Runtime network-level monitoring framework in the adaptation of distributed time-critical Cloud applications

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

[1]Faculty of Computer and Information Science, University of Ljubljana, Ljubljana, Slovenia
[2]School of Computer Science and Informatics, Cardiff University, Cardiff, United Kingdom
[3]Informatics Institute, University of Amsterdam, Amsterdam, Netherlands
[4]Faculty of Civil and Geodetic Engineering, University of Ljubljana, Ljubljana, Slovenia

{Salman.Taherizadeh, Vlado.Stankovski}@fgg.uni-lj.si, {JonesAC, TaylorIJ1}@cardiff.ac.uk, {Z.Zhao, P.W.Martin}@uva.nl

Abstract - *Many distributed time-critical applications have emerged on the Internet in recent decades, involving for example sensor-based early warning systems, online gaming and instant messaging. Such applications can be virtualised and distributed in a federated Cloud environment. Ensuring that these types of application are able to offer favourable service quality has been a challenging issue due to runtime variations in network conditions intrinsic to connections between individual application components replicated and distributed across different Cloud infrastructures. In this paper, we propose a lightweight method for performing network-level monitoring that can be used to guide the autonomous selection of optimal connections between running components, so improving distributed application performance at runtime. This solution contributes towards realising self-adaptation capabilities for time-critical applications by implementing a non-intrusive monitoring technique for key network-level parameters including round-trip time (RTT), packet loss, throughput and/or jitter. The experimental results show that the proposed framework has a low communication overhead and requires little processing power and memory capacity.*

Keywords: Monitoring System, Network QoS, Distributed Time-Critical Applications, Multi-Cloud Environment

1. Introduction

In recent years, time-critical systems such as early warning systems, multimedia applications and Cloud-based gaming have emerged as Internet services which are increasingly widely used and important, especially to organisations that want to leverage the benefits of distributed applications. Decomposing such complex applications, each application component can be distributed to a different machine such that each component interacts with other components regardless of deployment location. Accordingly, by using a multi-Cloud environment, companies can use Cloud infrastructures to run and replicate their application components in different locations.

Time-critical applications have specific network QoS (Quality of Service) requirements between their components, such as demanding minimal delay and packet loss, and require suitable support to achieve guaranteed application performance for their users. This is a challenge because the network connection quality between different components, as a key influencer of the overall application performance, is difficult to maintain when Cloud infrastructures continuously change. In particular, time-critical Cloud application providers have to dynamically adapt their services to network conditions to deliver high performance and a seamless experience. In essence, the main problem encountered by time-critical service providers is that there are limited automated and intelligent adaptation capabilities in existing Cloud infrastructures based on real-time network features that can be used to satisfy application performance requirements. Therefore, to avoid application performance issues, providers must carefully monitor the network QoS of connections within and between all of their own servers hosting application components in different Cloud infrastructures; all while being non-intrusive to the ordinary operation of the application [1].

This paper presents a lightweight monitoring approach based upon a non-intrusive design intended to enable distributed applications to autonomously reconfigure and adapt to changing network conditions at runtime. Replicating application components in different Cloud infrastructures to increase availability and reliability under various network conditions and varied amounts of traffic, and dynamically connecting each component to the best possible component in each different tier, together offering fully-qualified network performance, is often an essential requirement for providers of time-critical applications running on the Cloud. If such a network performance metric can be measured, then the system can be made automatically capable of improving the deployment of an application when performance drops. Under our proposed system, the network performance metric is a combination of measurements including network throughput, round-trip time, packet loss and/or jitter, which can be measured and responded in order to enhance application performance and hence user experience.

The rest of the paper is organised as follows. Section 2 presents summary of related work supporting network-level Cloud monitoring. Section 3 describes the use case. Section 4 discusses the architecture and implementation of our proposed approach, followed by empirical evaluation results and finally conclusion respectively in Sections 5 and 6.

2. Related Work

To achieve the objective of providing high-quality services in time-critical systems, it is essential to implement trustworthy techniques that can be responsible for maintaining QoS when considering the limitations imposed by the network. There have been many research approaches, all trying to provide QoS guarantees over Cloud networks. Anouari and Haqiq [2] analysed the performances of VoIP and Video stream traffic that is characterized by the ability to transmit real-time and interactively visual and auditory information. These types of traffic are highly delay-intolerant and need high priority transmission. Addressing this concern, their study was focused on using different service classes with respect to QoS parameters such as average delay, average jitter and throughput. Sodangi [3] designed and simulated two Cloud-based networks. The first scenario involved running multimedia applications (voice and video) and the second one involved running traditional applications (email, file transfer, web browsing). These were compared, and the main finding was that multimedia applications need appropriate throughput and are sensitive to delay, resulting in data loss, whereas traditional applications can use minimum throughput and with typical data loss levels are normally insensitive to changes in delay. In [4], the results showed that network performance varies substantially from one Cloud provider to another. Their approach can guide customers in selecting the best-performing provider for their applications. To measure the performance of internal connections between a customer’s instances and to the shared services offered by a Cloud, they used throughput and latency as metrics.

With regard to network-based measurement, associated QoS attributes change constantly and so network-layer parameters need to be closely monitored. Table 1 shows the most important metrics to be analysed for Cloud network measurement: (I) Throughput, which is the average rate of successful data transfer through a network connection. (II) RTT, which is the time elapsed from the propagation of a message to a remote place to its arrival back at the source. (III) Packet loss, which occurs when one or more packets of data traveling across a network fail to reach their destination. (IV) Jitter, which is the variation in the delay of successive packets.

Lampe *et al.* [5] mostly focused on the QoS parameter of latency, since this parameter plays an important role in the overall game experience. The authors conducted their research only on network latency measurement. Their experiments could be extended through the consideration of additional metrics; for example, the effects of network disturbances, such as increased packet loss or fluctuating throughput. Samimi *et al.* [6] introduced a model including a network-based monitoring system and the enabling of dynamic instantiation, composition, configuration and reconfiguration of services on an overlay network. Mohit [7] selected throughput, RTT and data loss for Cloud network measurement. The author suggests a solution that involves use of different technologies such as high-capacity edge routers which have a high cost and cannot be afforded in all use cases. Cervino *et al.* [8] presented an experimental validation of the Cloud infrastructure's ability to distribute streaming sessions with respect to some key streaming QoS parameters. Next, the authors performed experiments to evaluate the benefits of deploying VMs in Clouds to aid P2P streaming, by measuring the QoS improvement. Chen *et al.* [9] focused on the users’ perspective in Cloud gaming systems; from their point of view, the QoS metrics have an important effect on gaming experience. In other words, they proposed a suite of measurement techniques to evaluate the QoS of Cloud gaming systems.

Table 1. Relevant research on network-based measurement of Cloud environment performance

Title	Field	Measured Metrics	Results
To frag or to be fragged - an empirical assessment of latency in cloud gaming [5]	Audio/video stream	Limitations of the network infrastructure, such as high latency, potentially affect the QoS of the cloud gaming system.	While cloud gaming substantially reduces the demand of computational power on the client side, thus enabling the use of thin clients, it may also affect the QoS through the introduction of network latencies.
Service clouds: distributed infrastructure for adaptive communication services [6]	Adaptive communication services	Monitors carry out measurements on data streams. The metrics can be generic in nature (e.g., packet delay and loss rate) or domain-specific (e.g., jitter in a video stream).	Service clouds are distributed infrastructures which are designed to facilitate rapid prototyping and deployment of adaptive communication services in clouds, and they are appropriate choices when service platforms’ workloads are dynamic or they need a lot of resources.
A comprehensive solution to cloud-traffic tribulations [7]	General systems	Regarding network-based measurement, the three significant parameters to be analysed are throughput, RTT and data loss.	Computation-based infrastructure measurement is insufficient for the optimal operation and future growth of the cloud. Network-based measurements of the cloud computing service are also very important.
Testing a cloud provider network for hybrid p2p and cloud streaming architectures [8]	Online real-time streaming	Authors considered four very important network parameters for video/audio streaming and for many other real-time services: bandwidth, delay, jitter and packet losses.	Using a cloud network infrastructure to cross continents has improved the majority of QoS problems. It means that using connections between distant cloud datacentres can help to improve the QoS response of streaming even in videoconferencing P2P systems.
On the quality of service of cloud gaming systems [9]	Cloud gaming systems	Authors concentrate on the metrics related to network conditions namely delay, packet loss, bandwidth and also other types of metrics which are graphic quality and frame rate.	Packet loss and bandwidth limitations impose negative impact on the frame rates and the graphic quality in the cloud gaming systems. The network delay does not predominantly affect the graphic quality of the games on the cloud gaming systems.

3. Use Case

A typical example for time-critical services considers disaster early warning systems developed for the purpose of providing proper alert before disaster occurs. Figure 1 depicts the basic framework of such a system.

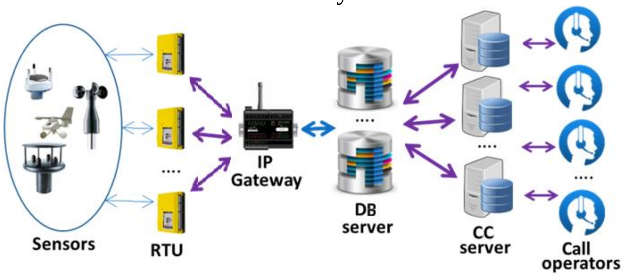

Figure 1. The basic framework of an early warning system

All application components are defined in Table 2. IP Gateway and RTU cannot be virtualized as these components have physical items like attached antennas.

Table 2. Components of a disaster early warning system

Component	Functionality	Type
Call Operator	The Call Operators decide whether or not to send an alert to emergency systems or to the public.	Dedicated and ad-hoc agents
CC Server (Contact Centre Server)	The server checks sensed data stored in DB Server and statistics in real-time and sends notifications (such as e-mail, SMS or voice call via SIP based IP telephony or ordinary PSTN) to Call Operators if values are outside predetermined thresholds for sensors.	Apache web server
DB Server (Database Server)	This is a Time Series Database which is used for storing and handling sensed values indexed by time.	Cassandra
IP Gateway	The IP Gateway is a node that allows communication between networks. It receives data over direct radio link or GSM/GPRS from sensors, aggregates the data and sends the data to the database.	E.g. TA900e or Cisco-ASA
RTU	Remote terminal units (RTUs) connect to sensors in the process and convert sensor signals to digital data.	E.g. Modbus-RTU
Sensors	Sensors can measure temperature, barometric pressure, humidity and other environmental variables.	E.g. DHT11

In this case, the overall application performance is the system's reaction time, which means the length of time taken from sensor data acquisition to when a notification is sent to the Call Operator. This application performance metric is mainly affected by the network communication quality between the DB Server and the CC Server. Due to the Cloud-based environment, several DB Servers and CC Servers can be running in various Cloud providers' infrastructures in different geographical locations, all connecting with each other. Assume that the data is replicated among DB Servers and also that each CC Server is dedicated to a certain number of Call Operators who must send warning messages through various communication channels in each region. The proposed mechanism aims at providing the ability to connect each CC Server to the best possible DB Server which has the superior network QoS in relation to the CC Server. Therefore, a Monitoring Probe is running on each CC Server's VM to measure the network performance metric (NPM) between the CC Server and every single DB Server. Our proposed approach shows how different Cloud providers can offer varying network performance in the execution of real-time applications depending on various aspects. We introduce (1) to calculate NPM including three important network parameters which are network throughput (NT), average delay (AD) and packet loss (PL).

$$NPM = \frac{\left(1-\frac{PL}{100}\right)*NT}{AD} \qquad (1)$$

In this use case, jitter is not taken into account; since this disaster early warning system is not a real-time service involving e.g. video/audio streaming in which lower jitter is advantageous (because lower jitter means the delay times are more consistent, and therefore a connection is more stable).

4. Architecture and Implementation

Cloud-based applications can be viewed from both design-time and run-time perspectives. In the design-time view, the whole Cloud service, including application topology and application components, is shown. In the run-time view, instances of application components are examined as they are deployed and executed in VMs. Considering these two views, Figure 2 presents an overview of the proposed architecture to make an effective improvement in the performance of the aforementioned disaster early warning system. In this figure, at run-time, for example there are three running CC Servers and two running DB Servers which are dynamically connected to each other in the best possible way to maximise the overall application performance.

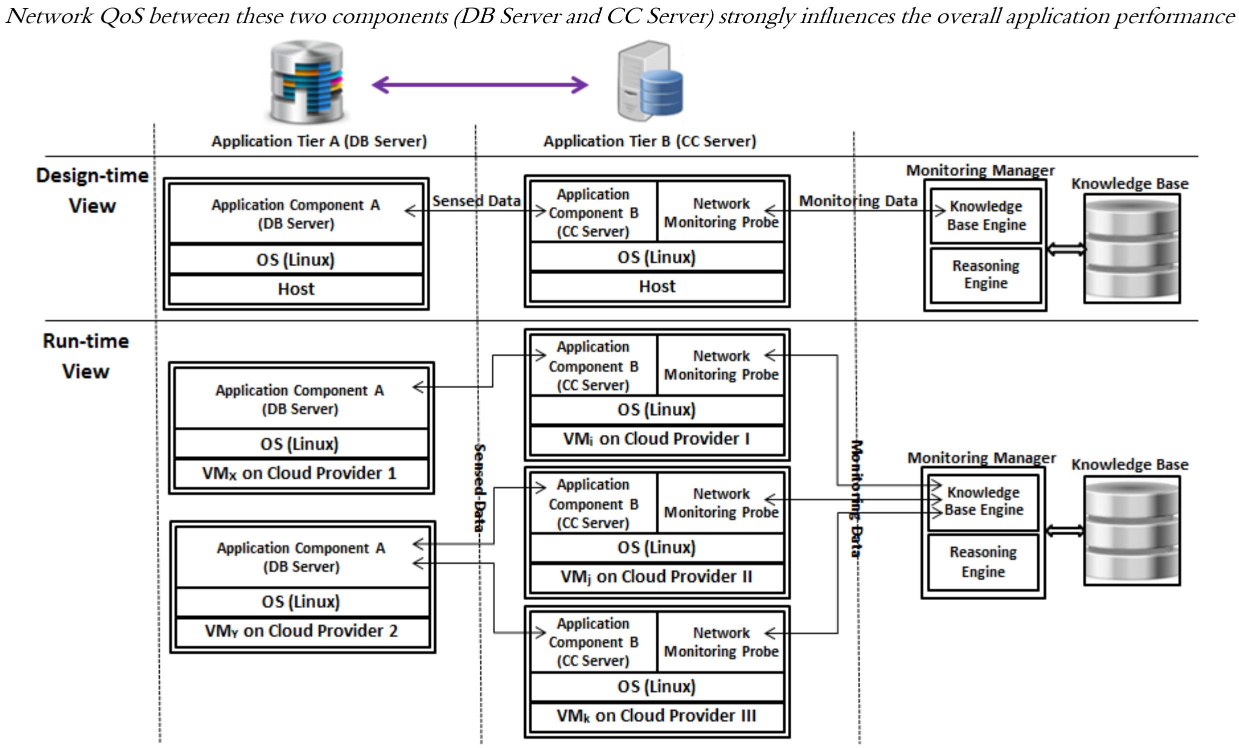

Figure 2. Overview of the proposed architecture to improve the performance of early warning system

```
/* Probe resides in $VM_i$ where the CC Server is running */
/* Packet Loss (PL), Network Throughput (NT), Average Delay (AD) */
/* Network Performance Metric: NPM */
while(true){
  TS ← TimeStamp()
  for each DB Server running on a $VM_x$ do {
    PL ← Calculate_PL($VM_i$, $VM_x$)
    NT ← Calculate_NT($VM_i$, $VM_x$)
    AD ← Calculate_AD($VM_i$, $VM_x$)
    NPM ← ((1 - (PL/100)) * NT)/AD
    Message ← Make_Message($VM_i$, $VM_x$, TS, NT, AD, PL, NPM)
    Send_To_Knowledge_Base_Engine(Message)
  } // end of for
  wait(interval)
} // end of while
```

Figure 3. Pseudocode for Monitoring Probe which is deployed along with the CC Server

As depicted in Figure 2, this monitoring system employs a number of distinct components. The *Network Monitoring Probe* is responsible for monitoring network QoS parameters of links between instances of two application components (the DB Server and the CC Server). For each CC Server, the network performance metric for every connection with potential DB Servers is simultaneously evaluated periodically at regular intervals by a Network Monitoring Probe. The pseudocode of the developed algorithm for the Monitoring Probe is depicted in Figure 3.

The *Monitoring Manager* is responsible for aggregating and analysing network QoS data received from Monitoring Probes. The Monitoring Manager consists of two parts; a Knowledge Base Engine and a Reasoning Engine. The *Knowledge Base Engine* is responsible for all the work that controls the collection of network QoS values as RDF (Resource Description Framework) triples, along with actually storing and also retrieving these data on disk. This proposed monitoring system incrementally stores information about the environment in a *Knowledge Base* (KB) that will be used for interoperability, integration, analysing and optimisation purposes. Maintaining a KB enables analysis of long-term trends, supports capacity planning and allows for a variety of strategic analysis like year-over-year comparisons and usage trends. The *Reasoning Engine* is responsible for network-based QoS analysis and evaluating relevant policies such as interpreting the network performance metrics between CC Servers and DB Servers. Therefore, based on network-based analysis, the Reasoning Engine will return decisions such as which CC Server should be automatically and dynamically connected to which DB Server when current conditions do not satisfy the expected requirements. Each alternative possesses different attributes which can be compared and evaluated using network-level criteria; the proposed framework via the Reasoning Engine can then choose the best one at real-time.

For our experiments, the actual network QoS parameters for time-critical services are measured by using ICMP (Internet Control Message Protocol) requests. The “ping” tool operates by sending echo request packets to the target host and waiting for an echo reply packets. It measures the round-trip time from transmission to reception and reports errors and packet loss. We used different command options to enable the monitoring system to adjust the size of the ICMP packet, determine the number of echo requests to send, and specify wait period between pings. Moreover, we used an option to set the “Do Not Fragment” bit on the ICMP packet which does not allow fragmentation to occur in the path of the data flow by intermediate routers. We implemented the Knowledge Base Engine using a Jena Fuseki server to load an

RDF dataset and make it accessible through a REST API as a SPARQL endpoint, to expose the CRUD operations for creating, retrieving, updating and deleting records. Jena Fuseki is an open source, lightweight database server, easy to install and able to efficiently store large numbers of RDF triples on disk [10].

5. Empirical Evaluation Results

As a preliminary set of proof-of-concept results to test the design of the monitoring components, we performed an initial set of experiments to measure the network-based metrics between a particular CC Server (Host$_i$) and two replicated DB Servers (Host$_x$ and Host$_y$) at runtime. Periodically (every 10 seconds), our Network Monitoring Probe deployed on the VM, hosting also the CC Server, sends 10 ICMP packets to the both DB Servers with a 0.2 second delay between sending each packet, and then calculates the network metrics. The CC Server will then be automatically connected to the DB Server providing the highest connection quality.

The following experiment shows how this configuration allows us to check the network-based QoS features related to two different connections with the same source: the first link between Host$_i$ and Host$_x$ and the second one between Host$_i$ and Host$_y$. Table 3 shows features of these three hosts. Host$_i$ which is a CC Server, belongs to the Flexiant Cloud infrastructure in the United Kingdom. Two DB Servers—Host$_x$ and Host$_y$—are in different locations in Slovenia and belong to different Cloud infrastructure providers: the ARNES (the Academic and Research Network of Slovenia) and the FGG (the Faculty of Civil and Geodetic Engineering, University of Ljubljana).

Table 3. Features of infrastructures used in our experiment

Feature	Host$_i$	Host$_x$	Host$_y$
Type	CC Server	DB Server	DB Server
OS	Ubuntu 14.04	Debian 7.8	Ubuntu 14.04
CPU(s)	2	1	1
CPU MHz	2600.030	2666.760	2397.222
Memory	1024 MB	1024 MB	1024 MB
Speed	1000 Mbps	1000 Mbps	1000 Mbps
IP	109.231.121.55	193.2.91.109	194.249.0.142
Cloud	Flexiant	FGG	ARNES

The round-trip delay gives the total end-to-end time, and hence is an important metric in evaluating the performance of the time-critical Cloud service. A lower average delay is always preferred; because it takes less time for packets to reach and return between the servers. Therefore, Figure 4 shows that according to the average delay, the network quality of Host$_x$ is a little bit better than that of Host$_y$ for a period of time.

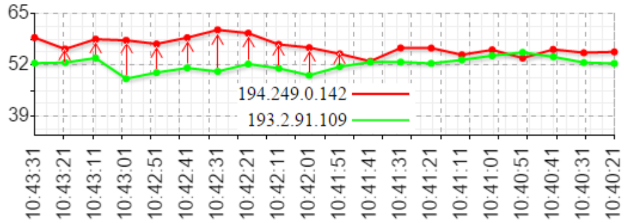

Figure 4. Average Delay (ms) for 200 second monitoring window

Time-critical Cloud applications require network services with minimal packet loss. The possibility of packet loss increases as traffic travels a longer distance and over more hops in the network. Data loss has one of the biggest impacts on time-critical applications, seriously affecting the quality of services, and this is the reason that the network should be engineered for zero percent packet loss. Our test system showed that packet loss ratio was zero, which indicates that there was no drop in either connection related to the servers deployed during the experiment.

Network throughput is the amount of data moved successfully from one place to another in a given time period. It is possible to benchmark network throughput and find bottlenecks in the network to ensure that network interfaces are fast enough to achieve desired performance. The amount of traffic in current high-speed, heavy-traffic and multi-service networks increases continuously, and traffic characteristics change heavily in time—for example network throughput fluctuates due to time of day, server backup operations, DoS (Denial of Service) attacks, scanning attacks and other anomalous network traffic. The performance of Cloud services must be independent of such states and must continue to behave reliably in all possible cases. Our proposed monitoring system sends ICMP packets, each one containing 500 bytes of data, from the first node (CC Server) to the second node (DB Server). Then it receives the results including the average delay (“*Avg*”). To make the proposed monitoring system lightweight, network throughput was estimated from the latency based on (2), which converts bytes per millisecond into kilobytes per second:

$$\textit{Network Throughput (KB/s)} = \frac{500 * 10^{6}}{\textit{Avg} * 2^{10}} \qquad (2)$$

Figure 5 shows no major variation in throughput belonging to either server; however in real-time systems, continuous fluctuation is important to be taken into account.

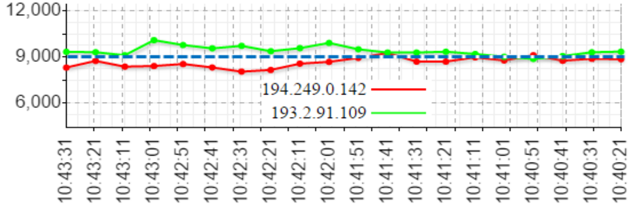

Figure 5. Network Throughput (KB/s) for 200 second monitoring window

Finally, regarding NPM, Figure 6 shows that Host$_i$ has better network performance quality with Host$_x$ compared to Host$_y$ during the last 10 intervals. Therefore, if Host$_i$ is connected to the Host$_y$, adaptation should occur and thus Host$_i$ will be connected to the Host$_x$ instead of Host$_y$.

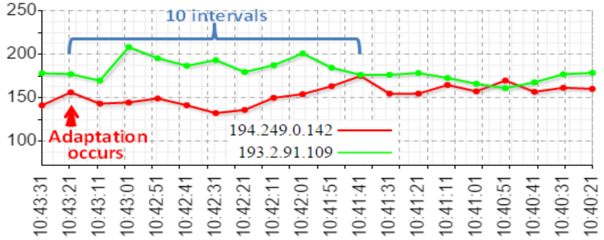

Figure 6. Network performance metric (NPM) for 200 second monitoring window

By employing only the last measurement explained above, this metric can have significant effect on the application performance and hence users' satisfaction; Cloud services for time-critical applications can automatically optimise the process of choosing the best possible application components, which are responsible for offering acceptable network QoS.

A challenge in designing a monitoring framework in the Cloud environment is ensuring that the overhead of the monitoring system is kept to the minimum [11]. The distributed nature of proposed monitoring framework quenches the runtime overhead of system to a number of Monitoring Probes running across different VMs. A detailed view on the resource consumption of the Monitoring Probe revealed that our approach is lightweight in terms of CPU and memory overhead. To confirm this, we applied the “top” tool which provides a dynamic real-time view of tasks currently being managed by the Linux kernel. Our running Monitoring Probe consumes only 0.3 percent of the whole CPU time and 3.1 percent of the whole memory usage in average.

Furthermore, comparing with the average network throughput of CC Server, the running Monitoring Probe consumes a small fraction of network bandwidth. To this end, we parsed the output of “nethogs” tool to estimate the bandwidth overhead introduced by our Monitoring Probe. We found out our Monitoring Probe transmits 1282944 bytes during 15 minutes, which means ~712 bytes per second for every DB Server in average.

Since the architecture includes a knowledge base, average “write” performance in milliseconds for the Fuseki backend implementation was calculated. The Fuseki server has one CPU 2397 MHz and 2GB total memory. During 15 minutes, 90 “write” queries were executed for each DB Server and the average query execution time was 3.93 ms.

6. Conclusion

In distributed time-critical Cloud applications, network-level features such as throughput and latency of packets travelling between application components directly affect user experience. Therefore, time-critical service providers must constantly monitor the network performance between their current servers running on different Cloud infrastructures, and other alternatives. In this way, preventing and predicting potential network performance drops related to the connections between the servers or possible overloads in the system will give more time to take action like dynamically changing connectivity topology among running components and switching from one server to another server to adjust the system in an anticipatory manner.

This research paper presented a lightweight network-based monitoring approach that is particularly suitable for autonomously adapting distributed time-critical Cloud applications. The lightweight feature for the implemented monitoring approach is a significant property in Cloud computing environments because of the necessity of being non-intrusive to the normal flows of application. The proposed solution is general and extensible, and it can be applied to any distributed Cloud application. The goal of the paper was to investigate network QoS properties that are especially important for the development of modern time-critical Cloud applications. We extend the current state-of-the-art by proposing a turnkey approach that not only monitors network QoS, but also stores the monitoring information, processes it, and integrates it with other system information for controlling the overall performance.

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).

8. References

[1] Ma, K., Sun, R., and Abraham, A., "Toward a lightweight framework for monitoring public clouds"; Proceedings of 4th International Conference on Computational Aspects of Social Networks (CASoN 2012), Brazil, Pp. 361—365, 2012.

[2] Anouari, T. and Haqiq, A., "Analysis of VoIP and Video Traffic over WiMAX Using Different Service Classes"; Journal of Mobile Multimedia, Vol. 9, No. 3&4, 2014.

[3] Sodangi, L. S., "Distributed Multimedia Applications in Quality of Service for Wireless Wide Area Network"; International Journal of Engineering Research and Technology (IJERT), Vol. 2, No. 10, Pp. 4088—4104, 2013.

[4] Li, A., Yang, X., Kandula, S., and Zhang, M., "Cloudcmp: comparing public cloud providers"; Proceedings of the ACM SIGCOMM conference on Internet measurement, 2010.

[5] Lampe, U., Wu, Q., Hans, R., Miede, A., and Steinmetz, R., "To Frag Or To Be Fragged - An Empirical Assessment of Latency in Cloud Gaming"; 3rd International Conference on Cloud Computing and Services Science, 2013.

[6] Samimi, A. F., McKinley, P. K., Sadjadi, S. M., Tang, C., Shapiro, J. K., and Zhou, Z., "Service Clouds: Distributed Infrastructure for Adaptive Communication Services"; IEEE Transactions on Network and Service Management, Vol. 4, No. 2, Pp. 84—95, 2007.

[7] Mohit, M., "A comprehensive solution to cloud traffic tribulations"; International Journal on Web Service Computing, Vol. 1, No. 2, Pp. 1—13, December 2010.

[8] Cervino, J., Rodriguez, P., Trajkovska, I., Mozo, A., and Salvachua, J., "Testing a Cloud Provider Network for Hybrid P2P and Cloud Streaming Architectures"; IEEE International Conference on Cloud Computing, Pp. 356—363, 2011.

[9] Chen, K. T., Chang, Y. C., Hsu, H. J., Chen, D. Y., Huang, C. Y., and Hsu, C. H., "On the quality of service of cloud gaming systems"; IEEE Transactions on Multimedia, Vol. 16, No. 2, Pp. 480—495, February 2014.

[10] Roda, C., Navarro, E., and Cuesta, C. E., "A comparative analysis of Linked Data tools to support architectural knowledge"; ISD2014 International Conference on Information Systems Development, 2014.

[11] Aceto, G., Botta, A., De Donato, W., and Pescape, A., "Cloud Monitoring: definitions, issues and future directions"; IEEE 1st International Conference on Cloud Networking (CLOUDNET), Pp. 63—67, November 2012.