A network edge monitoring approach for real-time data streaming applications

Salman Taherizadeh[1,4], Ian Taylor[2], Andrew Jones[2], Zhiming Zhao[3], and Vlado Stankovski[4]

[1] University of Ljubljana, Faculty of Computer and Information Science
{Salman.Taherizadeh}@fgg.uni-lj.si
[2] Cardiff University, School of Computer Science and Informatics
{TaylorIJ1, JonesAC}@cardiff.ac.uk
[3] University of Amsterdam, Informatics Institute
{Z.Zhao}@uva.nl
[4] University of Ljubljana, Faculty of Civil and Geodetic Engineering
{Vlado.Stankovski}@fgg.uni-lj.si

Abstract. Renting very high bandwidth or special connection links is neither affordable nor economical for service providers. As a consequence, ensuring data streaming systems to be able to guarantee desired service quality experienced by the users has been a challenging issue due to real-time changes in the network performance of the Internet communications. This paper presents a network monitoring approach that is broadly applicable in the adaptation of real-time services running on network edge computing platforms. The approach identifies runtime variations in the network quality of links between application servers and end-users. It is shown that by identifying critical conditions, it is possible to continuously adapt the deployed service for optimal performance. Adaptation possibilities include reconfiguration by dynamically changing paths between clients and servers, vertical scaling such as re-allocation of bandwidth to specific links, horizontal scaling of application servers, and even live-migration of application components from one edge server to another to improve the application performance.

Keywords: Edge computing, network monitoring, real-time data streaming

1 Introduction

Real-time applications such as online gaming, telemedicine services, environment monitoring systems, and video conferencing have highly on-demand needs to provide not only a high-quality result but also deliver the result as early as possible for the best real-time user experience—such as shorter response time via closer interaction with the application server or higher resolution via more stable connection. However, using cloud infrastructure to deploy such real-time applications provides some benefits, such as reduction of operation costs, on-demand resource allocation should be flexible enough in order to dynamically assign infrastructure according to needs of such application and hence save the expense.

Since real-time applications may become sensitive to the network quality, such as latency between clients and running services, the requirements of such applications could potentially be addressed by emerging edge computing technologies, which allow computations to be performed at the edge of the network. The rationale of employing these technologies is that computing should happen at the proximity of data sources—e.g. cameras or sensors—and closer to where the results are needed [1].

Due to the federated nature of edge computing scenarios, real-time applications can be deployed on different edge nodes with diverse properties (for instance network performance, physical location, reliability, connectivity and so on). Hence, the performance of such real-time applications varies significantly depending on the runtime properties of their infrastructural resources as well as their clients' network conditions. To come up with these challenges, implementing effective, transparent and elastic methods to monitor the Quality of Service (QoS) at the network edge is difficult, yet also necessary. It is necessary because obtaining such network QoS parameters makes it possible to take appropriate adaptation decisions at strategic level (e.g. about the application topology and the selection of one or more physical machines on which it will be running at geographic locations) and dynamic level (e.g. about the application reconfiguration, vertical or horizontal scaling, re-location and so on). The edge servers should continue to monitor these parameters and determine if user experience needs to be improved. In this way, more dynamic adaptations to the user's conditions (e.g. network status) can be accomplished by utilising network edge-specific knowledge [2]. Therefore, particular attention has to be paid to monitoring network links between end-users' clients and edge servers.

The goal of the present paper is to implement a network edge monitoring approach that considers critical QoS metrics including delay, packet loss, throughput and jitter which are specific to the real-time applications we envisage. Therefore, it can be used for functionalities such as runtime service adaptations for streaming cases, for example automatically tuning the network quality by changing network paths to re-route via other edge servers or dynamically connecting the clients to the best servers based on their network edge conditions, location, etc. In this way, our proposed system is able to ensure the best possible Quality of Experience (QoE) for the users.

The rest of the paper is organized as follows. Section 2 presents the related work. Section 3 describes a real-time data streaming use case. Section 4 presents our network edge monitoring approach, followed by preliminary results in Section 5, and the discussion and conclusions appear in Section 6.

2 Related work

There have been many research approaches, trying to provide network QoS guarantees for real-time data streaming services. A new paradigm, called edge computing, is emerging as an extension of cloud computing to support and meet the QoS requirements of real-time applications which are delay and jitter-sensitive [3]. Since edge computing is deployed at the edge of the network, it provides low latency,

location awareness, and optimizes users' experience under QoS requirements for streaming and real-time applications [4].

Chen *et al.* [5] focused on the users' perspective in online gaming systems; from their point of view, the QoS metrics related to network conditions—namely delay, packet loss, bandwidth—have an important effect on gaming experience. Their results showed that packet loss and bandwidth limitations impose negative impact on the frame rates and the graphic quality in these systems. To provide more stable network performance for real-time services and optimizing the network path and resources, Jutila [6] presents adaptive edge computing solutions based on different traffic management methods that monitor and react to network QoS changes. To check the network quality in the context of such applications, the most important metrics to be analysed for adaptation are network throughput, latency, packet loss and jitter [7].

According to the cited literature, we can conclude that monitoring of these network-related metrics can help data streaming service providers guarantee QoE to end-users facing network resources limitations. Furthermore, an investigation of the recent related work supports the conclusion that a current challenge in this area is to continuously adjust the deployed environment according to the runtime changes in network conditions intrinsic to connections of both application servers and also users; this is the main focus of the research presented in the paper.

For dynamic adaptation of edge-based applications, emphasis should be put on the importance of scalability, robustness, non-intrusiveness, interoperability and possibilities to support live-migration of the service.

- Scalability: A scalable monitoring system is able to handle huge amounts of monitoring data across large numbers of resources and services [8].
- Robustness: A robust monitoring system is able to be highly tolerant of many failure scenarios and detect changes in environment, adapting to a new situation and continuing its operation [9].
- Non-intrusiveness: A non-intrusive monitoring system is capable of being lightweight to the normal flows of application and infrastructure [10].
- Interoperability: An interoperable monitoring system is not specific to a given infrastructure and is able to monitor an application that resides on other cloud providers' infrastructure [11].
- Live-migration support: In live-migration, applications migrate from a physical host to another one at any time without stopping operations [12].

Table 1 presents the analysis of the essential properties for the widely used multi-cloud monitoring tools. The goal of the comparison is to specify and trade-off the strengths, drawbacks and challenges which have been encountered in the context of self-adaptive edge-based applications.

Table 1. Requirement analysis for multi-cloud monitoring systems

Tool	Scalability	Robustness	Non-intrusiveness	Interoperability	Live-migration support
Zenoss[1]	Yes	No	Yes	Yes	No
Ganglia[2]	Yes	Yes	limited	Yes	Yes
Zabbix[3]	Yes	No	Yes	Yes	No
Nagios[4]	No	No	limited	Yes	No
OpenNebula[5]	Yes	Yes	Yes	No	limited
Lattice[6]	Yes	Yes	Yes	Yes	Yes
JCatascopia[7]	Yes	Yes	limited	Yes	Yes

Comparison in Table 1 is upon the reviewed literature and based on conducting experiments with the tools. These tools are investigated in order to find out an appropriate base-line technology for the needs of monitoring edge-based applications and the requirements of automatic adaptation to guarantee the QoS and the QoE performances which are subjective measure from the users' viewpoint on the overall value of the provided service.

3 Real-time data streaming use case: WebRTC/MCU

Real-time communication plays an increasingly important role for many business applications, including cooperative working environments and video-conferencing for instance via WebRTC[8] (Web Real-Time-Communications) technology [13]. The WebRTC use case is explained here as an example of a large range of new potential real-time applications which need to have very high QoS in regard to their communication service, detect and respond to network-based urgent events very rapidly and also operate reliably and robustly throughout their lifetime.

The WebRTC open project enables real-time communications directly in the browser, and its performance may be influenced by highly fluctuating quality of the Internet connections. To this end, intermediate devices called Multipoint Control Unit (MCU) servers, which can be running on different data centers all around the world, are being used to manage the communication between the clients. The function of these MCU servers deployed at the edge of the network is to coordinate the distribution of audio, video, and data streams amongst the multiple participants in a multimedia session. These data centers allow interconnecting MCU servers in different regions. Therefore, for every user, there is an opportunity to have more than one MCU server to provide the service and hence it would be possible to connect a user to the best possible MCU. Figure 1 shows an example of how to interconnect all MCU servers to each other.

[1] Zenoss monitoring system, http://www.zenoss.org
[2] Ganglia monitoring system, http://ganglia.info/
[3] Zabbix monitoring system, http://www.zabbix.com/
[4] Nagios monitoring system, https://www.nagios.org/
[5] OpenNebula, http://www.opennebula.org/
[6] Lattice, http://reservoir-fp7.eu/
[7] JCatascopia monitoring system, http://linc.ucy.ac.cy/CELAR/jcatascopia/
[8] WebRTC, https://webrtc.org/

Fig. 1. A deployment of MCU servers' interconnection globally running all over the world

There are plenty of other applications, similar to MCU servers in a WebRTC video-conference, in which communication between users is required to pass through intermediate servers. Examples include the Openfire[9] server in instant messaging (IM) group chat, and CipSoft[10] servers in online gaming.

4 Design and implementation of the monitoring approach

In our work, we focus on performance indicators from the user perspective; since they can be used to evaluate the network quality delivered to an end-user, then it is possible to improve the overall acceptability of the service, as perceived subjectively by each user. Figure 2 provides the schema of a user's communication via an MCU server as intermediary, which has to be monitored and compared with the other alternatives as potential MCU servers deployed in highly distributed, edge computing infrastructures. Supported by edge computing platforms such as Docker, the intermediary service can be deployed on-the-fly or on several running instances in different edge computing nodes.

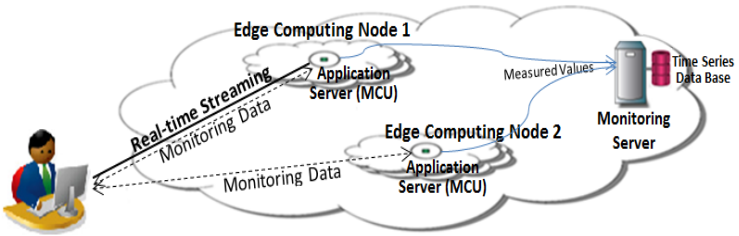


Fig. 2. Use and monitoring of MCUs to support real-time streaming

[9] Openfire, http://www.igniterealtime.org/projects/openfire/
[10] CipSoft, http://www.cipsoft.com/

An overview of the proposed monitoring architecture is shown in Figure 3.

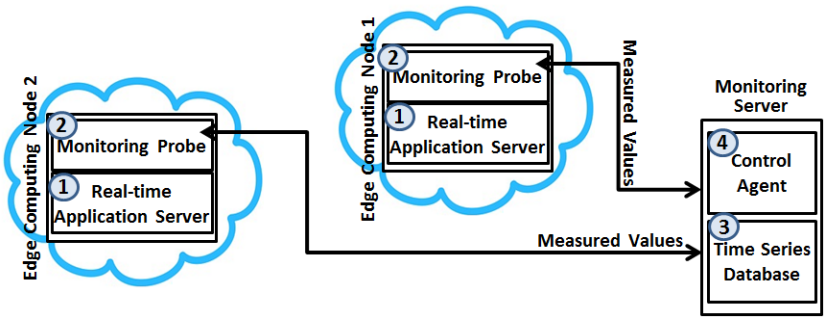


Fig. 3. Overview of the proposed model for user-centric network monitoring

This monitoring system employs a number of distinct components. The light-weight, scalable, custom-made monitoring system implemented in JCatascopia framework [14] is responsible for monitoring QoS parameters of connections between the real-time application edge server and clients at the network layer. The implemented monitoring system is not limited to operating on specific cloud providers and can be utilized to monitor federated cloud environments where applications are residing on multiple infrastructures. As shown in Figure 4, the network-level monitoring probe could separately represent a standalone application that runs amongst other running applications.

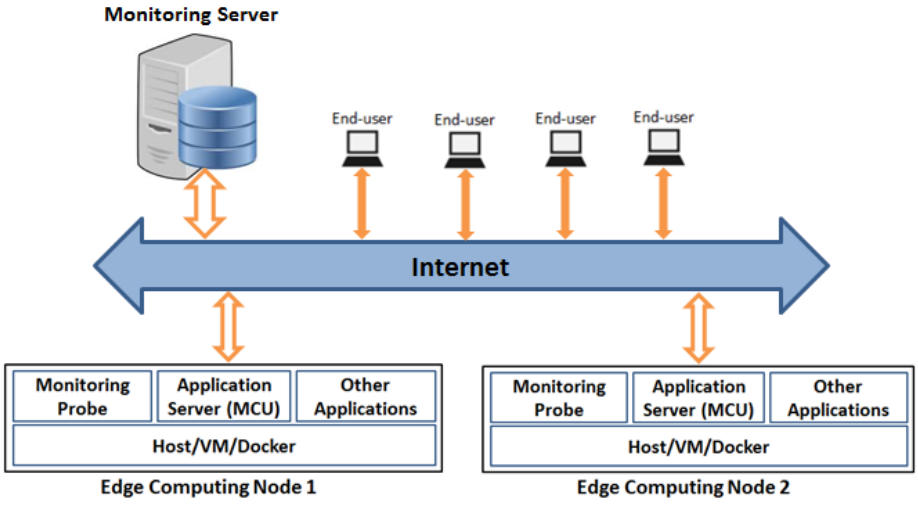


Fig. 4. Network-level monitoring probe running amongst other applications

Different network-level QoS metrics—including throughput, delay, jitter and packet loss, which have been considered as important parameters for various video/audio streaming applications—are measured by monitoring probes on top of each edge node. The pseudocode of the developed algorithm for the monitoring probe is depicted in Figure 5.

```
/* Monitoring probe running on a Host/VM/Docker which resides in Edge Computing Node i (ECN_i)*/
/* PL: Packet Loss */
/* NT: Network Throughput */
/* AD: Average Delay */
/* AJ: Average Jitter */
while(true){
  TS ← TimeStamp()
  for each current User_x do {
    PL ← Calculate_PL(ECN_i, User_x) //Calculate PL of link between ECN_i and User_x
    NT ← Calculate_NT(ECN_i, User_x) //Calculate NT of link between ECN_i and User_x
    AD ← Calculate_AD(ECN_i, User_x) //Calculate AD of link between ECN_i and User_x
    AJ ← Calculate_AJ(ECN_i, User_x) //Calculate AJ of link between ECN_i and User_x
    Message ← Make_Message(ECN_i, User_x, TS, PL, NT, AD, AJ)
    Send_To_Monitoring_Server(Message)
  } // end of for
  wait(interval)
} // end of while
```

Fig. 5. Pseudocode for the implemented monitoring probe

A monitoring server is responsible for orchestrating and collecting QoS data from each monitoring probe. The monitoring server consists of two parts: a Time Series Database (TSDB) and a control agent. The TSDB, implemented by Apache Cassandra server, is used to store the measured values, while the control agent, implemented in Java, is responsible for network-based QoS analysis, evaluating relevant policies and returning decisions consistent with these policies. It analyses the running servers' status and provides adaptation plans, for instance, changing network paths to re-route via other edge servers, or vertical scaling by resizing the resources e.g. to offer more bandwidth, or application server check-pointing/live-migration, and so forth. The pseudocode of the developed algorithm for the control agent is depicted in Figure 6 where the coefficients C_1, C_2, C_3 and C_4 are the weights assigned to each network-level QoS metric. These weights could be dependent on the use case. For example, for VoIP applications, jitter is more important than delay. Consequently, jitter should have bigger weight in this case as it has more influence on user experience.

```
1: /* Control agent running as a part of monitoring server*/
2: /* NQ_{i,x}: Network Quality of link between Edge Computing Node i (ECN_i) and User_x */
3: /* BECN_x: The Best Edge Computing Node to provide the service for User_x: */
4: while(true){
5:    for each current User_x do {
6:       for each ECN_i do { //Calculate Network Quality for each possible Edge Computing Node
7:          NQ_{i,x} ← C1*PL + C2*NT + C3*AD + C4*AJ
8:       } // end of for
9:       BECN_x ← Choose_The_Best_Node_With_Maximum_Network_Quality(NQ_{i,x})
10:      if BECN_x has better network quality compared to the current Edge Computing Node
11:      during the last α intervals do{  // α can be defined according to the use case
12:          Adaptation plan is launched
13:      } // end of if
14:   } // end of for
15:   wait(interval)
16: } // end of while
```

Fig. 6. Pseudocode for the implemented control agent

Considering Figure 3, when our monitoring solution executes, it proceeds as follows: (1) A real-time application server (e.g. MCU server implemented by Medooze[11] as an open source conference application in our experiment) typically serves a large number of users and can be deployed and run on a selected edge computing node. (2) All relevant QoS metrics are measured at regular intervals by a monitoring probe and then all measured values will be reported to the monitoring server. (3) The monitoring server stores the collected metrics in a TSDB. The collected data can be analysed and used for capacity planning and strategic analysis like longer-term usage trends. (4) The control agent checks possible degradation of the required network quality for each edge computing node, and relates any such information to the current demand. When the current condition does not satisfy the expected requirements, an adaptation plan to achieve the desired performance can be launched. Using Docker's container-based virtualization, the control agent includes Kubernetes[12] technology for dynamically automating deployment, scaling, and management of containerized applications.

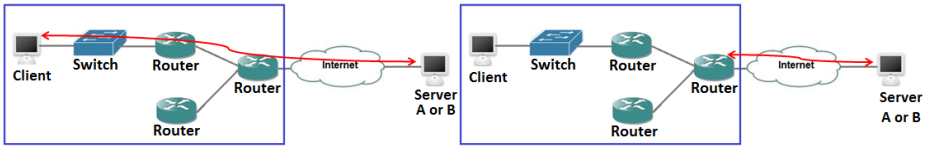
5 Measurements

In order to demonstrate the presented monitoring approach, network QoS is evaluated through four metrics, namely delay, packet loss, throughput and jitter. Deployed monitoring probes can measure these network QoS metrics which particularly affect the application performance. In case of any deterioration of system health, for example due to the presence of excessive jitter, control agent may trigger an adaptation mechanism to fix the QoS-related problem. The possible adaptation mechanism could be, for instance, re-connecting users to a set of the best reliable servers offering fully-qualified network performance.

[11] Medooze, http://www.medooze.com/
[12] Kubernetes, http://kubernetes.io/

For experimentation, we used a WebRTC client with a low-throughput connection and two MCU servers (A and B) running on infrastructures in different geographical locations with one Gbps bandwidth and the same processing power and memory—2397.222 MHz and 2 GB RAM respectively. The monitoring probes use *ping* results to periodically measure the QoS metrics via the ICMP (Internet Control Message Protocol) protocol. The monitoring probes deployed on the two MCU servers send 10 ICMP packets at an interval of 200 ms to the WebRTC client and then they calculate the QoS metrics on an average basis. Each ICMP packet includes 500 bytes of data. The set of measurements are repeated continuously at a frequency of 15 times per minute which means every 4 seconds.

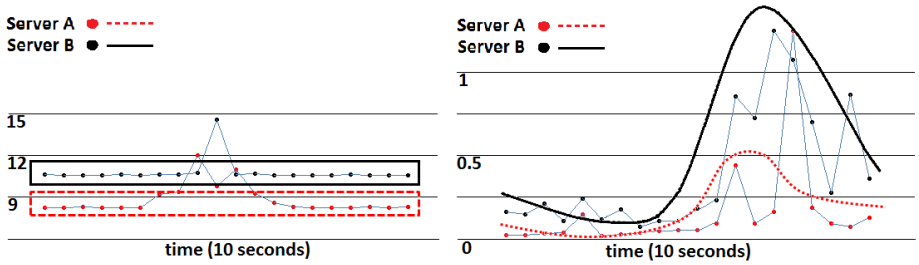
The implemented network edge monitoring technique takes into account two types of conditions, shown in Figure 7: when ICMP packet filtering is (a) disabled or (b) enabled, inside a private network which is depicted by a rectangle. It is not unusual that ICMP traffic is filtered in private administrative domains due to various security concerns. In such case, the ping command returns no response and the packet loss is 100%; hence, the monitoring probe changes its mode of operation and uses traceroute to identify the path to the edge router. With this information available, the monitoring probe measures the QoS metrics between the edge server and the edge router (instead of the client). This measurement is an appropriate approximation of network-based QoS between the server and the client; since our main target is to compare the QoS of different routes from the MCU servers to the edge router.



(a) Without ICMP Packet Filtering (b) With ICMP Packet Filtering through edge router

Fig. 7. Path between an edge server and a client with different administrative domains

The measurements presented in Figure 8 show how this tradeoff helps the system monitor the network-level QoS metrics related to two different connections with the same destination, the first connection between edge server A and a client, the second one between edge server B and the same client.



(a) Delay (ms) (b) Jitter (ms)

Fig. 8. Experimental results related to two different connections with the same destination

Figure 8 (a) shows that according to the delay, the performance of edge server A is better than that of edge server B for a certain period of time. Since jitter is calculated as the magnitude of the delay variation, it will always be a positive number with zero indicating that no jitter is present. The standard deviation of delay was computed to measure jitter. Server A, as depicted in Figure 8 (b), provides better QoS in terms of jitter on the average than the server B. Besides, throughput belonging to the both MCU servers was almost steady in the conducted experiment, whereas in real-time systems, continuous fluctuation is significant to be considered as an issue. Also, the system showed that packet loss ratio is zero, which indicates efficient packet transmission in both connections related to server A and server B.

A running monitoring probe does not provoke a meaningful network overhead for one user as we investigated the output of *nethogs* tool to compute its bandwidth overhead. We discovered that the implemented monitoring probe transmits ~1714 and receives ~1397 bytes per second for every user in average. Furthermore, in our experiment, it consumes only 0.7 percent of the whole CPU time and 2.28 percent of the whole memory usage in average.

6 Discussion and conclusions

Cost issues and other associated network-related parameters such as bandwidth capacity and data transfer are outstanding issues for multi-cloud service providers [15]. Therefore, effective usage of network resources plays a significant role in the distributed interoperable environment to save the expense. To this end, our study showed how edge services for time-critical applications could be used to automatically optimize the process of allocating and choosing the best infrastructure, which is responsible for offering acceptable network QoS and QoE.

This research paper presented a network monitoring approach that is particularly suitable in the adaptation of real-time data streaming applications running on edge computing platforms. Adaptation approaches could be, for instance, re-configuration of connections among running application servers and users, or vertical scaling by

resizing the network resources e.g. to offer more bandwidth, or even live-service migration by moving running application servers from the current infrastructure to another one. One of the goals of this paper was also to investigate those network-level metrics that are particularly important for the development and adaptation of time-critical applications. Because these network-based measures can vary greatly and have significant effects on the system's performance and users' satisfaction. One important aspect is the design of adaptation mechanisms that can help the real-time services to react to environmental conditions and events, such as sudden increase in workload or in the number of users.

Due to the distributed nature of edge computing, companies can run their real-time applications in different edge nodes, connecting each one with the users that are using the service in each region. We applied this method in a novel manner within an edge computing platform, such that this extensible architecture can be combined with monitoring information that is only available at the edge of the network. As the major finding, more dynamic adaptation to the user's conditions (e.g. network status and user's geographical location) can also be accomplished with network edge specific knowledge. Our future work includes applying more optimisation algorithms (e.g. Pareto-based multi-objective optimisation approach [16]) and comparing the feasibility of these approaches in the edge computing environment.

7 Acknowledgements

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No. 643963 (SWITCH project: Software Workbench for Interactive, Time Critical and Highly self-adaptive cloud applications).

References

1. Shi, W., Cao, J., Zhang, Q., Li, Y., Xu, L.: Edge Computing - Vision and Challenges. *Technical Report MIST-TR*, Wayne State University, 2016.
2. Zhu, J., Chan, D., Prabhu, M., Natarajan, P., Hu, H., Bonomi, F.: Improving web sites performance using edge servers in fog computing architecture. *IEEE 7th International Symposium on Service Oriented System Engineering (SOSE)*, Pp. 320-323, 2013.
3. Shojafar, M., Cordeschi, N., Baccarelli, E.: Energy-efficient Adaptive Resource Management for Real-time Vehicular Cloud Services. IEEE Transactions on Cloud Computing, Vol. PP, Iss. 99, 2016.
4. Stojmenovic, I., Wen, S.: The fog computing Paradigm - Scenarios And Security Issues. *Conference on Computer Science and Information Systems (FedCSIS*), 2014.
5. Chen, K.T., Chang, Y.C., Hsu, H.J., Chen, D.Y., Huang, C.Y., Hsu, C.H.: On the quality of service of cloud gaming systems. *IEEE Transactions on Multimedia*, Vol. 16, No. 2, Pp. 480-495, 2014.

6. Jutila, M.: An Adaptive Edge Router Enabling Internet of Things. *IEEE Internet of Things Journal*, Vol. PP, Iss. 99, 2016.
7. Cervino, A.J.: Contribution to multiuser videoconferencing systems based on cloud computing. *Doctoral thesis*, Technical University of Madrid, 2012.
8. Clayman, S., Galis, A., Mamatas, L.: Monitoring virtual networks with lattice. *Proceedings of 2010 IEEE/IFIP Network Operations and Management Symposium Workshops (NOMS Wksps)*, IEEE, Osaka, Pp. 239–246, 2010.
9. Fatema, K., Emeakaroha, V.C., Healy, P.D., Morrison, J.P., Lynn, T.: A survey of cloud monitoring tools: Taxonomy, capabilities and objectives. *Journal of Parallel and Distributed Computing*, Vol. 74, No. 10, Pp. 2918-2933, 2014.
10. Taherizadeh, S., Jones, A.C., Taylor, I., Zhao, Z., Martin, P., Stankovski, V.: Runtime network-level monitoring framework in the adaptation of distributed time-critical cloud applications. *Proceedings of the 22nd International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'16)*, ACM, Las Vegas, 6 pages.
11. Alhamazani, K., Ranjan, R., Mitra, K., Rabhi, F., Jayaraman, P.P., Ullah-Khan, S., Guabtni, A., Bhatnagar, V.: An overview of the commercial cloud monitoring tools: research dimensions, design issues, and state-of-the-art. *Computing*, Vol. 97, No. 4, Pp. 357-377, 2015.
12. Nadjaran-Toosi, A., Calheiros, R. N., Buyya, R.: Interconnected cloud computing environments: Challenges, taxonomy, and survey. *ACM Computing Surveys (CSUR)*, Vol. 47, No. 1, 2014.
13. Perkins, C., Westerlund, M., Ott, J.: Web Real-Time Communication (WebRTC) Media Transport and Use of RTP. *IETF Active Internet Draft*, 2012.
14. Trihinas, D., Pallis, G., Dikaiakos, M.D.: JCatascopia - Monitoring Elastically Adaptive Applications in the Cloud. *14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, 2014.
15. Sookhak, M., Gani, A., Talebian, H., Akhunzada, A., Khan, S.U., Buyya, R., Zomaya, A.Y.: Remote data auditing in cloud computing environments: a survey, taxonomy, and open issues. *ACM Computing Surveys (CSUR)*, Vol. 47, No. 4, 2015.
16. Al-Jubouri, B., Gabrys, B.: Multicriteria approaches for predictive model generation: A comparative experimental study. *2014 IEEE Symposium on Computational Intelligence in Multi-Criteria Decision-Making (MCDM)*, IEEE, Pp. 64-71, IEEE, 2014.