Idle Time Estimation for Bandwidth-Efficient Synchronization in Replicated Distributed File System

Fidan Kaya Gülağız, Süleyman Eken, Adnan Kavak, and Ahmet Sayar
Department of Computer Engineering, Kocaeli University, Turkey

Abstract: *Synchronization is a promising approach to solve the consistency problems in replicated distributed file systems. The synchronization can be repeated periodically, with fixed time interval or a time interval which can be adjusted adaptively. In this paper, we propose a policy-based performance efficient distributed file synchronization approach, in which synchronization processes occur in varying time intervals and adjusted adaptively. The study is based on tracing network idle times by means of measuring and clustering Round Trip Time (RTT) values. K-means clustering is used to cluster RTT values as idle, normal, and busy. To estimate the most suitable synchronization time intervals, the measured RTT values are included into these classes with an algorithm similar to Transmission Control Protocol (TCP) Additive-Increase/Multiplicative-Decrease (AIMD) feedback control. The efficiency and feasibility of the proposed technique is examined on a distributed file synchronization application within the scope of Fatih project, which is one of the most important educational projects in Turkey.*

Keywords: *Idle time detection algorithm, cloud traffic, round trip time, K-means clustering, distributed file synchronization, policy-based synchronization.*

Received October 4, 2015; accepted January 3, 2016

1. Introduction

With the development of the Internet technologies, demand for services is growing and size of data is increasing exponentially each day. Whereas, limitations such as network traffic cause problems like inefficient communication and high latency in data transfer between user nodes and cloud services. The limited network bandwidth and increase in data size are two major parameters negatively affecting performance in distributed system applications. To overcome these problems, or at least their effects, we propose a client-side proxy server to reduce network traffic between the users and remote cloud services [1]. In the proposed system; if a user is not in usage, or coverage, area of a proxy server, it communicates with the cloud server directly. If a user is in the coverage area of a proxy server, it communicates with the cloud services over a proxy server. In case of using proxy server for cloud services, there are three different kinds of locations keeping the same data. Those locations are end users' devices (tablets), proxy server and cloud server [7, 9, 10]. Since the same data (i.e., files) is replicated in three different locations, the system must guarantee their consistencies. However, to keep the replicated data consistent, an advanced distributed synchronization mechanism needs to be employed. The proposed system is developed in Fatih project ecosystem as shown in Figure 1. The data to be synchronized are educational files in various types such as word, power point, and text etc., used in school domain.

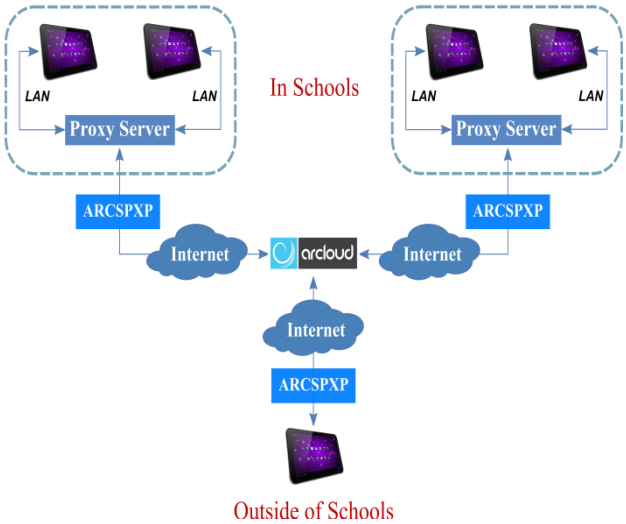

Figure 1. Proposed system architecture with school level proxy in Fatih Project.

In the current architecture, no matter where the tablets (end users) are, i.e., at school or at home, they first need to communicate with the cloud server. This is needed for the administration, authorization and authentication purposes. Proxy servers installed at schools enable the overall system to become manageable and scalable. They also enable overall network traffic to be decreased. This is the similar logic used in Content Delivery Network (CDN) in which data

is replicated in the intermediary servers geographically closer to the end users. The intermediary server in this study is a proxy server deployed in a corresponding school's Local Area Network (LAN). However, such architecture requires effective synchronization time detection algorithm to achieve the following:

- Data synchronization between the proxy server and a tablet.
- Update educational data in the proxy server, which is already modified on a tablet.
- Synchronization and back up of data from tablet to the cloud server.
- Synchronization of educational data stored in the proxy server to the cloud server.

The synchronization process needs to be determined with respect to application needs under the consideration of limited network bandwidth and big data sizes. For triggering a synchronization process at certain time intervals, a policy-based approach is proposed considering network idle time. We apply some data mining techniques such as K-means clustering algorithm to cluster Round Trip Time (RTT) record values as idle (class 1), normal (class 2), and busy (class 3). Then measured RTT values are included into these classes with an algorithm similar to Transmission Control Protocol (TCP) Additive-Increase/Multiplicative-Decrease (AIMD) [24] feedback control to estimate the most suitable synchronization time intervals. Experimental results show that proposed technique works very well for distributed data and computing systems requiring transportation of huge datasets on the conventional network with limited bandwidth.

The rest of the paper is organized as follows. In section II, related works are given. The third section explains idle time estimation technique based on measuring RTT times and applying K-means and AIMD algorithms. Section IV presents experimental results. Conclusion and some future enhancements are given at the last section.

2. Related Works

Estimation of available bandwidth by using RTT has been studied by some researchers. Imai *et al.* [15] proposed a new available bandwidth estimation method based on frequency of minimum RTT of probe packets in multi hop links. They think that these probe packets give heavy overhead loads on networks. So, their proposed method requires fewer number of probe packets and less estimation time and also shows better performance than other estimation techniques. Same researchers test error analysis of their method in another work [15, 16]. They clarify that the overall errors consist of the sampling errors and the errors caused by the probe packets size.

There are also many data mining methods to estimate network RTT's. Borzemski [5] modelled RTT and throughput values of a network with data mining techniques. He estimated the network traffic of servers via classification and clustering methods, in which RTT measurements are taken hourly and daily basis. Li *et al*. [19] applied adaptive filtering method on past RTT values to improve RTT estimation. Some researchers used machine learning techniques to increase the accuracy of RTT estimations. Improvements have been made in consideration of differences between actual and estimated RTT measurements [22]. The studies that used different methods and data mining techniques have shown that RTT provides significant clues in estimation of network load. Besides, there are various tools and software to characterize networks. Used metrics by these tools can be given as end-to-end available bandwidth, one-way and two-way delay, throughput and etc. All aforementioned tools and software require number of probe packets to estimate affecting the performance of the system.

Ijtihadie *et al*. [14] proposed a method for sharing e-Learning content between distributed learning management systems by means of unidirectional (master to slave) dynamic content synchronization. In this model, master is allowed to make changes while the content is not at the slave side. This approach does not consider the available synchronization time. In our previous work [8], we evaluated different classification methods considering RTT values for data synchronization between proxy and cloud servers in Fatih project. The most well-known classification approaches such as C4.5 algorithm and Bayes classifier are applied on previously obtained RTT values via jNetPcap. Experimental results show that to use at least two approaches in terms of system efficiency is more precise. However, it is not known when the next synchronization process is going to be performed.

In one of our previous works [10]; we have studied estimation of frequency of synchronization times. We took into account the usage statistics of the communication nodes' local resources. By the local resources we meant Central Processing Unit (CPU) and memory. For this purpose, we employed min-max normalization and multiple linear regressions. In the first step, we utilized jNetPcap, open-source java library, to obtain incoming and outgoing packets. These packets and other system criteria were stored in a text file as a historical archive of all network traffic for further analysis. In the second step, in order to minimize effect of too large or too small data in the average, we used min-max normalization. In the last step, we investigated impact of CPU and memory usage on packet size, as well as measured and estimated numbers of packages were compared. Estimations are performed with a regression based mathematical model. Although proposed technique gives effective results, it is time consuming and sometimes system information

can be affected from other parameters.

Also, there is need to explain proxy servers in detail. In computer networks, a proxy server is a server acting as an intermediary machine for requests from users seeking resources (e.g., file, web page, connection, and etc.,) from the specified servers. Proxy servers have a variety of potential purposes [13, 20]:

- To hide the IP address of the user computer for security.
- To speed up Internet surfing.
- To block undesired sites-black list, permit only authorized sites.
- To bypass security restrictions and filters.

As seen above, they have two main purposes: improvement performance and filtering requests. In our earlier work, we used proxy servers to decrease network traffic and increase the efficiency in data transfers between the end users and cloud services. In this paper, we use two-way delay metric RTT to obtain idle time (network availability) of a proxy server for data synchronization between cloud and proxy servers. Our approach is applicable for effective data management.

3. Tracking Bandwidth Availability Using RTT and K-Means Clustering

The components and architecture of the proposed system consist of five stages as shown in Figure 2.

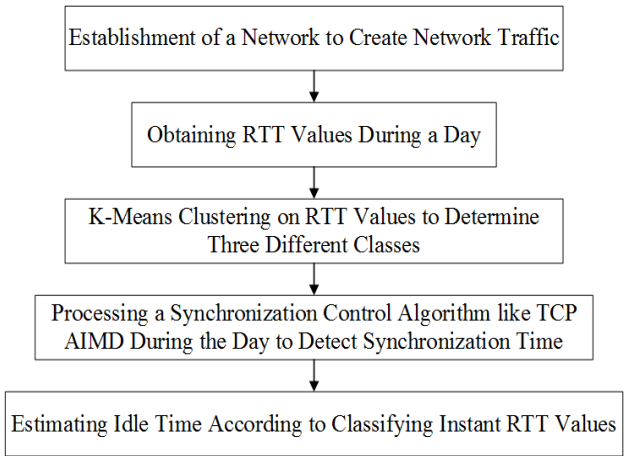

Figure 2. Main steps for estimation of server idle time.

These are:

1. Establishment of a network to create network traffic.
2. Obtaining of RTT values during the day.
3. Implementation of K-means algorithm on recorded RTT values for clustering.
4. Performing a synchronization control algorithm like AIMD to detect appropriate synchronization times.
5. Estimating idle time via classifying measured RTT values. Each stage is detailed in the following subsections.

3.1. Network Traffic Model

While network traffic is modelled, the density of the network is usually taken into account. In our scenario, network is considered as busy with approximately four thousand tablet users. These users are in communication with cloud server via wireless during busy period. In the design phase of the model, arrival time of the user's packet are taken into account.

The most common network traffic models considering the arrival time of packets are Poisson distribution, Pareto distribution and Weibull distribution [1, 12]. Poisson distribution is the member of exponential distribution family but Pareto and Weibull distributions are only related to exponential distribution.

The first traffic model that used is Pareto distribution process. Most of the network traffic models use Pareto distribution for arrival time of packets. Pareto distribution has two parameters, α and β. α is called as location parameter and β is called as shape parameter. The probability distribution function of Pareto distribution is given by

$$f(t) = \beta \alpha^{\beta} t^{-\beta-1}, \ \alpha, \beta \geq 0, t \geq \alpha \quad (1)$$

In Equation 1, if $\beta \leq 2$, distribution has infinite variance, and if $\beta \leq 1$, distribution has finite mean. The second traffic model is Weibull distribution process. It has also two different parameters, β and α. α is called as shape parameter and β is called as scale parameter. Weibull distribution can model the fixed rate in period and ON/OFF period lengths when producing self-similar traffic by multiplexing ON/OFF sources [6]. Weibull distributed function is expressed as,

$$f(t) = \alpha \beta^{-\alpha} t^{\alpha-1} e^{-\left(\frac{t}{\beta}\right)^{\alpha}}, t > 0 \quad (2)$$

Different network traffic models are compared in the literature and Pareto distribution appears to have the largest performance in high speed networks under unexpected packets flow [1]. So TCP connection count and packet sizes that will be sent with each TCP connection can be determined using Pareto distribution. Pareto distribution is also a good model for busy networks, so we chose it for modelling our network. In order to obtain busy network, requested packets must be big in size. The frequency of packets being sent can be modelled with exponential distribution.

In Table 1, β parameter for Pareto distribution is determined as 1.1. The reason for this is to obtain an infinite distribution. If $\beta \leq 2$, distribution has infinite variance. Since we want to send at least one packet in a minute, the value of λ is determined as 16. λ stands for the frequency of sending packets from a user to a proxy server. A value is determined in the range between 250 and 1000, and represents the numbers of students

sending files to a proxy simultaneously at any given time [4]. Our system is designed for schools so in this scenario students are representing the users.

Table 1. Details of network traffic model.

Component	Model	Probability Density Function	Parameters
File Sizes	Bounded Pareto	$f(t) = \beta\alpha^{\beta} t^{-\beta-1}$	β=1.1, α=[1024-102400] KB
Request Sizes (number of concurrent requests)	Bounded Pareto	$f(t) = \beta\alpha^{\beta} t^{-\beta-1}$	β=1.0, α=250-1000
Frequency of sending packet	Exponential	$f(t) = \lambda e^{-\lambda t}$	λ = 16

The details of the network traffic model are given in Table 1. The model mentioned above is used by four users simultaneously. Users send various sizes of packets to the cloud server over wireless links. Three different scenarios are taken into consideration while creating traffic. The first scenario is created according to the cloud server busy status (with cloud server busy status; we are considering the status of the cloud server and the status of its links/network). The second scenario is created according to cloud server's normal status and the third scenario is created according to the cloud server's idle status. After having defined network traffic models, estimated and measured RTT values are defined in the following section. This will form the basis for the idle time algorithm presented in sub-section 3.4.

3.2. Round Trip Time

RTT is the length of time it takes for a signal to be sent plus the length of time it takes for an acknowledgment of that signal to be received. This time delay therefore consists of the propagation times between the two points of a signal [3]. The estimated RTT is calculated by the formula given in Equation 3. The estimated RTT value is weighted combinations of the average value of recorded RTT values and the last measured RTT value.

$$RTT = (\alpha\ Old\_RTT) + ((1-\alpha) New\_RTT\_Sample) \quad (3)$$

In Equation 3, α is a constant value changing between 0 and 1. Choosing a value close to 0 makes the weighted average to respond to changes in delay very quickly [3]. The value of α is usually set to 0,125. In this work, created model calculates the average RTT values during a day and records these RTT values in a text document. Every day, classes are created according to K-means algorithm. Detailed information regarding the classification process is given in the next section.

3.3. Clustering of RTT Values

K-means is one of the simplest unsupervised learning algorithms that solve the well-known clustering problem. K-means algorithm clusters available data into a number of k clusters according to the features without the knowledge of any class. Grouping is related to sum of square of difference between centred of the cluster and each object in the data set. After the classification process, each cluster is labelled.

K-means algorithm used in our study is shown in Figure 3. The number of clusters is set as 3. At first, cluster centres are determined. The centre of the first cluster is determined as the minimum value in RTT data set, the centre of the second cluster is determined as the maximum value in RTT and the centre of the third cluster is determined as the average value in RTT data set. According to these centre values, clusters are established.

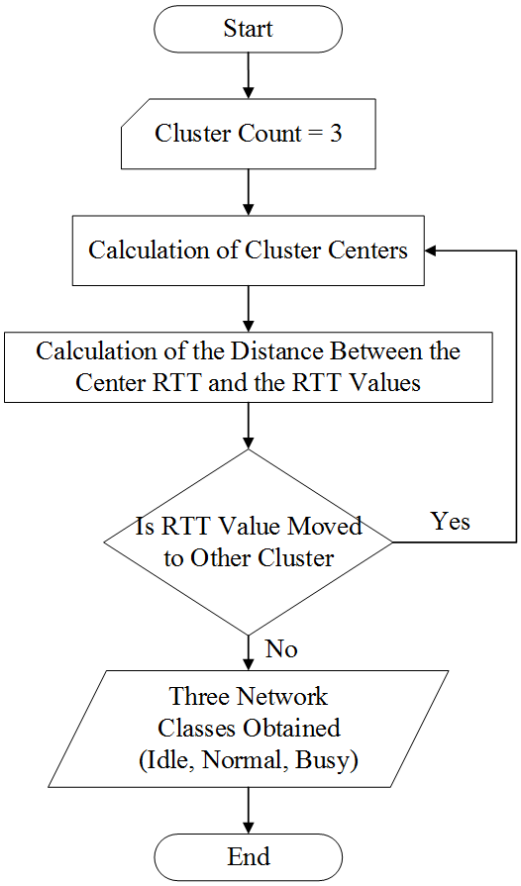

Figure 3. Flow chart representing RTT clustering.

After this stage, the centre points of the clusters and members of each cluster are determined using the algorithm illustrated in Figure 3. The sample clusters are given in Table 2 in section 4. The first cluster includes minimum RTT values and represented as not busy/idle, the second cluster includes average RTT values and represented as normal and the third cluster includes maximum RTT values and represented as busy. After having defined the clusters for RTT values, idle time algorithm built on this is explained in the following section.

The algorithm presented in this paper is based on and inspired from TCP AIMD algorithm. AIMD is a part of congestion control strategy used by TCP [2, 23]. AIMD

algorithm is a feedback control algorithm, best known for its use in TCP congestion avoidance. AIMD combines linear growth of the congestion window with an exponential reduction when congestion takes place. This approach is based on increasing the transmission rate (window size), by probing for usable bandwidth, until loss occurs. The policy of additive increase may, for instance, increase the congestion window by a fixed amount at every RTT. When congestion is detected, the transmitter decreases the transmission rate by a multiplicative factor; for example, cut the congestion window in half, after a loss. The result is a saw-tooth behaviour that represents the probe for bandwidth [21]. AIMD algorithm can be expressed as,

$$W(t+1) = \begin{cases} W(t) + \varepsilon & if\ congestion\ is\ not\ detected \\ W(t)\rho & if\ congestion\ is\ detected \end{cases} \quad (4)$$

where, $W(t)$ represents the sending rate during time slot t, ε represents the additive increase parameter under the condition $\varepsilon > 0$ and p is the multiplicative decrease factor under the condition $0 < p < 1$.

The proposed algorithm aim is to find an ideal time value for a synchronization process to start. A synchronization time should not be too short or too long. It is not feasible to perform synchronization algorithm while server is busy. If the server is busy, waiting time should be keeping longer, and if it is idle, waiting time should be keeping shorter for synchronization process. To the best of our knowledge, the most crowded school has almost 4000 students (users) in Turkey. Based on this information, for the worst case scenario, we consider each school has 4000 students to define the parameters needed for our study. According to this scenario, 4000 students (users) uploaded 5 MB files at the same time. In such situation, files are uploaded to the cloud server in 30 minutes. Therefore the upper limit of waiting time is set as 30 minutes, and therefore, waiting time is initially set to 30 minutes (see also Table 3). If a measured RTT value falls in the range of threshold values of idle cluster (class 1), then waiting time is reduced exponentially. If measured RTT value is in the range of the threshold values of normal cluster (class 2), then the waiting time is decreased linearly. If measured RTT value falls in the range of threshold values of busy cluster (class 3), then the waiting time is doubled. However, the waiting time should be between upper and lower limit values. The upper limit of a waiting time is determined as 30 minutes and the lower limit of waiting time is determined as 5 minutes. The steps of the idle time algorithm are shown in Figure 4. Exponential and linear growths are given in Equations 5 and 6 for idle RTT and normal RTT clusters, respectively.

$$T = T_0 2^i, \quad i = 0,1,2,\ldots, \quad T_0 = 5\ minutes \quad (5)$$

$$T = T_0\,(1+i), \quad i = 0,1,2,\ldots, \quad T_0 = 5\ minutes \quad (6)$$

In Equation 5, i is the number of consecutive RTT values falling into the idle class. In Equation 6, i is the number of consecutive RTT values falling into the normal class.

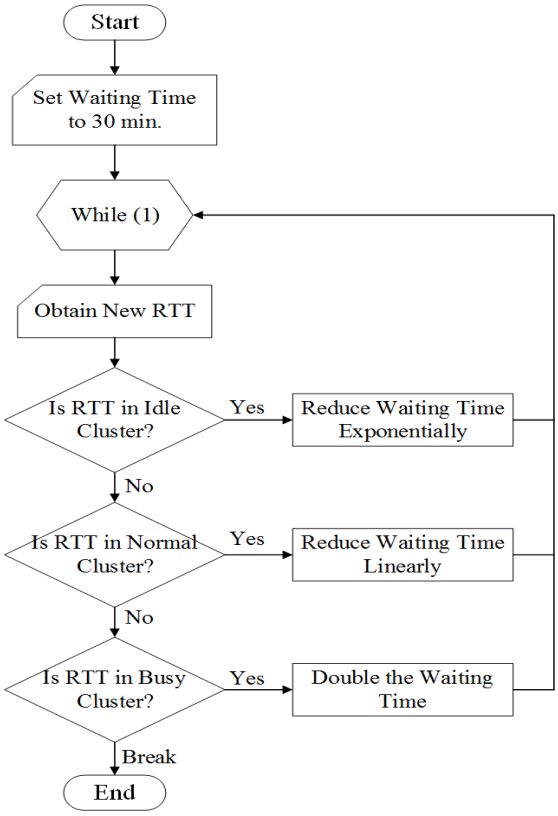

Figure 4. Idle time detection algorithm.

4. Experimental Results

To evaluate the efficiency of the proposed idle time estimation algorithm, the test setup is designed as illustrated in Figure 5. Experimental tests are realized on the server with Intel(R) Core(TM) i5-3317U CPU @ 1.70 GHz, 8 GB RAM 350 GB hard disk and Windows 7 Ultimate operating system. Network traffic is created with four users which are called as "clients" in Figure 5. Jmeter tool is used for simulation of network traffic. RTT Client in Figure 5 is used to calculate measured RTT values. This client has same properties with proxy server.

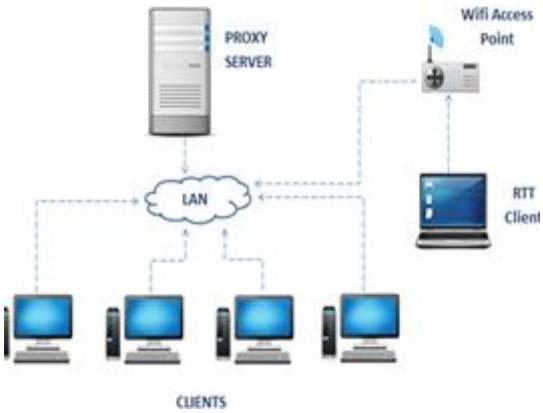

Figure 5. Establishment of test environment.

According to the proposed algorithm, first, RTT classes need to be created. To achieve this, measured RTT values are recorded in a text file for further analysis and they are clustered by using K-means algorithm (see Table 2). In Table 2, the first column shows RTT values obtained in one day and the second column shows class of RTT values, idle, normal and busy.

Table 2. RTT ranges of classes after k-means algorithm.

RTT values (ms)	Class
5.2 - 386.8	Idle (Class 1)
1204.0 - 1345.0	Normal (Class 2)
2330.9 - 3635.2	Busy (Class 3)

After clusters are obtained, measured RTT values during that day (e.g. we can say day is Tuesday) are included to appropriate clusters. Table 3 shows waiting times of classes according to measured RTT values for synchronization.

Table 3. Waiting time variation according to network availability.

Time	RTT values	Class	Waiting Time
02.04.2014 – 08:00	6.0	Idle	30 min
02.04.2014 – 08:15	54.8	Idle	25 min
02.04.2014 – 08:30	125.0	Idle	15 min
02.04.2014 – 08:45	1200.8	Normal	10 min
02.04.2014 – 09:00	2450.5	Busy	20 min
02.04.2014 – 09:15	11.0	Idle	15 min
02.04.2014 – 09:30	101.4	Idle	5 min
02.04.2014 – 09:45	388.5	Normal	5 min
…	…	…	…

The waiting times between any two successive synchronization processes are given in Table 3. These are calculated by the proposed technique using classification classes given in Table 2. The waiting time for first synchronization process is set to 30 minutes. The reason for this is explained in section 3.3. According to the first row in Table 3, the first measured RTT value showed that proxy server is idle, so the waiting time is reduced exponentially according to Equation 5. For the second synchronization, the parameter is reduced as multiple of five minutes and the waiting time is now 25 minutes. Measured RTT value is again calculated and RTT value shows that server is still in idle time. In this case, the waiting time is reduced 10 minutes exponentially according to Equation 5. So, the waiting time is reduced to 15 minutes. As a result of the third synchronization control, the proxy server is detected as normal class. In this case, the waiting time is reduced linearly according to Equation 6 and the waiting time is now 10 minutes. In the fourth synchronization control, server is identified as busy class and the waiting time is doubled, so the waiting time is now 20 minutes In short, waiting times are determined according to network density.

The relation between the number of users and the operation time is shown in Figure 6. The test environment can handle 1000 users simultaneously. The number of users depends on transmitted data structure. After 1000 users, response time is increasing very quickly. Despite the relation between numbers of users and response times, we performed synchronization between proxy server and cloud server and recorded throughputs. In our system throughput means total data/total sending time.

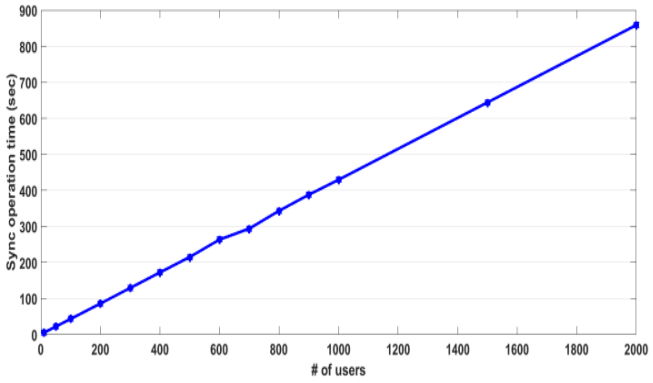

Figure 6. Relation between number of user and operation time.

The change in the value of throughput is shown in Figure 7. System can answer 1000 users simultaneously. After starting synchronization process, number of users decreases to 600-700 users. While synchronization process is performed and approximately 700 tablet users are requested to the server simultaneously, the RTT value changes between 2000-3000 ms. this range is classified as busy which is shown in Table 2. In this case, the synchronization process will not start and extra synchronization overhead will not occur. Experimental results show that the algorithm is working properly and the proposed system answers to the users' requests efficiently.

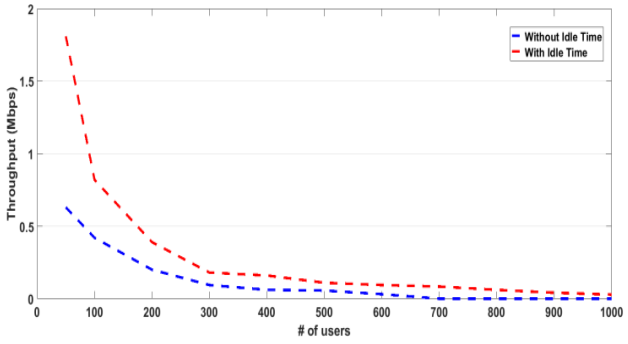

Figure 7. Relation between number of user and throughput.

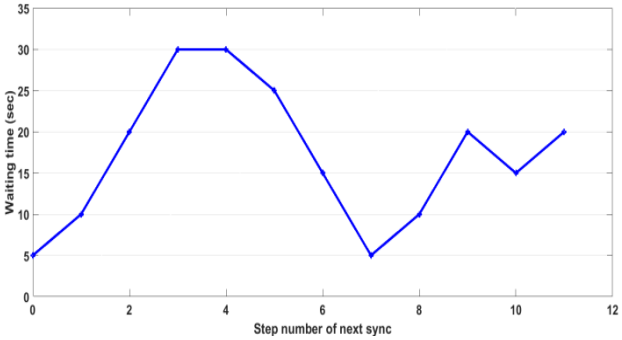

Figure 8. Change of waiting time.

When the system is busy, it kills users' requests and the system should wait to start synchronization process. Relationship between RTT values and waiting time is shown in Figure 8. If the system is busy, waiting time increases. Thus, system overhead that caused by the synchronization process is prevented.

Table 4. Comparison of total sync control count of CouchDB and our proposed idle time algorithm .

Waiting Time (sec)	Network State Class	CouchDB Total Sync. Control Count	Idle Algorithm Sync. Control Count
5	Busy	5	1
10	Busy	15	2
20	Busy	35	3
30	Busy	65	4
30	Busy	95	5
25	Idle	120	6
15	Idle	135	7
5	Normal	140	8
10	Busy	150	9
20	Busy	170	10
15	Normal	185	11
20	Busy	205	12

In Table 4, every synchronization control represented as step number. This figure is a graphical representation of waiting time between synchronization controls. Total waiting time in this figure is about 205 minute and during this waiting period 11 synchronization controls (11 steps) is realized. As seen, every control has different waiting period. If the system works in default state of CouchDB, the synchronization control would be made every one minute. In this case total number of synchronization control would be 205. The count of synchronization reduced from 205 to 11 with developed system. Synchronization count and unnecessarily controls are reduced and the performance of the system is improved.

5. Conclusions and Future Work

The proposed technique is not application specific; it can be used to measure the degree of availability of a network for a communication, or synchronization, to start. RTT values vary depending on connection types and network bandwidth. The degree of network availability is an important factor for the realization of efficient data transfer between distributed nodes. This becomes more important when the data to be transferred is big in sizes such as educational files (These system uses different file types such as text, multimedia, pdf, etc.,). The work presented in this paper proposed an algorithm which is measuring network availability by means of measured and recorded RTT values. According to the proposed algorithm, network specific RTT clusters can be created by setting different threshold values. Synchronization periods continuously updated according to the predefined RTT clusters and measured RTT values. So, the synchronization periods get longer in the case of heavily loaded networks, and get shorter in the case of lightly loaded networks. As a result, the data contained in both the proxy server and the cloud server is kept up to date all the time.

The main contribution of this paper is to find suitable synchronization times between distributed nodes such as student (users) tablets, school level proxy servers, and cloud servers by using K-means clustering technique and AIMD feedback control algorithm. The proposed approach concerns distribution of content among nodes with limited network infrastructure in terms of bandwidth and availability. In addition, it provides both decreased network traffic and increased efficiency of data transfers.

In the future, we plan to enhance the system by using some other network metrics and other classification techniques such as neural network back propagation algorithm, in order to detect most efficient synchronization times and periodicities.

Acknowledgment

This work is supported by the TUBİTAK under grant EEEAG 113E033 within 1003 Fatih Project Call.

References

[1] Aduba C. and Sadiku M., "Simulation and Analysis of Different Traffic Models for ATM Networks," *in Proceedings of the South East Conference*, Columbia, pp. 73-75, 2002.

[2] Al-Khatib W. and Gunavathi K., "A Novel Mechanism to Improve Performance of TCP Protocol over Asymmetric Networks," *The International Arab Journal of Information Technology*, vol. 5, no. 1, pp. 66-74, 2008.

[3] Almes G., Kalidindi S., and Zekauskas M., "RFC 2681: A Round-Trip Delay Metric for Ippm," *Internet Society*, New York, pp. 1-20, 1999.

[4] Barford P. and Crovella M., "Generating Representative Web Workloads for Network and Server Performance Evaluation," *in Proceedings of the International Conference on Measurement and Modeling of Computer Systems*, New York, pp.151-160, 1998.

[5] Borzemski L., "Data Mining in the Analysis of Internet Performance as Perceived by End-Users," *in Proceedings of the 18th International Conference on Systems Engineering*, Las Vegas, pp. 34-39, 2005.

[6] Crovella M. and Bestavros A., "Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes," *IEEE/ACM Transactions on Networking*, vol. 5, no. 6, pp. 835-846, 1997.

[7] Eken S., Kaya F., Ilhan Z., Sayar A., Kavak A., Kocasarac U., and Sahin S., "Analyzing Distributed File Synchronization Techniques for

Educational Data," *in Proceedings of the 10th IEEE International Conference on Electronics and Computation*, Ankara, pp. 318-321, 2013.

[8] Eken S., Kaya F., Gungor A., Sayar A., Kavak A., and Şahin S., "Bulut-Vekil Sunucu ve Tabletler Arasında Veri Senkronizasyonu için RTT Kullanarak Farklı Kümeleme Metotlarının Değerlendirilmesi," *in Proceedings of the Elektrik-Elektronik, Bilgisayar ve Biyomedikal Mühendisliği Sempozyumu*, Bursa, pp. 670-675, 2014.

[9] Eken S., Kaya F., Ilhan Z., Sayar A., Kavak A., Kocasarac U., and Sahin S., "Analyzing Distributed File Synchronization Techniques for Educational Data," *in Proceedings of the 10th IEEE International Conference on Electronics and Computation*, Ankara, pp. 318-321, 2013.

[10] Eken S., Kaya F., Sayar A., and Kavak A., "A Method for Localization of Computational Node and Proxy Server in Educational Data Synchronization," *Wireless Internet*, Lisbon, pp. 180-190, 2015.

[11] Eken S., Kaya F., Sayar A., and Kavak A., "Tracking a Single Node's Availability for Communication by Means of Observing Local System Resources," *in Proceedings of IEEE International Symposium on Innovations in Intelligent Systems and Applications*, Alberobello, pp. 41-45, 2014.

[12] Frost V. and Floyd S., "Traffic Modeling for Telecommunication Networks," *IEEE Communication Magazine*, vol. 32, no. 3, pp. 70-81, 1994.

[13] Howard R. and Jansen B., "A Proxy Server Experiment: an Indication of the Changing Nature of the Web," *in Proceedings of the 7th International Conference on Computer Communications and Networks*, Lafayette, pp. 646-649, 1998.

[14] Ijtihadie R., Hidayanto B., Affandi A., Chisaki Y., and Us-agawa T., "Dynamic Content Synchronization Between Learning Management Systems over Limited Bandwidth Network," *Human-Centric Computing and Information Sciences*, vol. 17, no. 2, pp. 1-17, 2012.

[15] Imai M., Ozawa T., Sugizaki Y., and Asatani K., "Error Analysis of an Estimation Method Using RTT for Available Bandwidth of a Bottleneck Link," *in Proceedings of the 15th Asia-Pacific Network Operations and Management Symposium*, Hiroshima, pp. 1-3, 2013.

[16] Imai M., Sugizaki Y., and Asatani K., "A New Available Bandwidth Estimation Method Using RTT for a Bottleneck Link," *IEICE Transactions*, vol. 97, no. 4, pp. 712-720, 2014.

[17] Imai M., Sugizaki Y., and Asatani K., "A New Estimation Method Using RTT for Available Bandwidth of a Bottleneck Link," *in Proceedings of the International Conference on Information Networking*, Bangkok, pp. 529-534, 2013.

[18] Kaya F., Eken S., Ilhan Z., Kavak A., Sayar A., Kocasarac U., and Sahin S., "A Comparative Study of Signaling Protocols for Data Management and Synchronization in Fatih Project with School Level Cloud Proxy Server Deployment," *in Proceedings of the IEEE 3rd Symposium on Network Cloud Computing and Applications*, Rome, pp. 133-136, 2014.

[19] Li G., Zhao N., and Liu C., "Round Trip Time Estimation Based on Adaptive Filtering," *in Proceedings of the 1st International Conference on Information Science and Engineering*, Nanjing, pp. 1842-1846, 2009.

[20] Ma W. and Du D., "Reducing Bandwidth Requirement for Delivering Video Over Wide Area Networks with Proxy Server," *IEEE Transactions on Multimedia*, vol. 4, no. 4, pp. 539-550, 2003.

[21] Mehdi D. and Ramasamy K., *Network Routing: Algorithms, Protocols and Architectures*, Elsevier, 2007.

[22] Nunes B., Veenstra K., Ballenthin W., Lukin S., and Abraczka K., "A Machine Learning Approach to End-to-End RTT Estimation and Its Application to TCP," *in Proceedings of the 20th International Computer Communications and Networks*, Maui, pp. 1-6, 2011.

[23] Sammaneh H., Al-Karaki J., and Bataineh S., "An End-to-End Support for Short-Lived TCP Flows in Heterogeneous Wired-cum-Wireless Networks: An Analytical Study," *The International Arab Journal of Information Technology*, vol. 8, no. 2, pp. 212-220, 2011.

[24] Yang Y. and Lam S., "General AIMD Congestion Control," *in Proceedings of the IEEE International Conference on Network Protocols*, Osaka, pp. 187-198, 2000.

Fidan Kaya Gülağız has received her BEng. in Computer Engineering from Kocaeli University in 2010 and ME. in Computer Engineering from Kocaeli University in 2013. She is currently working towards Ph.D. degree in Computer Engineering from Kocaeli University, Turkey. Also, she is currently a Research Assistant of Computer Engineering Department at Kocaeli University in Turkey. Her main research interests include data synchronization, distributed file systems and data filtering methods.

Süleyman Eken has received his BEng. in Computer Engineering from Karadeniz Technical University in 2009 and ME. in Computer Engineering from Kocaeli University in 2012. He is currently working towards Ph.D. degree in Computer Engineering from Kocaeli University, Turkey. Also, he is currently a Research Assistant of Computer Engineering Department at Kocaeli University in Turkey. His current research interests include distributed file systems, data synchronization, satellite image processing, remote sensing, WEB-GIS applications and spatial databases.

Adnan Kavak received the B.S. degree from the Electrical and Electronics Engineering Department, Middle East Technical University, Ankara, Turkey, in 1992. He received the M.S. and Ph.D. degrees from the Electrical and Computer Engineering Department, The University of Texas at Austin, TX, USA, in 1996 and 2000, respectively. He worked as a Satellite Control Engineer with Turksat Satellite Control Center, Ankara, Turkey, from December 1992 to May 1994. He worked as a Senior Research Engineer at Wireless Systems Laboratory, Samsung Telecommunications America in Richardson, TX, USA, from January 2000 to July 2001. He then joined Kocaeli University, Turkey, in August 2001 and worked as an Assistant Professor there until May 2005. Currently, he serves as the Director of Wireless Communications and Information Systems (WINS) Research Center and as an Associate Professor with the Computer Engineering Department, Kocaeli University, Turkey. He works also as an Adjunct Faculty at the Department of Electrical and Electronics Engineering at Yeditepe University, Turkey. His current research interests include 3G and beyond wireless networks, software and cognitive radios, smart antenna and MIMO systems, resource allocation in wireless networks, and wireless sensor networks.

Ahmet Sayar received M.S. in Computer Science (2001) from Syracuse University and Ph.D. in Computer Science (2009) from Indiana University, USA. He has worked at Los Alamos National Laboratory (New Mexico, USA) and Community Grids Laboratory (Indiana, USA) as a researcher. Since 2010, he is a professor at Computer Engineering Department, Kocaeli University, Turkey. His current research interests include distributed systems, big data, data-Intensive computing, remote sensing, GIS, and spatial data-databases. He has co- authored 3 books and around 70 papers.