Estimation of Synchronization Time in Cloud Computing Architecture

Fidan Kaya Gülağız[(✉)] and Onur Gök

Department of Computer Science, Kocaeli University,
Umuttepe, 41380 Izmit, Kocaeli, Turkey
{fidan.kaya, ogok}@kocaeli.edu.tr

Abstract. Size of the electronic data is constantly increasing with today's technology. Distribution of this data is provided via servers or cloud servers. There are some restrictions caused by network traffic and network infrastructure between these servers. Some of these restrictions can be listed as bandwidth, packet transmission rate, number of users that can be simultaneously answered. These are cause problems about data traffic and efficient transfer of data. In this thesis study, it is aimed to develop an efficient data synchronization system architecture that is compatible with distributed proxy server/cloud server architectures. Thus, it is aimed to optimize the traffic of created by data synchronization.

Keywords: Distributed systems · Data synchronization · Nosql · CouchDB · Cloud computing

1 Introduction

Along with the developments in technology, data is transferred to electronic environment. The size of the data transferred to electronic environment is increasing day by day. The increasing size of data causes traffic in the some network architectures. In case of the increased traffic, delays take place between the ends where data is transmitted and re-transmissions take place because of the delays. There is a need for proxy server based studies in order to overcome this problem or decrease the effect of this problem.

In this thesis, a synchronization method which based on proxy server/cloud server architecture is suggested. In this architecture, proxies does not contain same data. Each proxy server contains information belonging to its subnet and only synchronize its data with cloud server not other proxies. However, the problem of consistency arises in such architectures due to the storage of data in different servers (proxy and cloud servers). In order to ensure the consistency of data, mechanisms of data synchronization is needed. For this purpose, the synchronization traffic that will emerge between proxy servers and cloud servers should be optimized. In the architecture to be developed, a dual synchronization process is needed. A dual synchronization will be conducted both from cloud server to proxy server and from proxy server to cloud server. In these architecture cloud server will collect whole data which stored each proxy and proxies will store their specific data.

In order to store the data, a noqsl database CouchDB will be used. It is aimed to increase the speed of access to the data by using a noqsl database. CouchDB database

architecture has a synchronization module in itself. With this module, it conducts the synchronization process via multi-master computing, which is a subtype of master-master replication. However, the mapping process here can be conducted either on-demand (instant) or in permanent mode. In case of the preference of permanent synchronization mode, dual synchronization of the selected servers is carried out in one minute intervals. However, in case of an increase in the number of data and proxy servers to be synchronized, this process will cause a serious load on the system.

A system should be developed that will carry out the synchronization process by determining the most proper timing and without increasing the load on the system. In this thesis, it is aimed to develop a flexible architecture that is compatible with distributed proxy server- cloud server architectures and will provide the synchronization of data as frequent as possible.

The rest of the article is structured as follows. In the second section, proposed method is given. In the third section, the effect of background traffic is given for proposed distributed network architecture. Fourth section explains the selected parameters for network analysis. Fifth section explains the simple network architecture formed by OPNET. Conclusion and some future enhancements are given at the conclusion section.

2 Proposed Method

The main objective of the method to be developed is to create an architecture that will be alternative for current synchronization methods in the CouchDB database. In CouchDB database, there are three different type synchronization methods. These are can be listed as long polling, triggered mode and continuous mode. We want to develop a new and intelligent polling architecture to CouchDB database. This architecture will also determine the timing of synchronization according to network congestion in the application layer. This architecture will detect the appropriate polling period for each subnet according to their historical data. Also these data will give information about user behavior in related subnet and intelligent polling method will detect synchronization time according to user's behavior. The steps to be followed for the development of the system can be listed as below;

- Establishment of a network to simulate the distributed network architecture
- Obtaining network parameters to be used for analysis during a specific time period
- Using profile hidden markov model to analyze the obtained parameters
- Processing a synchronization control algorithm to detect synchronization time
- Repeating the second and third steps until all network is being modeled

As it can be understood from the steps, the method to be developed has two basic modules. These are the construction of distributed network architecture and determination of mapping time. A detailed description of these modules is given in Sects. 2.1 and 2.2.

2.1 Distributed Network Architecture

The overall view of distributed network architecture to be designed is presented in Fig. 1. In the architecture, N number of proxy servers communicates with the cloud server. This communication is carried out via an internet cloud. The architecture seen in the figure is giant network architecture and it is difficult to be designed in real life. Therefore the architecture will be modeled in OPNET environment. To be compatible with real life, after the model is set up, packet size, packet transmission frequency and packet delay values will be determined according to the appropriate probability distributions. Then, background traffic at different rates will be added to the link between cloud server and internet cloud. In this way, all probable situations of the network will be tried to be modeled. When all the processes are carried out, the network architecture to work on will be obtained.
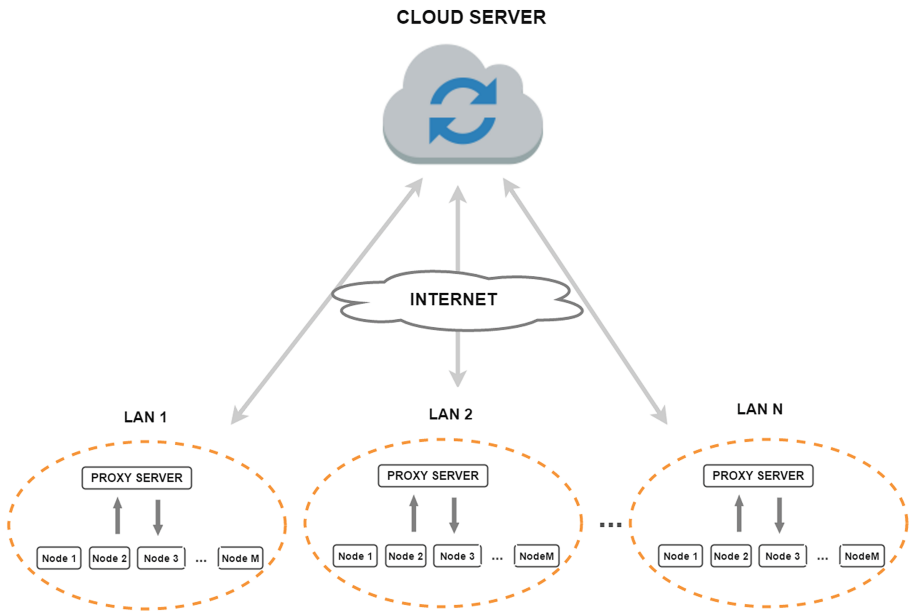

Fig. 1. General distributed network architecture

After the architecture is designed, different data sets that model different congestion situations in the network will be obtained. Via these data sets, an architecture that will instantaneously model the situation of the network in a specific time will be designed. Profile Hidden Markov Model will be used for the learning process on the obtained data sets. In this way, the most appropriate times of the network for synchronization will be detected by using the previous periods.

2.2   Detecting Synchronization Time

There are two main methods for data transmission in the given internet environment. These are poll and push techniques. In the poll technique, requests are sent to server from client and information as an answer is taken from server. Poll method is a more traditional method than push method. It is generally used to carry out planned tasks in a time slot. As an example for the uses of the poll method, applications in which delays in data can be neglected and data transfers in giant sizes can be given. In the push technique, the server informs the client about the changes in information. At the same time, the transfer of the updated information can also be carried out as a part of the informational message. Push method is more often used for real time applications. The loading of updates in mobile phone applications or applications in which data is processed in only one.

In the study to be conducted, the data to be transferred is giant size data called as bulk data. For this reason, the use of poll method will be more appropriate. However, there is a relationship between the practicality of the data obtained in poll technique and the traffic that emerges. Factors that negatively affect the traffic are listed below.

- The number of devices by which poll process is conducted.
- The number of objects needed for each poll process.
- The frequency of poll process.
- The current bandwidth and congestion.

Considering these factors above, it is needed to distribute the load for wide networks and categorize the local poll process [1]. Many researchers have examined the issue that polling process places a burden on the system and it requires planning. Lv and his colleagues conducted a study for the timing of poll process in optical access networks [2]. Jacob and his colleague developed a polling mechanism that was planned for wireless body area networks. In the study, a polling method that provides discrete-time access was suggested. The latency times of the transmission knots was tried to be predicted according to Karn's algorithm logic. With the method developed, it was shown that more realistic latency times could be obtained [3]. It is important in cloud computing architecture whether poll or push technique is used. These methods are used for mapping data between especially mobile devices and cloud server in cloud computing. In mobile devices, poll method is generally needed for the first update, but afterwards, the transmission of updates can be conducted via push method. In a study carried out by Carvalho and his colleagues, evaluation of the use of poll and push methods between cloud server and mobile devices was conducted. At the end of the study, it was found that if the application doesn't have to send more than one request in 40 min, it is appropriate to use the poll technique [4]. Li and colleagues stated that Android applications in mobile devices use traditional poll methods. They found that when the data draining frequency of the applied poll method is not detected accurately, problems such as information delays, unnecessary consumption of network traffic or unnecessary battery consumption in mobile devices may emerge [5]. Similarly, Fang and colleagues also suggested an architecture that provided the timing of broadcast poll approach fixed on mobile devices [6]. Saxena and colleagues suggested a hybrid method in order to shorten the access time to data in mobile devices. They provided a

probability based approach that tried to predict the next poll time or push time separately. With the method developed, access time to data under high system load was shortened. Therefore, they showed that scheduled mechanisms are very useful for access to data [7].

The CouchDB database that will be used in the suggested architecture carries out the synchronization process via three different mechanisms. These can be listed as continuous, triggered and long polling.

Continuous mechanism, as it can be understood from its name, runs on a permanently open connection. This open connection continuously listens to CouchDB's Changes API, which is the modified API of CouchB. When a modification takes place in any of databases that will be synchronized, it automatically starts the synchronization process. When this method is conducted, real time synchronization is provided.

The second mapping mechanism, long polling, is the most effective way of polling process. The server waits for a fixed time before it sends the response in case there will be a modification. Therefore, it eliminates the unnecessary polling processes when there is no modification. The method to be used before long polling is short polling method. In the short polling process, polling is continuously carried out in fixed intervals. Long polling has a longer poll time and aims to send modifications collectively. However, short polling mechanism causes congestions in traffic. In this method, there is a certain pre-scheduled transmission time between the transferred packets and if there are delays in the network, the time between two control periods will be insufficient for transmission [8].

Triggered method is an alternative method for continuous and long polling methods. CouchDB runs on triggered mode as default. This system also works with short polling logic; however it is the type of short polling that is triggered at a time. Only the CouchDB administrator can carry out the poll process on demand. The disadvantage of this method is that it requires constant triggering. Mapping will not be carried out without being triggered even if there is a modification.

In this thesis study, a different method that will be an alternative for the three synchronization mechanisms in the CouchDB structure is suggested. This method will be a smart poll method. The frequency period of poll time will be modified according to the situation of network by the use of this method. For this process, data that is saved from the network at different periods and Profile Hidden Markov Model will be utilized. Therefore, no time will be spent for synchronization when traffic or packet transfer to cloud server is condensed.

3   Effect of Background Traffic on Network

In the suggested system, data to be sent has to transmit via internet cloud in order to reach the server after leaving from LAN (Local Area Network) environment. Considering that internet is a giant environment and it is shared by a lot of different networks, the background traffic will have a serious effect on the application. Therefore, it is necessary to consider the effect of background traffic in the design process of the distributed system architecture to be developed.

Several researchers have studied how and to what extent the background traffic has an effect while developing distributed architecture systems. In a study conducted by Venkatesh and Vahdat [9], it was shown that the traffic stream in the background affects the service and protocol behaviors. They showed that synthetic traffic models that were used previously don't represent the effect of the traffic that is seen in real life, and therefore, sufficient congestion sampling cannot be obtained. Applications behave differently when they have to compete with the background traffic. Therefore, they carried out behavior analysis for both synthetic background traffic and real background traffic of different applications. As a result, they found out that every application is affected by traffic congestion to some extent depending on the type of the application. Venkatesh and Vahdat [10], tried to conduct a new research with software called Swing to obtain a more realistic traffic in the following years. In this study, they showed that structural traffic models special to applications can be created successfully with Swing software. It was proved that the traffic created by Swing is a realistic traffic under different application, network and user circumstances. In another study carried out by Nahum and colleagues [11], it was shown that WAN conditions have a serious effect on servers' performances. In this study, they showed that lost packets decrease the performance of servers by 50% and increase the response time at the same rate. In real life, every server in a WAN environment faces problems such as packet losses and delays. For this reason, while evaluating the performances of servers, WAN characteristics should also be considered. In the study, it is stated that packet losses take place more often when the internet is burst. Eylen and Bazlamaçı [12] attempted to develop a system to be able to measure unicast delays without the need for clock mapping. In this study, they stated that background traffic is needed to obtain traffic similar to real life traffic. Therefore, random delays were added to trial packages used in the study. In this way, three background traffics at different rates were created by using Poisson distribution. The aim here is to provide congestion at different rates. The first situation is a high loaded traffic situation called as "heavy load/high load". In this situation, the link via which the transmission will be done is overloaded. In other words, more background traffic is injected to the network than the capacity of the link. There will be congestion in this situation and queue size will increase. The second situation is "low load" situation. In this situation, packet transmission was tried to be done with a null transmission queue. The density of background traffic is much less than the capacity of the link. The aim of creating traffic at this rate is to catch instant modifications in delays. The third situation is the traffic rate called as "silence rate". In this situation, the rate of the traffic created equals to half of the capacity of the link. The reason for creating this situation is to determine the presence of packets that aren't delayed. Real traffic was modeled with the three types of traffic created. The analysis of the developed method was carried out more accurately under background traffic in this way. In another study conducted by Levesque and Tipper [13], background traffic was created to provide accurate predictions of point-to-point mapping time under asymmetric delay conditions. The main reason for queue delays in the network is the general traffic load of the network. If traffic load is considerably low, mapping errors will decrease at the same rate. Mapping errors will increase along with the increase in traffic load. The analysis of the developed method has been tested in different traffic conditions and the effect of background traffic is clearly shown.

In these studies, it has been found out that background traffic has a serious effect on both applications and servers. The dimension of the effect depends on several parameters. Therefore, in order to conduct analyses of applications realistically, background traffic should certainly be taken into consideration.

4 Determination of Parameters for Network Analysis

Packet losses and re-transmissions take place when the traffic is condensed. Congestions occur in the points where the traffic is too condensed. Different versions of current TCP transmission protocols can detect congestion situations with different ways [14]. TCP Reno, New Reno, Tahoe and SACK versions use packet losses as congestion signals. However, noticing a packet loss can take a very long time for congestion. During this time, more packet losses may occur due to filling of router's buffer. TCP Vegas, which is the last version of TCP, tries to do congestion control relying on round time value. In TCP Vegas, pending throughput and real throughput computations are done relying on RTT. The frame size is calculated according to results obtained from this process. However, the inappropriate choice of parameters in frame size calculation results in packet losses. TCP Vegas allows packet retention between 1–3 intervals in congestion queue. When background traffic increases, this situation causes an unnecessary obstruction.

Mechanisms that can produce a solution before congestion occurs by detecting the congestion situation in application level are needed for congestion detection and more effective use of congestion prevention mechanisms in TCP. Solutions for congestion situations in application layer can be listed as three items [15]. Response packets that servers send to clients can be sent in big sizes. In this way, the number of packets to be sent can be decreased. Data transmission should be schedulable so as not to collide with each other and users who carry out simultaneous transmissions can be restricted.

The congestion situations should be detected before occurring or shortly after occurring for the application of solutions suggested in this part. For this reason, the parameters to be used for accurate detection of congestion situations should be determined appropriately. There are software such as Netflow [16] and Sfolw [17] developed by Cisco for congestion management in the application level. These software use the following items as parameters for congestion control:

- Response time,
- Accessibility of network resources,
- Application performance,
- Jitter for video data transfer,
- Connection time,
- Throughput and
- Packet losses

In addition to these, parameters to be used in networks with a heterogonous structure for detection of congestion are listed by Floyd as following [18].

- Throughput
- Delay
- Lost packets
- Response time

These parameters are common parameters that are used for congestion detection and congestion prevention [19]. After background traffic at different rates is created in our study, the required congestion detection parameters of transmitted trial packages must be registered. Four different parameters will be used in this step. These are throughput, delay, re-transmission numbers and response time. These parameters are standard parameters used by Cisco for congestion detection. In this step, the number of packets pending in the queue, pending time of packets in the queue and the numbers of re-transmission of packages are aimed to be decreased in the analysis conducted with congestion parameters.

5 Forming Network Using OPNET

A WAN network has to be set up in order to obtain the data to be used for the study. The screenshot of the network set up is given in Fig. 2. In the figure, we gave sample architecture. After we obtained first results, subnet count will be increased and method will be tested in more realistic conditions. There are four subnets in the sample network. These are named as Subnet Istanbul, Test Subnet, Samsun and Erzurum. The subnet in Istanbul stands for the subnet where the cloud server exists. Test Subnet stands for the subnet where the network's values of throughput, delay, response time and re-transmission number will be registered. There is a proxy server here. The four parameters mentioned were registered via the probe packets sent from this proxy server.
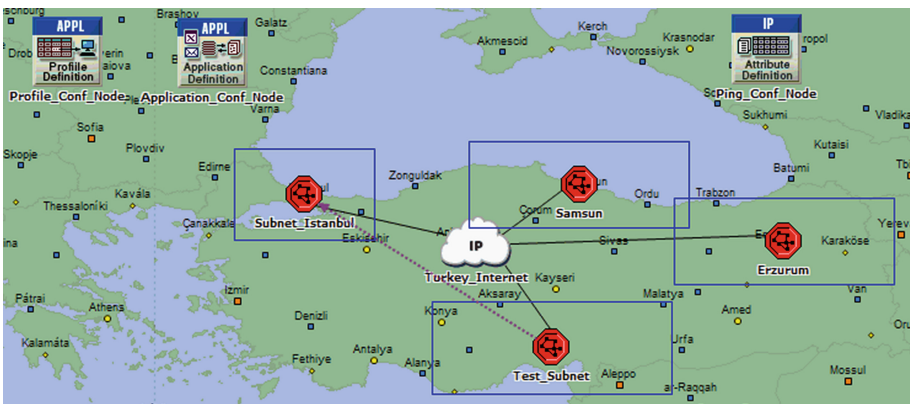

Fig. 2. Sample network architecture that is built using OPNET

The IP Cloud in Fig. 2 stands for the internet in Turkey. In order to make the simulation realistic, latency at different rates was added to the IP cloud and data was obtained according six different latency values. Packet discard ratio value was determined as 0.1. This value equals to loss ratio in cabled networks [20]. These two parameters should appropriately be modeled in order to model the real traffic accurately [21, 22]. In order to add random latencies to probe packets to be sent, random latencies were added to the link via which the server is connected to the IP cloud. These latencies were added according to Poisson distribution and implemented at three different ratios. The reason for adding latencies is to create congestion and then, solve the congestion situation [12]. The first one of the three ratios stands for high load case when the link is overloaded. In this situation, more background packet than the capacity of the link was injected to link. The second situation stands for low load case. In this situation, the traffic created is much lower than the capacity of the link. The third situation is silence rate situation. In the third situation, the traffic rate is half of the capacity of the link [23].

These three traffic ratios were applied with round bin logic during the simulation process. In the first situation, the situation in which the transmitter queue size increases was modeled by the application of high load situation. Then, low load situation was applied as the second step. The transmitter queue was decreased in this way. Silence rate situation was applied as the last step. Therefore, these three situations were modeled under the conditions of different latencies that may occur in IP cloud. The matlab code that was created in order to add traffic to the link is given in Table 1.

Table 1. Matlab code to generate background traffic

High Load	Low Load	Slience Rate
x=0;	B= poissrnd(900000,1,60);	C=poissrnd(30000000,1,60);
A=poissrnd(50000000,1,60);	for i=1:60	for i=1:60
yaz=fopen('dosya','w+');	t = num2str(B(i));	t = num2str(C(i));
fprintf(yaz,'seconds	t2=num2str(x);	t2=num2str(x);
bits/second\n');	t3='\t';	t3='\t';
for i=1:60	t5='\n';	t5='\n';
t = num2str(A(i));	t4 = strcat(t2,t3,t,t5);	t4 = strcat(t2,t3,t,t5);
t2=num2str(x);	fprintf(yaz, t4);	fprintf(yaz, t4);
t3='\t';	x=x+60;	x=x+60;
t5='\n';	end	end
t4 = strcat(t2,t3,t,t5);		
fprintf(yaz, t4);		
x=x+60;		
end		

After the background traffic is added, it is necessary to determine according to which distribution the packets to be sent from subnets in Samsun and Erzurum will be transmitted. There are several probability distributions in literature for modeling different situations of the network. However, the appropriate parameters for these distributions vary along with developments in technology. Distributions that can be

appropriate for different web packets and different traffic densities were determined by
Garsva and colleagues in a study in 2014. According to the study, it was found out that
the use of Pareto distribution when the traffic is condensed and the use of Weibull or
Lognormal distributions when density is low are appropriate for sizes of transmitted
data [24]. For this reason, three different packet transmission applications that were
appropriate for the three different background traffics created were designed.

After the packet sizes mentioned above were applied to the subnets in Samsun and
Erzurum, load, throughput, response time and re-transmission number parameters of
probe packets sent from test subnet were saved in a text document. A sample
screenshot of the data obtained is given in Table 2.

Table 2. A part of obtained dataset

Time (sec.)	Response time (sec.)	Delay (sec.)	Load (bytes/sec.)	Retransmission count
0.0	#N/A	#N/A	0	#N/A
1.2	#N/A	#N/A	0	#N/A
2.4	#N/A	#N/A	0	25
3.6	#N/A	#N/A	0	50
4.8	#N/A	#N/A	0	50
6.0	#N/A	#N/A	0	75
7.2	#N/A	#N/A	0	51
8.4	#N/A	#N/A	0	100
9.6	#N/A	#N/A	0	100
10.8	#N/A	#N/A	0	100
12.0	0.128293420847	0.063659785183	32000.000000000018	150
13.2	0.154222547248	0.071516954689	67840.000000000044	101
14.4	0.188230963541	0.083656841361	67413.333333333372	150
15.6	0.221081878731	0.115675131047	69119.999999999942	53
16.8	0.247895827499	0.115175899488	46080.000000000029	51
18.0	0.272231176872	0.130691457222	46080.000000000029	51
19.2	0.294266277436	0.113949038065	25173.333333333347	#N/A
20.4	0.307123112638	0.11617685566	26453.33333333335	26

6   Results

In this thesis proposal, a new method is suggested in order to solve the synchronization
problem in distributed network architectures. The main aim of the method is to add an
adaptive poll to CouchDB database, which is a noqsl database.

With the architecture to be developed, it is aimed to decrease the unnecessary
delays and re-transmissions of packets that may occur in proxy server- cloud server
architectures that require dual synchronization. In this part of the study, distributed
network architecture is implemented with low server numbers on OPNET. In this way,
data sets belonging to different time periods are obtained considering the different
situations of the network.

In the second part of the study, firstly, it is aimed to obtain the missing values in the obtained data with statistical methods. Then, by using the Profile Hidden Markov Model, different models will be designed for data belonging to different periods. The congestion situations in OPNET environment will be detected from the application layer via these models. Therefore, the synchronization time will be planned adaptively according to the situation of the network.

References

1. Kenyon, T.: Data Networks: Routing, Security and Performance Optimization. Digital Press, Newton (1960)
2. Lv, Y., Jiang, N., Xue, C.: Energy-efficient load adaptive polling sequence arrangement scheme for passive optical access networks. IEEE/OSA J. Opt. Commun. Networking **7**, 516–524 (2015)
3. Jacob, A.K., Jacob, L.: A discrete time polling protocol for wireless body area network. In: IEEE International Advance Computing Conference, pp. 294–299. IEEE Press, India (2014)
4. Carvalho, S.A.L., Lima, R.N., Silva-Filho, A.G.: A pushing approach for data synchronization in cloud to reduce energy consumption in mobile devices. In: Brazilian Symposium on Computing Systems Engineering, Brazil, pp. 31–36 (2014)
5. Li, P., Chen, Y., Li, T., Wang R., Sun, J.: Implementation of cloud messaging system based on GCM service. In: IEEE International Conference on Computational and Information Sciences, China, pp. 1509–1512 (2013)
6. Fang, Q., Vrbsky, S.V., Dang, Y., Ni, W.: A pull-based broadcast algorithm that considers timing constraints. In: International Conference on Parallel Processing Workshops, Canada, pp. 46–53 (2004)
7. Saxena, N., Pinotti, C.M., Das, S.K.: A probabilistic push-pull hybrid scheduling algorithm for asymetric wireless environment. In: IEEE Global Telecommunications Conference Workshops, USA, pp. 5–6 (2004)
8. Aron, W., Druschel, P.: TCP implementation enhancements for improving webserver performance. Technical report TR99-335, Rice University (1999)
9. Venkatesh, K., Vahdat A.: Evaluating distributed systems: does background traffic matter? In: USENIX Annual Technical Conference, Boston, Massachusetts, pp. 227–240 (2008)
10. Venkatesh, K., Vahdat, A.: Swing: realistic and responsive network traffic generation. IEEE/ACM Trans. Networking **17**, 712–725 (2009)
11. Nahum, E.M., Roşu, M.C., Seshan, S., Almedia, J.: The effects of wide-area conditions on www server performance. In: ACM Sigmetrics Conference on Measurement and Modelling of Computer Systems, USA, pp. 16–20 (2001)
12. Eylen, T., Bazlamaçı, C.F.: One-way active delay measurement with error bounds. IEEE Trans. Instrum. Meas. **64**, 3476–3489 (2015)
13. Levesque, M., Tipper, D.: Improving the PTP synchronization accuracy under asymmetric delay conditions. In: IEEE International Symposium on Precision Clock Synchronization for Measurement, Control, and Communication, Germany, pp. 11–16 (2015)
14. Venkataramani, A., Kokku, R., Dahlin, M.: TCP nice: a mechanism for background transfers. In: 5th Symposium on Operating Systems Design and Implementation, Boston, pp. 329–343 (2002)
15. Ren, Y., Zhao, Y., Liu, P., Dou, K., Li, J.: A survey on TCP incast in data center networks. Int. J. Commun. Syst. **27**, 1160–1172 (2014)

16. Cisco Systems: NetFlow Configuration Guide, Cisco IOS Release 12.4, San Jose, USA (2011)
17. Cisco Systems: Cisco Nexus 9000 Series NX-OS System Management Configuration Guide, Release 7.x, San Jose, USA (2015)
18. Floyd, S.: Metrics for the evaluation of congestion control mechanisms, RFC5166 (2008)
19. Mathis, M.: A framework for defining empirical bulk transfer capacity metrics, RFC3148 (2001)
20. Isobe, T., Ito, D., Akashi, D., Tsutsumi, S.: RADIC-TCP: high-speed protocol applied for virtual private WAN. In: 18th International Conference on Telecommunications, Cyprus, pp. 505–510 (2011)
21. Kang, S., Prodanoff, Z., Potti, P.: Performance model of a campus wireless LAN. In: Sobh, T., Elleithy, K., Mahmood, A., Karim, M.A. (eds.) Novel Algorithms and Techniques in Telecommunications, Automation and Industrial Electronics. Springer, Dordrecht (2008)
22. Lee, J., Payandeh, S., Trajkovic, L.: Performance evaluation of transport protocols for internet-based teleoperation systems. In: OPNETWORK 2010, Washington DC, USA (2010)
23. Narula, R.: Performance analysis and evaluation of hybrid network using different integrated routing protocols. Int. J. Comput. Technol. 11, 3090–3100 (2013)
24. Garsva, E., Paulauskas, N., Grazulevicius, G., Gulbinovic, L.: Packet inter-arrival time distribution in academic computer network. Elektron. IR Elektrotechnika 20, 87–90 (2014)