essentially, controller is an intermediate member that connects real network devices and some applications, realizing equipment operation logic. OpenFlow protocol defines the possibility and the scope of the interaction between the controller and the network equipment. Packet routing functions stay inside the routers that are called forwarders or commutators for the reason that they do not make decisions about the route selection, they simply follow the routing rules set by the controller. The controller (together with the applications) can be regarded as system intellectual center, because the route decisions are made on this level for the packet protocols instead of the packets individually and then the network devices configure via the OpenFlow protocol. The protocol allows providing the complete network presence on the controller, monitoring all the resources and networking visualization that allows creating complex solutions for the network management [3].

In that regard the problem remains up to date and valuable for the simulation and research of different variants of network organization.

2 Problem statement

One of the rapidly developing concepts nowadays is a distributed computing model. It is based on the network access provision to the common computing network resources, which considerably reduce the users expenses on the creation of the corresponding infrastructure and has flexible reaction to the changes of computational requirements. Distributed computing controlled by SDN can be more efficient from the side of the operation speed, as the network can be optimized by multiple special algorithms, considering topology and network dynamic characteristics. One of the computing optimization tasks in the distributed system of the network devices is a load balancing task that is being actively investigated today. In the paper [4] there is considered a games theory application to the task to provide the required level of service in the system of the applications that compete for the resources. A service quality optimization problem in distributed dynamic computing systems [5] is investigated with the help of the theory of probability. It is suggested to calculate the local times of the tasks completion, queue lengths, waiting times and other metrics. There was carried out a resources availability simulation depending on the scheduling algorithms and computing system characteristics. The approach using randomized algorithms is considered in [6]. Load balancing task of the computing network nodes in the stochastic statement is reviewed in [7].

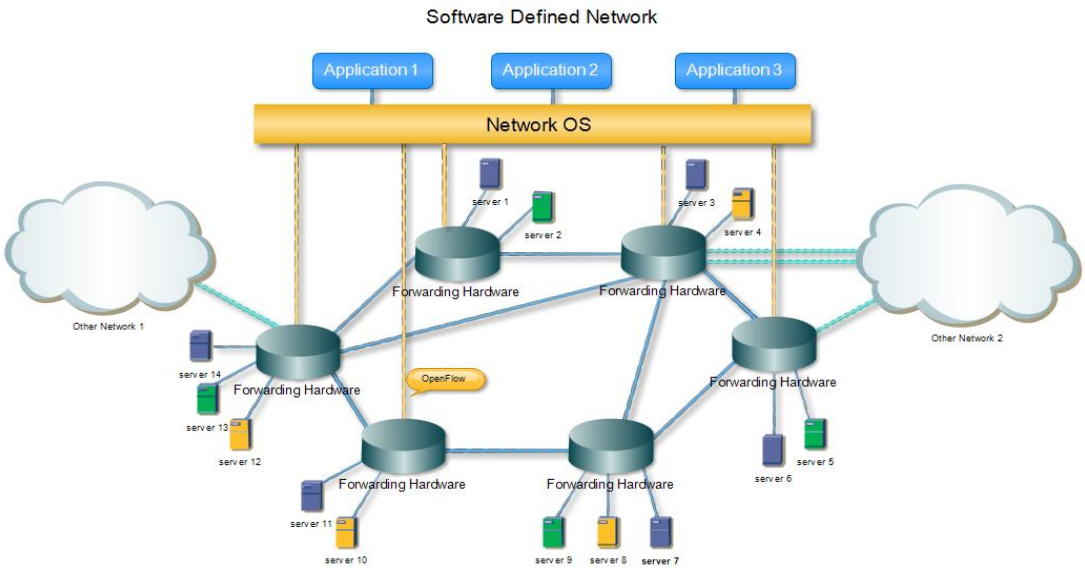

Figure 1 – Software defined network ontology

In the current paper a multi-agent approach to the computing problems scheduling in the distributed system based on the developed model of tasks and resources agents network is considered [8, 9].

3 Solution method and multi-agent scheduling system description

There is a system, consisting of several computational servers that are connected via the OpenFlow commutators to the network, managed by the SDN controller. The servers are designed to complete the tasks of different types with different speed, depending on the task type and resource (server) workload. A task flow enters the system and each one of the tasks can be executed on the bounded subset of servers (Fig. 1).

The compliance of the servers and tasks types is defined in the description of each task. Tasks arrive at any time and redistribute in the system. As a result a tasks queue for the execution is formed on every server. The order of tasks redistribution is defined on the logic layer of the controller. In that case dynamic characteristics of data transfer channels, queue statuses on each network device (commutators, servers) and type compatibility of tasks and servers are considered. The SND controller dynamically defines the most convenient servers for the tasks execution, including the consideration of tasks routes in the system. A dynamic rescheduling can be performed if necessary (for example, in case of unexpected network nodes turn-off). The global goal of modeling is to provide the tasks execution for the shortest time with the balanced load of all the network nodes.

It is suggested to fulfill the dynamic distribution of tasks over the network servers with the multi-agent approach. Basic system entities are associated with the program agent, task agent, resource (server) agent and commutator agent. Agents are described by attributes that include the most essential fields, properties and interaction rules. Each agent has its own goal and aspires to reach it, by sending and receiving messages from other agents. To solve the optimization problems of agents states “virtual costs” are introduced, which allow applying market approach in multi-agent system. At the same time there are changes in the agents attribute values, therefore at each moment of time there exists some state of multi-agent system that is responsible for the local optimum decision.

4 System logical architecture

System consists of the knowledge base – ontology, which includes descriptions of the concepts types: task agents, resource agents and commutator agents. Attributes, properties and relations are set between them. Virtual world scene serves for the agents' instances interaction through the message exchange. Generator of external events creates flows of the tasks entering the system with different static characteristics, for example events of productivity and servers types change, their inaccessibility periods, connection and disconnection of the highlighted network fragments.

User interface subsystem allows starting the events simulation, visualizing network graph, tasks cost dynamics, tasks distribution and queues on servers, transfer channels loading and others.

Task agent

Task agent represents a computing task that needs to be scheduled and executed with the smallest expenses on one of the servers in the SDN network.

At the moment of each task arrival into the system, task agent is created and in this domain area its attributes will be the following fields:

- task arrival time,
- task data volume,
- task type (correlates with the server type),
- task computational complexity, if it is known,
- task priority (high, medium, low), if it is known,
- desirable task starting time, if it is known,
- desirable critical time of task completion, if it is known,
- actual task implementation starting time on server,
- actual task implementation ending time on server,
- current virtual account state.

Every task agent connects with the piece linear function of bonuses/fines, which changes the current virtual account of task agent depending on the correlation of the desirable and actual time, priority and computational cost of the task. Computational

cost and correspondingly the critical ending time represent estimated upper limits that are not certain until the task implementation start on server. Actual values are defined at the start of real execution based on some statistics.

Resource (server) agent

Resource agent provides a server to the system, on which the tasks can be executed. Server agent goal is to schedule as much as possible tasks and get the maximum virtual profit from the tasks execution.

Resource agent attributes in this domain area will be the following fields:

- set of task types that can be executed on this server,
- resource productivity that defines the estimated time of tasks execution,
- current task list that are scheduled on the resource,
- allowable task queue volume,
- resource time unit cost table,
- current virtual resource account state.

Each resource agent associates with a penalty function that depends on the amount of the scheduled tasks on the specified scheduling horizon.

Commutator agent

Commutator agents provide actual information about network state to the scene and they are designed to provide the efficient tasks allocation to the nodes of the further movement.

Commutator agent is described with the following fields:

- allowable task queue volume,
- each connection capacity with the closest commutator agents,
- current task list for the transfer via any connection to commutator agents,
- current virtual commutator account state.

5 System operation

Initially the system has certain topology of connections, for example, ring. Tasks arrive in a random way on one or several commutators from the external networks. When a new task arrives, system automatically performs the scheduling in three stages:

1) Conflict-free scheduling stage

Task agent searches for the resource descriptions in the scene of a certain type, which are the closest from the side of the delivery time through the commutators on its way. The logical entity of the scene of multi-agent system corresponds to the physical entity of SDN controller. As a result of the scene operation, task agent creates a list of the most appropriate resources. Then the agent forms a message with a request to define the conditions of the initial task allocation on the resource taking into account its parameters. This message is sent to all the agents of the chosen resources. Resource agents analyze their schedule and calculate the possibility and cost of task allocation and form counter-offers with the shifted terms concerning the requested ones. As a result for all the sent messages resource agent gets a set of possible tasks allocation variants on the resources including allocation cost, which later on should be written-off from the task virtual account and charged to the chosen resource account. From the existing table of variants task agent chooses the best option by the initially set criteria with their weights.

If the task is successfully allocated to the resource it is fixed by a "contract" (relation) in the scene and task budget updates, becoming equal to the operation cost excluding resource cost. Resource agent also updates its account, increasing the budget on the specified cost and adds a job into the schedule. Times of start, end, lead and delay are being defined.

2) Task proactivity stage

Tasks proactivity consists in the activation of those tasks that are already scheduled, but are not executed. Their position in the resource schedule can be unprofitable from the point of view of the task agent. If task agent has enough amount of virtual money it can dynamically try to reschedule on the other appropriate resource, compensating possible deterioration of other task agents' states in the course of conflicts solving about tasks positions in the schedule.

If there are penalty functions, task agents' proactivity is a process controlled by the difference of the cost in the resource cost table, penalty functions representation that are involved in the task agents' conflict and dependent on a current virtual operations budget. Proactivity stage is designed to form the updated balanced state of the system with the quantitative improvement of tasks condition.

3) Resources proactivity stage and routes definition

Resource proactivity consists in the tasks allocation schedule change that begins by the resource initiative. When the schedule is not concentrated enough or vice versa the queues are too long, when there are events of tasks arrival, the resource can take off the schedule part of low-gain for him tasks. The routes of tasks movement in the network are defined by the choice of the least loaded channels. If it is necessary to transfer the profitable task to the resource and there is a sufficient virtual budget, then new channels are created. Dynamically created channels, that provide local routing optimum, are defined by the operation of the logical virtual layer of software defined network.

6 Advantages and restrictions

The advantages of the suggested multi-agent approach are the following:

- ontological description of the world of computing resources, tasks and network configuration scene allows to introduce new entities of SND and set their characteristics without significant expenses on reprogramming;
- dynamic management of the state of all SDN world scene objects with the help of the scenarios (scripts);
- changes in network topology can be made in real-time mode;
- dynamic tasks scheduling on servers in multi-agent system that is provided by the constant tasks and resources proactivity, on basis of self-organization based on agents aspire to maximize the virtual profit in the interaction acts through the message exchange;
- local optimality and consistency of the current schedule;
- local optimality of packets routing ways and tasks queues;
- the account of changes in the network surrounding and in statistic task flow characteristics in real-time (adaptivity) does not lead to the full schedule redesign, which reduces computational costs of simulation system;
- dynamic traffic redistribution with the account of channels loading;
- simplicity of agents interaction logic based on virtual profit maximization;

The developed system is a platform for simulation and experimentation on the choice of equipment operation logic and program system of network management.

7 Implementation

The system is developed as an OS Windows application using the .NET Framework platform on the programming language C#. Prototype kernel is implemented as a library (build) and the existing simulation logic can be extended with the use of it. User interface development is performed with the WPF technology. To display the network graph a Graph# library was used.

The current system implementation allows visualizing the studied network in the form of graph, executing initial network configuration and defining the parameters of individual devices: volume for task queues for individual commutators and servers, communication channels capacity, processing servers' productivity, devices turned on and off (Fig. 3).

In the course of the modeling process there is possible a modeling time scaling concerning the real, which allows to speed-up or slow down the simulation process at system user's will. Also in the top part of the user interface a system change log is displayed. Each event (new task arrival, task execution on the resource) is recorded in the log and is available for the analysis at any moment of simulation.

The system allows saving the previously formed network configuration to file. Moreover, the system can save the current state of the simulation process itself at any moment of time. At the same time simulation devices save all their current characteristics and in particular lists of all the executed and transferred tasks for each network device. Therefore, user has an opportunity to save the partial results of simulation on different stages and next time continue the simulation from this stage by loading the saved file.

Besides the network configuration, saving and loading options, there is a possibility to form simulation script that defines parameters of the flow of the arriving to the network tasks (tasks distribution rule by time on each device) and individual parameters of the arriving tasks (volume, computational cost, type, priority and others). After the modeling scripts are formed, they can be saved and loaded in future. Therefore, network configurations are separated from the modeling scripts, which allows to carry out a series of various simulation experiments for one or different network topologies. At the moment with the help of the developed system prototype the research of advantages and restrictions of multi-agent approach continues.

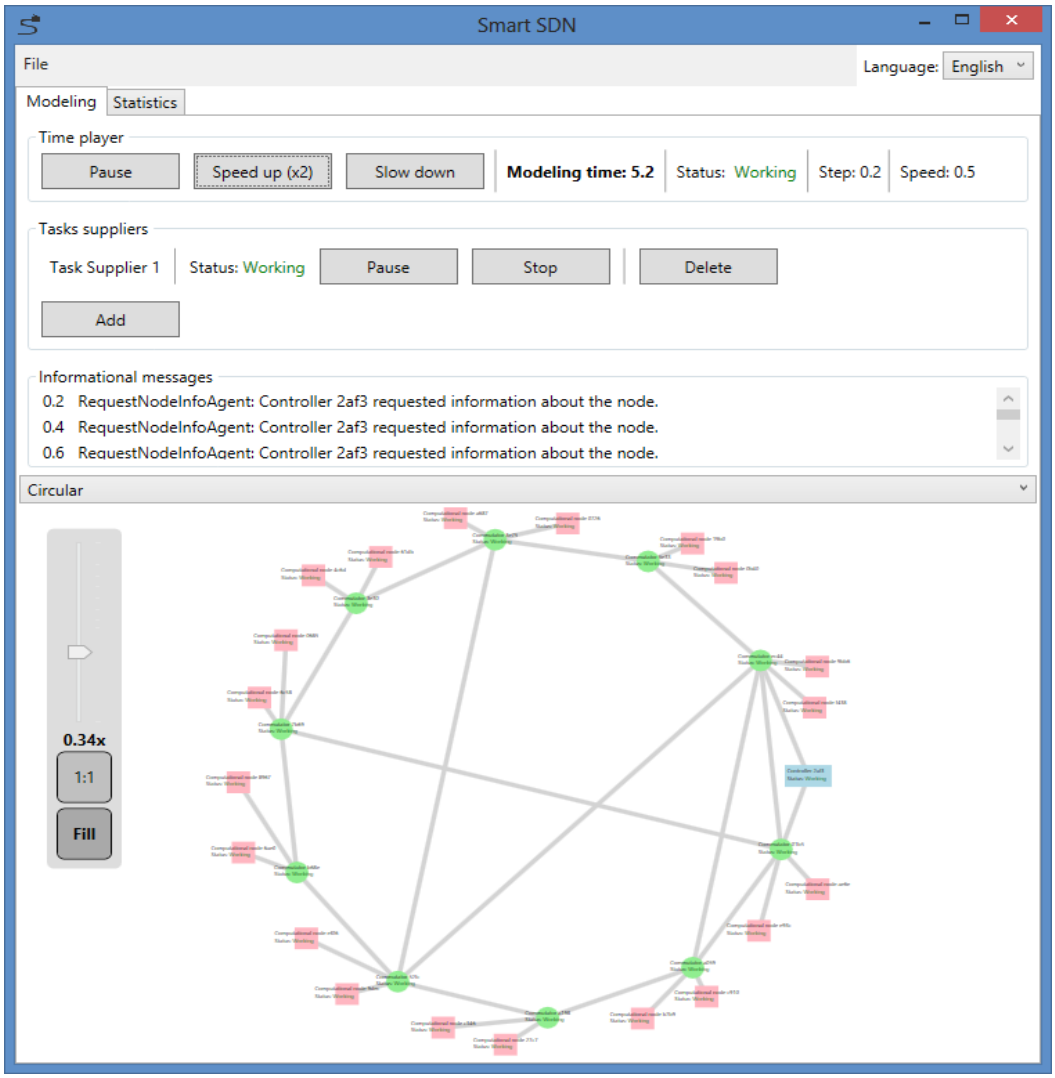

Fig. 3 – System interface

8 Conclusion

The developed multi-agent tasks scheduling system in software defined networks is based on the dynamic improvement of network state by the local optimization of servers and commutators load on the basis of the "market" approach by expressing all of the characteristics for concessions solving by artificial currency. Task, resource and channel agents tend to concordantly raise their virtual profit that should lead to the system characteristics improvement.

In future it is planned to carry out the research of task queues dynamics in SDN, servers load characteristics, channels capacity and the effect of topology change on the given parameters.

This work is performed under the state contract № 14.514.11.4086.

9 References

1. N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. Openflow: Enabling innovation in campus networks // SIGCOMM CCR, vol. 38, no. 2, pp. 69–74, 2008.

2. OpenFlow http://www.openflow.org/documents/openflow-spec-v1.1.0.pdf, last accessed date 19.08.2013

3. N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker. NOX: Towards an operating system for networks // SIGCOMM CCR, vol. 38, no. 3, 2008.

4. Martina Maggio, Enrico Bini, Georgios C. Chasparis, Karl-Erik Årzén, A Game-Theoretic Resource Manager for RT Applications // Proceedings of the 25th Euromicro Conference on Real-Time Systems, Paris, France, July 2013.

5. Zhoujun Hu, Zhigang Hu, and Zhenhua Liu , Resource Availability Evaluation in Service Grid Environment // Proceedings of the Asia-Pacific Services Computing Conference (APSCC,). – IEEE. – 2007. – P. 232-238.

6. Volkovic Ya.V., Granichin O.N., Adaptive server optimization that processes tasks queue // Stochastic optimization in informatics, Vol. 1, p.p. 17-28, 2005.

7. Amelina N.O., Fradkov A.L., Approximate consensus in the dynamic stochastic network with incomplete information and measurement delays // Avtomat. i Telemekh. 2012. № 11. p.p. 6-29.

8. Vittikh V.A., Skobelev P.O. Method of contiguous interactions for real time resource allocation management // Autometry, 2009. Vol. 45, № 2, p. 78

9. Skobelev P.O. Multi-Agent Technology for Industrial Applications: Towards 20 years Anniversary of Samara Scientific School of Multi-Agent Systems // Mechatronics, automation, management. – 2010, №12.