DOI: 10.21276/sjet.2016.4.11.2

Scholars Journal of Engineering and Technology (SJET) **ISSN 2321-435X (Online)**
Sch. J. Eng. Tech., 2016; 4(11):527-533 **ISSN 2347-9523 (Print)**

(An International Publisher for Academic and Scientific Resources)
www.saspublisher.com

Original Research Article

A Load Balancing Algorithm for SDN

Tkachova O, Uzor Chinaobi[1], Abdulghafoor Raed Yahya[2]
[1]Telecommunication systems department, 61166, Kharkiv National University of Radio Electronics, Ukraine
[2]Telecommunication systems department, 65029, Odessa, Odessa National Academy of Telecommunications named after O.S. Popov, Ukraine

***Corresponding author**
Tkachova O
Email: olena.tkachova@nure.ua

Abstract: The concept of Software-Defined Networking are growing quickly. The service providers used the concept of Software-Defined Networking has a lot of benefits, despite this they facing challenges too. The traffic distribution between multiple available paths is one of the urgent problem. Taking advantage of the global view of network topology by controller, a load balance algorithm based on current information about processors utilization on service equipment and channel load is proposed in the paper. The proposed algorithm allows to avoid channel overload due to the reservation of residual value of channel bandwidth.
Keywords: Software-Defined Networking; load balancing algorithm; adjacency matrix, bandthwidth utilization, maximal throughput.

INTRODUCTION

The complexity of management systems and network infrastructure are arising together with amount of function and services provided by multiservice networks. The main goal of Software-Defined Networking (SDN) [1, 2] concept is elimination of existing obstacles faced by most organizations as in improving the efficiency of network functionality. The main idea of SDN concept bases on decoupling the management functionality from data transmission functionality - management functions move from data transmission equipment such as routers and switches to the controllers.

One of the priority directions of multiservice networks development, in particular the SDN-based networks, is to improve the required quality of provided services. Network performance and reliability, packet loss, delay and delay variation are the key quality of service indicators in SDN-based networks like in the traditional network infrastructure [3]. However, for SDN-based networks in addition with such resources as network bandwidth, CPU time and the amount of buffer size the controller CPU and time of forwarding table operation should be take into account. Time for make a decision on OpenFlow switches and distribution data flows through the Flow Visor add new important parameters to the network functionality too [4, 5].

There for the load balancing algorithms used in SDN-based networks have complex mechanisms. The choice of load balanced algorithms depends of many factors. The analysis of main policies and characteristics like response time, throughput that archived through using of different load balancing algorithms give ability to choose the better solutions of load balancing for different type of SDN-based networks. The analysis of existing solutions [5, 6] and specification shows that for today the SDN concept have not strict requirements of load balancing and traffic distribution [7]. The lack of traffic engineering mechanisms such as load balancing algorithms, task scheduling and routing mechanisms raises the problem connected with performance of overlay network solutions.

Analysis of relative works

Load balancing is a methodology of distribution workload across multiple resources through appropriate network paths. This methodology allows achieving optimal utilization of resources, maximizing throughput and minimizing transmission time [3]. The main goal of load balancing algorithms is effectively load distribution over the resources in cloud architecture.

According to the resource allocation procedure load balancing algorithms can be divided into static [6] and dynamic [7].

Static load balancing algorithms are based on the information about the average behavior of system; transfer decisions are independent of the actual current system state. Static load balancing procedures are used in the presence of prior knowledge about the services and applications of statistical information about the network environment. The goal of static load balancing method is to reduce the execution time and minimize the communication delays. Round Robin [8], Randomized [9], Central Manager [10] and Min Min [9] algorithms are the static load balancing algorithms.

In Round Robin algorithm the execution processes are divided between all processors. Each process is assigned to the processor in a round robin order. However, the workload distributions between processors are equal but the processing time for different processes are not same. So at any point of time some nodes may be heavily loaded and others remain idle.

Randomized algorithm is a process that can be handled by a particular node with different probability. The process allocation order is maintained for each processor independent of allocation from remote processor. This algorithm works well in case of processes are of equal loaded. Randomized algorithm does not maintain deterministic approach.

Central Manager algorithm works on the principal of dynamic distribution. Each new request that arrived to the queue manager is inserted into the queue. When request for an activity is received by the queue manager it removes the first activity from the queue and sends it to the requester. If no ready activity is present in the queue the request is buffered, until a new activity is available.

The Min Min algorithm firstly finds the minimum execution time of all tasks. Then it chooses the task with the least execution time among all the tasks. The algorithm proceeds by assigning the task to the resource that produces the minimum completion time. The same procedure is repeated by Min Min until all tasks are scheduled [9].

In dynamic load balancing algorithms work load is distributed among the processors at runtime. The master assigns new processes to the slaves based on the new information collected [9]. Token Routing [7], Central Queuing [10], Least Connection algorithms are the main types of dynamic load balancing algorithms. Token Routing algorithm minimizes the overload in cloud network by use special tokens (agents). Agents gather statistics and distribute traffic according this statistic. The algorithm provides the fast and efficient routing decision.

Central Queuing algorithm works on the principal of dynamic distribution. Each new activity arriving at the queue manager is inserted into the queue. When request for an activity is received by the queue manager it removes the first activity from the queue and sends it to the requester. Is not ready activity is present in the queue the request is buffered, until a new activity will be available.

Least connection is a dynamic scheduling algorithm. It needs to count the number of connections for each server dynamically to estimate the load. The load balancer records the connection number of each server. This algorithm is suitable in case when the amount of nodes changes in some thresholds.

The analysis of the proposed solutions [5, 6] and specification shows that for today the SDN concept have not strict requirements of load balancing and traffic distribution [7]. The lack of traffic engineering mechanisms such as load balancing algorithms, task scheduling and routing mechanisms raises the problem connected with performance of overlay network solutions.

Design a load balancing algorithm for SDN-based networks

SDN architecture solutions are based on principles of network overlay or underlay [4, 5].The basic idea of overlay is leverage the existing physical network infrastructure and apply features and functions such as provisioning that can be used via abstraction. The management functions in overlay SDN-based networks are moved from the control plane of the network to servers. In this case such functions as server virtualization, task scheduling, L4-L7 load balancing and security are used [1]. Therefore, the implementation of SDN as an overlay network allows separate virtual network topology and configuration from the physical network topology. The virtual network creates the path and forward data between virtual components, physical network only delivers packets to destination node. This approach allows rebuilding and modernizing the physical network as needed, without changing anything on the virtual level. The basic architecture of overlay SDN based network is depicted in Figure 1.

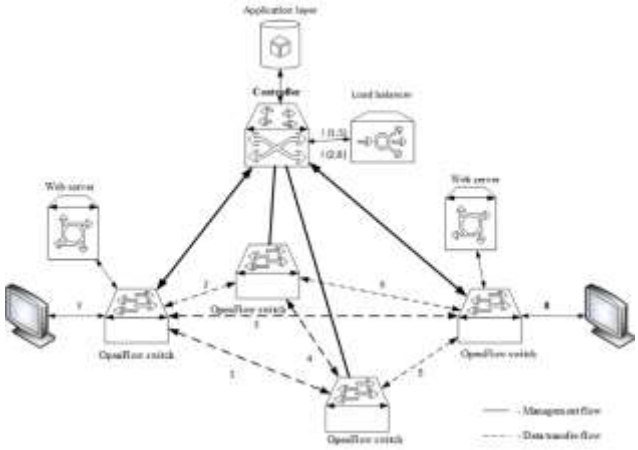

Fig-1: An architecture of overlay SDN-based network

The main task that need to be solved for provision high performance and efficiency of SDN infrastructure is optimal distribution of network and computing resources. The solution of this task should take into account the optimal distribution such network characteristics as channels throughput and compute node performance.

Formally, the task of optimal distributing the channels throughput can be represented as follow:

$$Q(\Lambda^{IN}, \{Th_{ini} ... Th_{inj}\}) \rightarrow \max_{CP_\alpha}, \quad (1)$$

where Q() is optimization function, Λ^{IN} is total input data flow, $\{Th_{ini} ... Th_{inj}\}$ is channels throughput, CP_α is policy of traffic distribution. The end goal of traffic distribution task can be represent as:

$$\Lambda^{IN} \rightarrow \max_{CP_\alpha} | Th_{ini} ... Th_{outj}. \quad (2)$$

The formalization (2) is achieved by the primary choice of channel with a maximum throughput for highest priority data flow and ranging the rest channel according next principle: less traffic priority to the less throughput. The Edmonds-Karp algorithm [11] and maximum throughput scheduling algorithm [12] are laying in the basis of proposed load balancing algorithm.

The proposed algorithm includes the next steps:

1. Controller executes topology discovery and gathering network statistic of overlay network. The LLDP is used for current topology detection and SNMP is used for statistic gathering [10].
2. An "adjacency matrix" construction. The "adjacency matrix" includes all possible relationships between set of computation nodes N_λ that belong to overlay network. The rows and columns of the matrix correspond to computation nodes (VM), $1 \le N_{\lambda i} \le M$.
3. The construction of "adjacency matrix" are corresponding to the rule: if the link between nodes exist the interception between them match as "1", if the link is absent the interception symbol is "0". Only those channels of communication are considered in the future work of the algorithm, the value of which in the matrix is equal "1", $Ch_{ij} = 1$
4. Weight Determination and scheduling of traffic distributing. Two additional matrices are built on basic of primary adjacency matrix M.

First matrix M_1 interprets the current throughput of network channels: if the link between nodes N_i and N_j exists the meaning $Ch_{ij} = 1$ changes to the current value of current throughput of channels.

The interception of rows and columns of matrix M_2 interprets the maximum bandwidth of channel for overlay network.

$$M_1 = \begin{bmatrix} 0, c_{12}, c_{13}, ...c_{1M} \\ c_{21}, 0, c_{13}, ...c_{2M} \\ ..... \\ c_{M1}, c_{M2}, c_{M3}, ...0 \end{bmatrix}, \; M_2 = \begin{bmatrix} 0, C_{12}, C_{13}, ...C_{1M} \\ C_{21}, 0, C_{13}, ...C_{2M} \\ ..... \\ C_{M1}, C_{M2}, C_{M3}, ...0 \end{bmatrix},$$

where M is total amount of nodes that belong to overlay network.

4.1 The weight of peer-to-peer channels and bypass channels between nodes N_i and N_j are determining.

4.2 Separation the incoming data flow according to the priority. In proposed algorithm proposed to separate incoming data flow to priority (real time services such as voice, video, games) and non-priority. Two new matrices that indicate the intensity of incoming flow. Matrix $M^p_{\lambda inp}$ indicates the intensity of incoming data flow with high priority:

$$M^p_{\lambda inp} = \begin{bmatrix} 0, \lambda^p_{inp12}, \lambda^p_{inp13}, ...\lambda^p_{inp1M} \\ \lambda^p_{inp21}, 0, \lambda^p_{inp23}, ...\lambda^p_{inp2M} \\ ..... \\ \lambda^p_{inpM1}, \lambda^p_{inpM2}, \lambda^p_{inpM3}, ...0 \end{bmatrix}.$$

Matrix $M^p_{\lambda inp}$ indicates the intensity of non-priority incoming data flow:

$$M^{np}{}_{\lambda inp} = \begin{bmatrix} 0, \lambda^{np}{}_{inp12}, \lambda^{np}{}_{inp13}, ... \lambda^{np}{}_{inp1M} \\ \lambda^{np}{}_{inp21}, 0, \lambda^{np}{}_{inp23}, ... \lambda^{np}{}_{inp2M} \\ ..... \\ \lambda^{np}{}_{inpM1}, \lambda^{np}{}_{inpM2}, \lambda^{np}{}_{inpM3}, ... 0 \end{bmatrix}.$$

The total flow of incoming traffic is defined as $\lambda_{inpij} = \lambda^{p}{}_{inpij} + \lambda^{np}{}_{inpij}$, where $\lambda^{p}{}_{inpij}$ is high priority incoming data, $\lambda^{np}{}_{inpij}$ is non-priority incoming data.

1. Traffic distribution between nodes N_i and N_j according to the weight of link in overlay network.

Firstly, the channel with maximal available throughput between nodes N_i and N_j is defined. Both direct channel and bypass channel can be chosen. Then the ratio of incoming data flow intensity to throughput is evaluated. The conclusion about possibility to use the channels for priority data flow bases on next formalisms:

$$\begin{cases} Ch_{ic_{ij}} / \lambda_{inpij} < 1, Ch_{c_{ij}} \text{ is overloaded} \\ Ch_{ic_{ij}} / \lambda_{inpij} > 1, Ch_{c_{ij}} \text{ is oloaded} \\ Ch_{ic_{ij}} / \lambda_{inpij} > Th_{c_{ij}}, \text{underloaded} \end{cases}, \quad (3)$$

where $Th_{c_{ij}}$ is current threshold value for possible channel load.

2. The calculation of the residual channel resource.

The calculation of residual bandwidth of chosen channel bases on the collected statistics according to the next rules:

$$Th_{c_{ij}} = \begin{cases} c_{ij} - \sum_{i,j=1}^{n} \lambda^{p}_{ij}(t_s) \mid \lambda^{p}_{ij}(t_s) \le c_{ij} \\ 0, \lambda^{p}_{ij}(t_s) = c_{ij} \end{cases}, \quad (4)$$

where $Th_{c_{ij}}$ is threshold value for the chosen channel.

Let's assume that incoming traffic flows, which can be divided into two classes of service, arrives in the time t_s

The priority flow $\lambda^{p}{}_{inp}(t_s) = \{\lambda^{p}{}_{inp1}(t_s), ..., \lambda^{p}{}_{inpj}(t_s)\}$ and non-priority flow $\lambda^{np}{}_{inp}(t_s) = \{\lambda^{np}{}_{inp1}(t_s), ..., \lambda^{np}{}_{inpj}(t_s)\}$. Controller determines the unloaded channel and checks the possibility high priority data through this channel:

$$Ch_{ic_{ij}}(t_s) / \lambda^{p}{}_{inpij}(t_s) \ge 1. \quad (5)$$

- if the statement (2) is true the incoming data flow is transmitted through the channel $Ch_{c_{ij}}$ during time interval Δt. The residual bandwidth of this channel is determined as follows:

$$Ch_{ic_{ij}} - \lambda^{p}{}_{inpij}(t_s) = \Delta Ch_{ic_{ij}}(t_s). \quad (6)$$

The value of non-priority data flow that can transmitted through channel $Ch_{ic_{ij}}$ during time t_s, $\Delta Ch_{ic_{ij}}(t_s)$, is calculated.

- if $\Delta Ch_{ri_{ij}}(t_s) \ge \lambda^{np}{}_{inpij}(t_s)$ the non-priority data flow can be transmitted through same channel. The possible value of non-priority data flow calculated as follows:

$$AL = \frac{\lambda^{np}{}_{inpij}(t_s)}{k}, \quad (7)$$

where k is load factor, it is using for limitation of total channel load in overlay network and give possibility to use the channel as reserved.

- if the statement (2) is false, another channel should be selected. The stages 2-4 doing recursively to find unloaded channel between nodes N_i and N_j.

The maximal value of load in the chosen channel are calculated as follows:

$$C_{ijmax}(t_s) = \sum_{i=1}^{n} \lambda^{p}{}_{inpij}(t_s) + \frac{\lambda^{np}{}_{inpij}(t_s)}{k}. \quad (8)$$

Either the checking of channel throughput performed again either at intervals Δt or in case when new incoming data flow $\lambda^{p}{}_{inpij}(t_{s+1})$ are arrived to demarcation devises.

The flow chart of proposed algorithm is depicted in Figure 2.

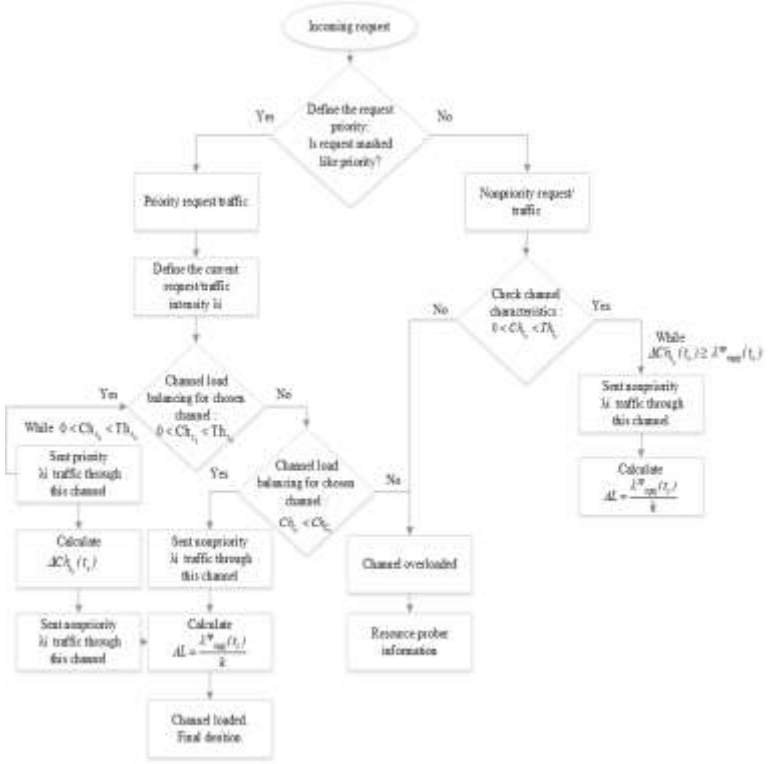

Fig-2: Flow chart of proposed network load balancing algorithm

EXPERIMENTAL RESULTS

For evaluation the efficiency of proposed algorithm of load balancing for SDN-based networks the network simulation tool called mininet [13] was used. The software POX [14] SDN controller and six OpenFlow Switches was simulate in the network fragment as shown in Figure 3. The OpenFlow Switches have the next characteristics: Sw1 capacity – 45Mbps, Sw2 capacity – 45Mbps, Sw6 capacity – 45Mbps, Sw2 capacity – 60Mbps, Sw4 capacity – 60Mbps, Sw5 capacity – 60Mbps.

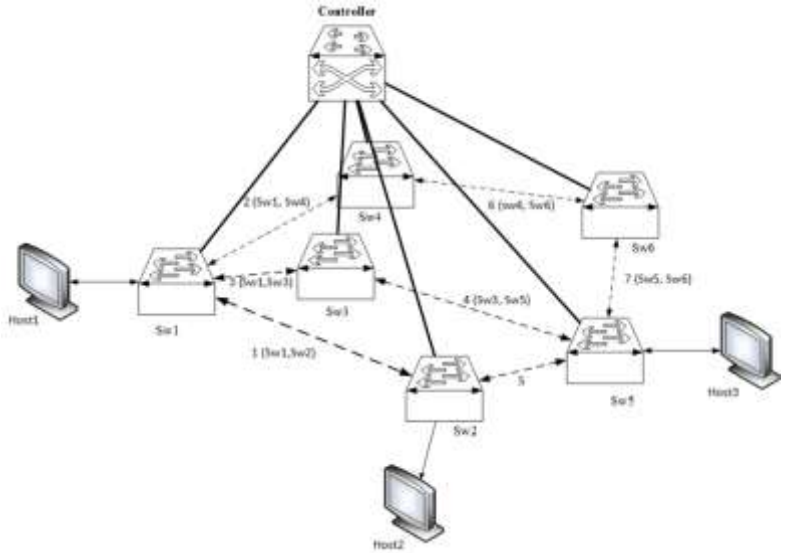

Fig-3: Fragment of experimental SDN-network

There the multiple path overlay SDN-based network. The follow paths from Host1 to Host3 are considered:

Path 1: Sw1 – Sw2 – Sw5;
Path 2: Sw1 – Sw3 – Sw5;

Path 3: Sw1 – Sw4 – Sw6 – Sw5.

The two existing load balancing algorithms were consider in order to evaluate the proposed load balancing algorithm: Round Robin algorithm [7] and Least connection algorithm [10]. During the experiment, the average throughput utilization and the transmission delay will be recorded. The experimental results show in Table 1 and Table 2.

Table 1: Comparison of average throughput utilization (Host 1 to Host 3), %

	Round Rabin	Central Queuing	Proposed algorithm
Path 1	0.305	0.390	0.299
Path 2	0.411	0.745	0.385
Path 3	0.899	0.842	0.870

Table 2: Comparison of average delay (Host 1 to Host 3), ms

	Round Rabin	Central Queuing	Proposed algorithm
Path 1	0.300	0.390	0.320
Path 2	0.317	0.445	0.335
Path 3	0.420	0.542	0.470

According to the results, proposed load balancing algorithm shows the better average throughput utilization compared with the static Round Robin and dynamic Central Queuing algorithms. The obtained results is connected with the advantages of main idea of proposed algorithm - focus on global path condition.

CONCLUSION

Overlay solutions of SDN-based networks became very popular. In overlay SDN-based network controller as a centralized management device gathers information about network topology and current network state. The load balancing algorithm proposed in the article are based on the main feature of SDN: centralization management and control and overlay. In the paper analyzed the static and dynamic algorithms of load balancing. Most existing strategies employ simple methods to decide best global transmission path with poor performance and shown that not whole factors are taken into account.

The proposed algorithm give ability to track the current state of the network and respond to changes in real-time. In this way application of the proposed algorithm allows to avoid channel overload, because the residual value of channel bandwidth are lived as reserved. The load balance strategy proposed in this paper is applied in the experiment using mininet. The static Round Robin, dynamic Central Queuing algorithms are take into account. The comparison of obtained experimental results show that proposed load balancing algorithm allows to provide better average throughput utilization and achieve the decreasing of network delay in comparison with Central Queuing algorithms.

REFERENCES

1. Raghavan B, Casado Koponen MT, Ratnasamy S, Ghodsi A, Shenker S. Software-defined internet architecture: Decoupling architecture from infrastructure, in Proceedings of the 11th ACM Workshop on Hot Topics in Networks, ser. Hot Nets-XI. New York, NY, USA: ACM. 2012;43–48.
2. Kim H, Feamster N. Improving network management with software defined networking, Communications Magazine, IEEE. 2013;51(2):114–119.
3. Koerner M, Kao O. Multiple service load balancing with Open Flow, IEEE HPSR. 2012.
4. Vishnio R, Poddar V. Effective Switch Memory Management in OpenFlow Networks. Proceedings of the ACM International Conference on Distributed Event-Based Systems. 177-188.
5. Wang SC. Towards a load balancing in a three level cloud computing network. Proceedings of 3rd International Conference on Computer Science and Information Technology (ICCSIT), IEEE. 2010;108-113.
6. Ren X, Lin R, Zou H. A dynamic load balancing strategy for cloud computing platform based on exponential smoothing forecast. International Conference on. Cloud Computing and Intelligent Systems (CCIS), IEEE. 2011;220-224.
7. Hiranwal S, Roy KC. Adaptive Round Robin Scheduling using shortest burst approach based on smart time slice, International Journal of Computer Science and Communication. 2010;2(2):319-323.
8. Mitzenmacher M, Upfal E. Probability and Computing: Randomized Algorithms and Probabilistic Analysis. Cambridge University Press. 2005;110.
9. Yagoubi B, Slimani Y. Task Load Balancing Strategy for Grid Computing. Journal of Computer Science. 2007;3(3):186-194.

10. Ren X, Lin R, Zou H. A dynamic load balancing strategy for cloud computing platform based on exponential smoothing forecast. proc. International Conference on. Cloud Computing and Intelligent Systems (CCIS), IEEE. 2011; 220-224.
11. Sobrinho JL. Network routing with path vector protocols: Theory and applications. Proc. ACM SIGCOMM. 2003;40-60.
12. Bryant RM, Finkel RA. A Stable Distributed Scheduling Algorithm in Proc. 2nd Int. Conf. Dist. Comp., 341-323.
13. Mininet tutorials http://mininet.org/.
14. POX controller simulator https://github.com/noxrepo/pox.