QoS Adaptation for Publish/Subscribe Middleware in Real-Time Dynamic Environments

Basem Almadani[1], Shadi Abudalfa[1], and Shuang-Hua Yang[2]
[1]Computer Engineering Department, King Fahd University of Petroleum and Minerals, KSA
[2]Computer Science Department, Loughborough University, UK

Abstract: *Automatic Quality of Service* (*QoS) adaptation is promising approaches in developing real-time systems built on publish/subscribe middleware, and its importance increased when developing huge real-time distributed systems in dynamic environments. In this paper we proposed a new approach by building a closed loop QoS adaptive control for adapting some QoS polices automatically in publish/subscribe middleware. We use some techniques of artificial intelligence to automate QoS adaptation and improve the performance of the real-time systems in dynamic environments. The proposed approach improves the performance of the real-time system by saving its computing capabilities and making it more stable. Experimental results shown in this paper illustrate the effectiveness of the proposed approach.*

Keywords: *Publish/Subscribe middleware, QoS, adaptation, DDS, real-time, dynamic environments, clustering, online k-means.*

Received April 4, 2014; accepted may 24, 2015

1. Introduction

Complex heterogeneous distributed systems that cover large geographic areas and consist of a large number of nodes are common in today's networked world [8]. The widespread use of the Internet and advanced networking equipment facilitates the building of increasingly large systems with relatively low costs. Large-scale distributed systems are being built between organizations, individuals, and also between organizations and individuals. Publish/subscribe has been grown as a scalable communication mode for large distributed systems where traditional modes have weaknesses.

Publish/subscribe is not a magic solution to be used in all distributed systems. Publish/subscribe systems have been supported for specific systems where event services transmit through a large geographic area, because the publish/subscribe communication model performs extremely well under high latency conditions compared to other alternatives.

Publish/subscribe paradigm is an interest-oriented communication model. Event notifications are published by the producer (publishers) and consumed by the receiver (subscribers). In fact publish/subscribe system is a large group of nodes, these nodes are divided into clients who represent the beneficiaries from the network, and mediators which are responsible about routing notifications from publishers to subscribers.

A publish/subscribe system decouples event producers from event consumers by driving an abstract event service through the communication nodes. The event service delivered published events from publishers to subscribers who have registered their interest in the given event. The decoupling of publishers from subscribers and using asynchronous messaging allows publish/subscribe systems to improve their scalability for an increasing number of nodes and geographic distances between them.

The Object Management Group (OMG) defines the Data Distribution Service (DDS) for real-time systems [12] as standard specifications for publish/ subscribe architecture and runtime capabilities. The DDS standard enables applications to exchange data in event-based distributed systems. DDS offers high-performance communication capabilities and is currently used for several distributed critical mission applications. DDS is decentralized and scalable middleware for asynchronous distribution of publish/subscribe data. DDS is different from traditional publish/subscribe middleware. It can evidently control the efficient use of network resources, which are critical for real-time applications. The proposed approach of our work depends on a publish/subscribe middleware based on the OMG's DDS standard.

The DDS middleware has a layer that depends on a rich set of Quality of Service (QoS) which includes Topics, Data Readers, and Data Writers policies. Behaviours of these entities are configured by using a set of 22 QoS policies. On the other hand, The DDS middleware also has a layer for facilitating transparent information management and object oriented language constructs. Publishers control collection of data writers and subscribers control collection of data readers.

QoS policy [3] has many parameters and each parameter includes a range of values. Configuration of

QoS policy associated with individual publishers or subscribers has many complications and lack of attention in setting some parameters may affect overall behaviour of system QoS. It is a boring task to configure QoS policy manually when implementing any system by using the DDS middleware. Many compatible QoS policies should be checked to make the system able to run. In this paper we propose a new approach for adapting the QoS policies automatically.

In the dynamic environments systems that based on publish/subscribe middleware; some values of specific parameters might change while the system is running. This variation will affect some QoS policies and may cause the whole system to stop. This trouble is very acute when using critical mission systems because maintaining these systems costs additional expenses and wastes time, and it may also raise level of risk to human life.

To encourage the need for autonomic QoS adaptation in publish/subscribe middleware, we describe the challenge in this research associated with Search And Rescue (SAR) operations in Pilgrims Tracking and Monitoring System (PTMS) during the Hajj season. There are many operations help to locate and extract risky pilgrims in a large area with a huge population density. This system uses Unmanned Aerial Vehicles (UAVs), existing infrastructure cameras, and Command and Control Center (CCC) as shown in Figure 1. The CCC receives and processes data stream from sensors and cameras, and transmits action to emergency equipment that can be send to areas where risky pilgrims are identified. In this scenario UAVs deliver infrared scans along with GPS coordinates and infrastructure cameras deliver video steam. These infrared scans and video stream are sent to the CCC for processing by intelligent applications to detect risky pilgrims. Once risky pilgrims are detected the application can illustrate a multi dimensional view with accurate position to start the rescue operations. This system requires a huge amount of resources for storing and manipulating the data stream. And this system depends on keeping the history of the tracing process for pilgrims which need a plenty of storage to achieve. Storing all of these data may conflict with the resources which are available while the system is running in the dynamic environments. And if the limit of resources exceeded for any reason the system would halt and some time may be exhausted for maintaining.

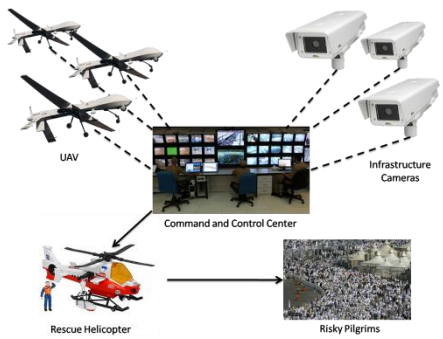

Figure 1. Pilgrims tracking and monitoring system.

The idea in our work is how to manipulate the challenges of checking compatible QoS policy configurations, by developing a new self-adaptive system publish/subscribe middleware in real-time dynamic environments. And our study will concentrate on using history QoS and resource limit QoS, and adapting the policy between them in the real-time systems.

The rest of the paper is organized as follows: section 2 describes review of literature and related studies. Section 3 illustrates our contribution by presenting our new mechanism for QoS adapting in publish/subscribe system. Section 4 previews some of experimental works to demonstrate the effectiveness of the proposed system. Section 5 illustrates results and analysis the experiment work. Finally, section 6 concludes the paper and presents suggestions for future work.

2. Literature Review

Hoffert *et al*. [5] developed modeling language to help developers for choosing valid sets of values when configuring QoS policies in publish/subscribe middleware. This language helps in ensuring that the QoS policy configurations are compatible and do not conflict with each other. Also it helps to transform QoS policy configuration design into the correct Publish/subscribe middleware specific implementation in automated manner.

Article [6] describes architecture of QoS enabled middleware and corresponding algorithms to support specified QoS in dynamic environments. This article has prototyped approach in the adaptive middleware and network transports platform that supports environment monitoring and provides timely autonomic adaptation of the middleware. The article explains how the architecture monitors environment changes that affect QoS, and after that determines in a timely manner which appropriate transport protocol changes are needed in response to environment changes. This approach integrates the use of multiple supervised machine learning techniques to increase accuracy, and autonomically adapts the network protocols used to support the desired QoS.

Some composite metrics are defined in [6] to estimate different QoS worries and. These metrics also analyze distinct modification mechanisms that are used for the QoS configurations of a DRE system in a dynamic environment. This research evaluated adaptation approaches for smart city ambient assisted living applications. It evaluated policy-based adaptation approaches, reinforcement learning, and supervised machine learning.

Traditional machine learning tools, such as decision trees and reinforcement learning, have been used to support autonomic adaptation in embedded and undistributed real-time systems [15]. These techniques

aren't used widely for Distributed Real-time and Embedded (DRE) systems.

Not all machine learning techniques are usable for DRE systems. Some techniques like reinforcement learning [2] seek the whole solution space to find an adequate solution without care of the elapsed time. Other techniques like decision trees have a time complexity that depends on data values and not known in advance. On the other hand, decision trees may use different length of branches, which make adaptations of the DRE systems unpredictable when finding the required adaptations.

Machine learning algorithms are described as either supervised or unsupervised. The difference is dragged from how the learner classifies data. In supervised algorithms [10], the classes are predetermined. These classes can be imagined of as a finite set previously determined. In other words, a certain segment of data will be labeled with these classifications. Unsupervised learners [4] are not provided with classifications. The main idea of unsupervised learning is to develop classification labels automatically.

The evaluation of Artificial Neural Networks (ANNs), which is a supervised machine learning technique, to treat time complexity worries of adaptive DRE systems is presented in [6]. This article integrates the ANN machine learning technique with the DDS middleware to guarantee accurate, timely, and predictable adaptation in dynamic environments. This system takes into account increasing data sending rate and number of data receivers while system is running. The results of the article concluded that ANNs treat the complexity of DRE systems by undertaking that adaptations are suitable for the dynamic environments.

Mabrouk *et al*. [11] have developed an algorithm for service selection by providing the appropriate ground for QoS awareness in dynamic service environments. This algorithm depended on k-means [1] algorithm, which is an unsupervised machine learning technique, in selecting the best set of services.

Machine learning techniques are applied in [6] to facilitate QoS configuration and change transport protocols dynamically to preserve specified QoS while systems is running. The approach used decision trees and ANN for classifying the protocol parameters to use in dynamic environments for critical mission.

Two supervised machine learning techniques are used in [6] to determine the appropriate techniques for two situations, environment configurations known a priori or unknown until runtime. This approach used ANNs and Support Vector Machines (SVMs) to reduce time complexity, latency, and development complexity. The results showed that ANNs are generally more accurate in adapting environments whose properties are known before operating the system, while SVMs are more accurate in adapting environments whose properties are known after running the system. The article concluded that both techniques can be used together with publish/subscribe middleware in DRE systems to control the development complexity, accuracy, and timeliness in dynamic environments.

Hoffert and Schmidt [7] describes how the authors are evaluating multiple QoS concerns based on variation in computing and network resources and configuring the publish/subscribe middleware automatically in cloud environments. The proposed approach adjusts network transports platform in alignment with the middleware to control problems of QoS configuration in DRE systems for cloud environments. The approach uses ANN tools to determine dynamically the suitable transport protocol for the publish/subscribe middleware after running the DRE system. ANN tools are trained by using data configurations to support the best QoS and predict the response times which are needed in DRE systems. The approach middleware used the Adaptive Network Transports (ANT) framework to choose the best transport protocol that concerns multiple QoS in compatible with availability of computing resources.

There are many challenges should be taken into account when using distributed systems worked in dynamic environments and based on publish/subscript middleware. SAR operations must adjust in a timely manner while the environment behaviour changes dynamically. In case of resources limits, if the amount of requested data is increased, then operations must be adjusted to preserve the service with minimum level. Of course, modifying QoS manually will be too slow and sensitive to error. The SAR system should control multiple QoS polices which interact together such as history and recourse limit. This demand will be more needed when a lot of organizations request video streams, infrared scans, and also the merge data of the SAR operations. On the other hand, the SAR systems should be designed with concentration on decreasing complexity in specifying QoS. Application developers need to support a various QoS polices to handle dynamic environments.

3. Proposed Solution

The goal of our work is to design a dynamic mechanism for adapting QoS behaviour in distributed systems worked in dynamic environments and based on real-time publish/subscript middleware for critical mission.

In our work, we designed publish/subscribe system that solves some of problems that might occur in a real-time system during the running time while the QoS behaviour is changed. The proposed system deals with monitoring the QoS behaviour while the system is running and checks inconsistency between specific QoS polices and then adapting the QoS behaviour to obtain the best performance for the whole system. The

main components of the proposed system are included in Figure 2. The proposed system consists of data writers for collecting a stream of data. The type of gathered data depends on the purpose of the system. In the PTMS system which is represented in the introduction section, the data writers are used to gather video streams and infrared scans. The second component of the proposed system is a data reader which used to read and manipulate the gathered data to take a suitable decision later by the subscriber that is connected to this data reader. We can say that data reader and the corresponding subscriber resemble the CCC in the PTMS system. The third component of the proposed architecture is a system monitor. The system monitor tracks the QoS behaviour and collects a specific statistical data based on the system performance. The last part of the proposed system is adaptive controller which receives the statistical data that is collected from the system monitor and classifies it for selecting the best situation and then sends the correct action to the data readers to change their behaviour by modifying specific settings of QoS.

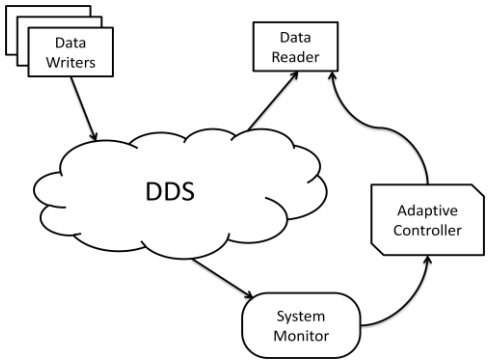

Figure 2. Proposed system architecture.

The proposed system based on the OMG's DDS standard [12]. For building the proposed system and testing its performance, we use Real-Time Innovations (RTI) Connext™ DDS solutions which provide real-time information commute between applications and systems [14]. RTI Connext™ DDS software supports a messaging solution for optimizing communication between intelligent machines. It is a powerful data-centric approach for addressing the contemporary challenges of big data in machine-to-machine communication.

DDS supports a set of QoS police which can be used to configure behaviour of large event-based distributed systems. There are many constraints in DDS should be taken into account when configuring the distributed systems to be compatible with QoS policies. If these constraints are not tuned then the data will not flow from data writers to data readers. If there are some QoS settings are incompatible, then the system will not behave as needed. It is not easy to check compatibility and inconsistent among all QoS policies, and it takes a lot of time to achieve manually. So, the need to develop a new approach for adapting QoS policies automatically is increasing every day.

Our approach deals with monitoring some QoS polices and modifies other specific QoS policy to increase the performance of running real-time systems in dynamic environments. Our system address three QoS polices: history, resource_limits, and time_based_filter. The history policy controls whether the service should deliver only the most recent value or do some jobs in between. Resource_limits policy specifies the maximum number of resources that the service can consume to meet the requested QoS. Time_based_filter policy controls the data reader to avoid receiving more than one value every minimum separation period, regardless of how fast the data changes occur.

The proposed system monitors the history and resource_limits QoS polices and modifies the time_based_filter QoS policy. Our system checks periodically the behaviour of history and resource_limits QoSs because the performance of the real- time system will be affected if these QoSs are overloaded. If the overload exceeds some limit the adaptive controller will urge the data reader to increase the value of time_based_filter QoS for saving the capacity of the system.

The publish/subscribe model of the proposed approach consists of two topics as shown in Figure 3. The App topic is used for collecting data such as video streams and infrared scans in PTMS system. The Monitor topic is used to gather statistical data for checking the QoS behaviour of the whole system. The proposed system uses many data writer for the App topic to enable the system to gather data from various recourses such as cameras and UAVs in PTMS system. There are also many data readers to enable many authorities to use online data and working together for applying the best service. In the other side of the system which is used for monitoring, we use only one data writer and one data reader to observe the system behaviour and gather the required data. Using single data writer and single data reader is enough to accomplish the required tasks and helps in not increasing traffic in the network.

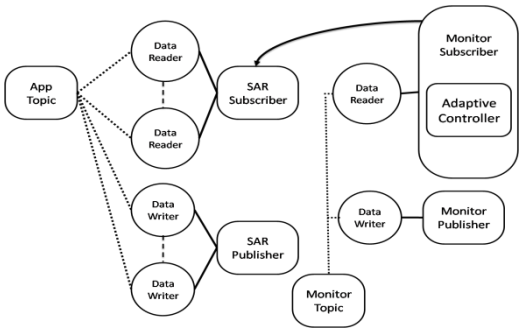

Figure 3. Publish/subscribe model of proposed system.

The monitor publisher observes the behaviour of data writers of SAR publisher and collects specific statistical data. The monitor subscriber has an adaptive controller unit to cluster the statistical data which is gathered by the monitor publisher. Depending on the results of the clustering, the monitor subscriber sends the suitable action to the SAR subscriber and change QoS behaviour of its data readers if necessary. Figure 4 shows a flowchart that illustrates the working mechanism of proposed approach.

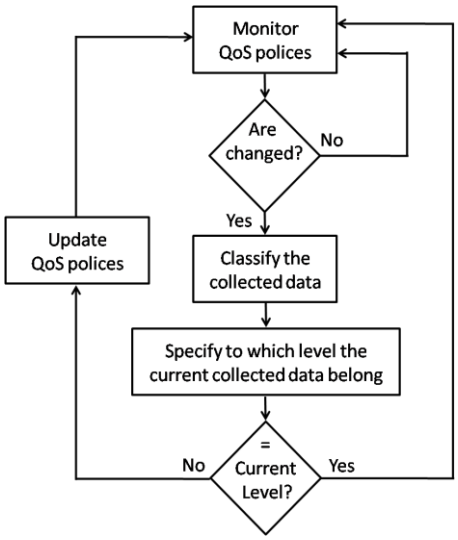

Figure 4. Working mechanism of proposed system.

Hoffert *et al*. [6] used supervised machine learning technique to classify the gathered data from the monitor publisher. In our work we use unsupervised machine learning technique instead of supervised machine learning technique to rid of the training step in the supervised machine learning technique. Using the unsupervised machine learning technique is more suitable in real-time environments, because it can be applied immediately on the online data that may change frequently. On the other side, we need to label some data firstly for training phase in supervised machine learning. So accuracy of the supervised machine learning technique is sensitive to nature of the trained data. Using supervised machine learning will increase also the time consumed in changing the QoS behaviour by trying to train the dynamic data many times to preserve the system accuracy.

We use online k-means algorithm in the adapter controller for clustering the statistical data which is collected from the monitor publisher. Online k-means provides a simple and efficient way to cluster a data set into a fixed number of clusters; in addition to that it is used for clustering non-stationary data which is suitable to cluster a dynamic data that is gathered in real-time systems [9].

We assume that transport among nodes is unreliable, and it causes a loss of packets and our approach will decrease this loss which is characterized by Packet Loss Rate (PLR) by preserving the resources. As illustrated in Figure 5, the Markov probability model of our approach is a first order chain with two states [13]: "Good", with a state dependent error rate equal to 1-K, and "Bad", with a state dependent error rate equal to 1-H. Typically, K is assumed equal to 1 so that the "Good" state implies that no losses are applied, while in our model we assume that 1-H is equal to PLR.

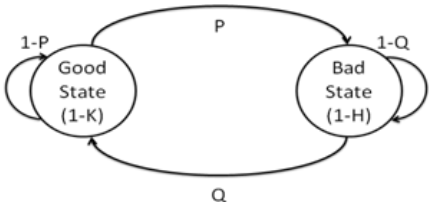

Figure 5. Markov probability model.

The proposed model is characterized by four transition probabilities: The probability P to pass from state "Good" to state "Bad"; and the probability 1-P to remain in state "Good"; the probability Q to pass from state "Bad" to state "Good"; the probability 1-Q to remain in state "Bad". It is possible to compute P and Q as follows:

$$P = \frac{PLR.Q}{1 - PLR} \quad (1)$$

$$Q = ABL^{-1} \quad (2)$$

Average Burst Length (ABL) is the mean number of consecutive lost packets. So if K=1, then no packets are lost and the model in the "Good" state, while if H=0, then all packets are lost and the model in the "Bad" state. It is easy to illustrate that the probability to lose a packet is equal to the mean PLR over time.

In addition to the above design of the proposed system, we will analyze it theoretically as illustrated in [16]. The probability that *i* events occurred during the interval between disconnection and restoration is:

$$P = \left(\frac{\lambda_{pub}}{\lambda_{pub} + \lambda_{recov}^{sub}} \right)^{i} \frac{\lambda_{recov}^{sub}}{\lambda_{pub} + \lambda_{recov}^{sub}} \quad (3)$$

The average number of events that subscribers miss is:

$$E(N) = \left(\frac{\lambda_{recov}^{sub}}{\lambda_{pub} + \lambda_{recov}^{sub}} \right)^{NL} \frac{\lambda_{pub}}{\lambda_{recov}^{sub}} \quad (4)$$

Where NL is the maximum number of events that the CCC can save in the PTMS system. λ_{pub} is the publication rate of publishers. λ_{recov}^{sub} is the recovery rate of subscription and links.

We improved the performance of the proposed system by clustering the statistical data that is collected by the monitor publisher into a number of levels. The Adaptive controller clusters the statistical data by using online k-means algorithm [9] to commensurate with

the nature of the real-time systems which deliver non-stationary data in the dynamic environments. The statistical data which is related to the values of history and resourse_limits QoSs of the middleware is clustered into a number of clusters whereas each cluster describes a specific load level in the system. The adaptive controller modifies the behaviour of real-time system by changing the value of time_based_filter QoS for all data readers of the SAR subscriber. The value of time_based_filter QoS is changed by depending on the value of current load level. Figure 6 illustrate the idea of clustering the statistical data into three clusters which are corresponding to three load levels. The x-axis (NSM_{DW}) denotes to number of data samples that are managed by data writers in a point of time. This value reflects the usage of resourse_limits in the data writers. The y-axis (NSK_{DW}) denotes to number of data samples that are kept by data writers in a point of time. This value reflects the usage of history in the data writers. The lowest level (K_{val}=1) denotes that the system used a little amount of resources (history and resourse_limits) and it is in the safe mood. The highest level (K_{val}=3) denotes that the system used a huge amount of resources and there is overload in the real-time system. The traditional real-time systems didn't manipulate the highest level (overload level) and causes reduction in the system performance. The proposed approach manipulates this dilemma dynamically by saving the system capabilities and improves efficiency of its work.

The proposed approach needs to determine the number of clusters K_{num} in advance before running the system. We can increase the value of K_{num} to increase the accuracy of the system performance. Increasing the value of K_{num} will cause a smooth changing in the QoS behaviour over the time of running the system, but in the same time it will also increase the number of switching operations to the value of time_based_filter QoS which may increase time complexity of the proposed system. So there is a tradeoff between the accuracy and time complexity of using the proposed system when selecting the value of K_{num}.

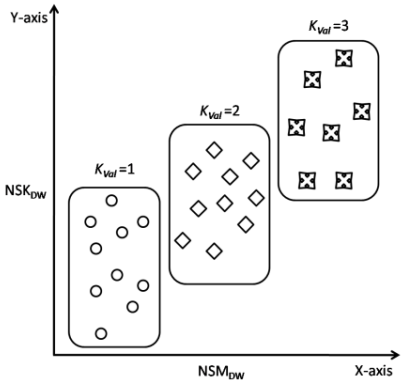

Figure 6. Example of selecting three load levels (K_{num}=3) by using online k-means clustering.

The adaptive controller of our proposed system uses a new algorithm to improve the performance of the whole system. This algorithm runs online while the system is running. There are three input parameters for the proposed algorithm that are determined in advanced before running the system: the first parameter is PR which determines the required performance ratio of the system. This parameter is compared periodically with the consumed percentage of resource limits and history in the data writers at the publisher side. The second parameter is K_{num} which determines number of load levels as illustrated in Figure 6. The value of this parameter equals to the number of clusters which is used by online k-means algorithm that is called for clustering the statistical data which is gathered by the monitor publisher. The last parameter is $MinSep_{Max}$ which determines the maximum value of minimum separation that has been accepted to set in the time_based_filter QoS in the data readers of SAR subscriber. The algorithm uses a statistical data which are gathered periodically by data writers of the monitor publisher. This data is gathered as data points of two values: the first value is NSK_{DW} which determines the number of samples that are kept in queue for all data writers of SAR publisher (y-axis of Figure 6). The second value is NSM_{DW} which determines the number of samples that are buffered by all data writers of SAR publisher (x-axis of Figure 6). The algorithm will change the value of minimum_separation in time_based_filter QoS for all data readers of SAR subscriber when behaviour of the real-time system is changed in a point of time. The value of minimum_separation will be modified depending on the value of point ($NSM_{DW,}$ NSK_{DW}) which is located in one of the clusters that generated by using online k-means. We present the pseudocode of the proposed algorithm as shown in Algorithm 1:

Algorithm 1: *Adaptive_Controller.*

Input: *PR*, $K_{num,}$ $MinSep_{Max}$

1. *IF Overload(PR) == TRUE then*
2. $MinSepData_{Reader} = SelectLoadLevel(NSM_{DW,} NSK_{DW}, K_{num,} MinSep_{Max})$
3. *IF* $MinSep_{Data\ Reader} < Deadline_{Data\ Reader}$
4. $Time\_Based\_Filter_{DataReader}=MinSep_{Data\ Reader}$
5. *END IF*
6. *END*

Function boolean Overload(PR)

1. *CurrPerf=0*
2. *For i←1 to NumDataWriters*
3. $CurrPerf = CurrPerf + (\frac{UsedSamples(i)}{maxNumSamples(i)} + \frac{NumSamplesKept(i)}{maxHistoryDepth(i)}) / 2$
4. *END FOR*
5. $TotalCurrPerf = CurrPerf/NUM_{DataWriters}$
6. *IF TotalCurrPerf > PR*
7. *return TRUE*
8. *ELSE*
9. *return FALSE*
10. *END*

Function float SelectLoadLevel ($NSM_{DW,}$ NSK_{DW}, K_{num}, $MinSep_{Max}$)

1. $K_{centroides}$= *OnlineKMeans*($NSM_{DW,}$ NSK_{DW}, K_{num})
2. K_{val} =*SelectCluster*($NSM_{DW,}$ NSK_{DW}, $K_{centroides}$)
3. *MinSep* =K_{val}×($MinSep_{Max}$/K_{num})
4. *return MinSep*

The pseudocode is called periodically by adaptive controller to check whether the system is overloaded or not by using Overload function. If the system becomes overloaded (exceeded PR ratio) in a specific time, then the SelectLoadLevel function will be called to calculate a new value of $MinSepData_{Reader}$. The value of minimum_separation in time_based_filter QoS will be set after that by the new value of $MinSepData_{Reader}$. SelectLoadLevel function will call OnlineKMeans function for clustering the accumulative collected data and return the cluster centers. The number of the cluster, which is contains the values of current NSM_{DW} (resourse_limits) and NSK_{DW} (history), is calculated by SelectCluster function. Finally, equation in line 3 of SelectLoadLevel function divides the $MinSep_{Max}$ value into K_{num} levels and calculates the new value of $MinSepData_{Reader}$.

4. Experimental Work

In this section, we demonstrated some experiments to prove the performance of the proposed approach. The experiments were carried out in COE distributed real-time systems lab at KFUPM by using specified hardware and software tools. The measurement tools and hardware platform specifications are described in Tables 1 and 2 respectively.

Table 1. Tools and programs.

Tool	Version	Purpose
RTI Analyzer	5.0.0	QoS monitoring
RTI Monitor	5.0.0	Measure CPU usage
MS Excel	2010	Plotting Graphs
Eclipse	2013	Compiling Java code
Java platform	jdk1.6.0_21	Implementing Java code

Table 2: Platform specifications.

	Publishers	Subscribers
CPU	Intel(R) Core (TM) 2 1.66 GHZ	Intel(R) Core (TM) 2 1.66 GHZ
Memory	1.00 GB	1.00 GB
OS	Windows 7	Windows 7
Network	LAN 100Mbps	LAN 100Mbps

The experiments framework is shown in Figure 7. The framework consists of many publishers and subscribers connected together through DDS middleware technology over LAN. We used many publishers and subscribers to illustrate the performance of the proposed system by increasing number of nodes (publisher/subscriber) when increasing the overload on the system and measure the results in a worse case. We didn't use computers with very high capabilities to make the scenario more realistic when building the real-time systems in dynamic environments and to increase the load in the experimental framework.

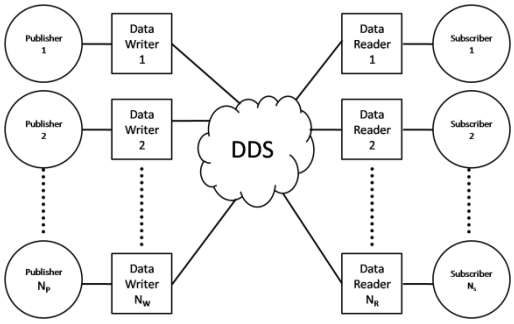

Figure 7. Experiment framework.

5. Results and Analysis

Time_based_filter QoS allows applications to specify minimum separation time between data samples. This policy controls network bandwidth, memory, and processing power for subscribers. If the subscribers have limited computing capabilities and connected over limited bandwidth networks, then we can use this policy to improve performance of the whole system. To clarify the effect of time_based_filter QoS on the computing capabilities we applied an experiment to measure CPU utilization while changing value of minimum separation. The experiment consist of three publishers (N_P=3) and five subscribers (N_S=5). To increase the load in the system we set deadline period to infinite and the depth of history equals to 100 for all subscribers and publishers. We set also durability QoS to transient_local to be compatible with history policy. It shouldn't be volatile in critical real-time systems such as rescue system. The results of the experiment are shown in Figure 8.

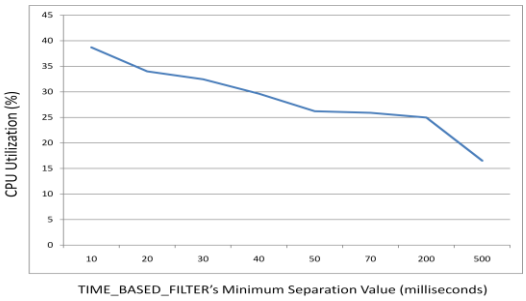

Figure 8. CPU Utilization.

The CPU utilization, which indicates to the total CPU usage by all subscribers, is decreased as expected while the value of minimum separation value of time_based_filter increased for all subscribers. This implies that we can save the subscriber capabilities by decreasing the value of minimum separation for the time_based_filter QoS. We noted also that the percentage of reduction in CPU utilization is about 25% while the modification in the minimum separation value did not exceed 0.5 sec. This means that we can save valuable amount of CPU utilization by decreasing little amount from the value of minimum separation. So we can conclude that it is beneficial to change the value of time_based_filter for saving a lot of

computing capabilities at data readers when the real-time system running in the dynamic environments.

We demonstrated the efficiency of modifying time_based_filter QoS when running the real-time system by applying another experiment. This experiment measured the changing in percentage of filtered bytes at data readers while changing the minimum separation value of time_based_filter QoS. The percentage of filtered bytes is calculated by measuring the number of filtered bytes divided by the number of received bytes in the data readers. The results of this experiment are shown in Figure 9. We note that the percentage of filtered bytes changed slightly (from 0.246 to 0.366) in a long period (between 1 and 5000 msec). This result proves efficiency of our approach in changing minimum separation value of time_based_filter QoS when running the real-time system. When applying the proposed approach we can save significant computing capabilities in the subscriber and the quality of service will be decreased by a negligible amount. We can also conclude that the proposed approach will decrease overload on the whole system by saving resources of the middleware and decreasing the number of events which are executed to read a lot of data samples.

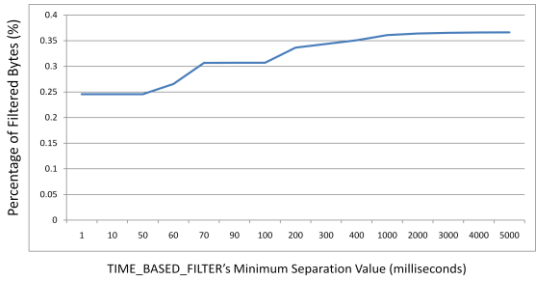

Figure 9. Percentage of filtered bytes.

We concluded that modifying the time_base_filter automatically will be more effective in real-time system that runs in dynamic environments. The proposed approach will save resources of data writers and computing capabilities of data readers. The proposed system is more stable and it is able to bear more burden than traditional system. The traditional system will be halted and overloaded when using a huge number of publishers and subscribers in dynamic environment.

6. Conclusions and Future Work

The necessity of developing a new approach for adapting QoS policies automatically is increasing every day. Its importance increased after developing huge real-time distributed systems worked in dynamic environments. A new approach has been developed in this paper for adapting a specific QoS in publish/subscribe middleware at real-time systems which works in dynamic environment. The proposed approach improved performance of the critical mission systems and made them more stable in the dynamic environments. Some data mining methods are used to develop the proposed system and improve its efficiency. Some experiments are demonstrated in this paper to prove the effectiveness of the proposed system in the real-time dynamic environments.

We can extend this research by developing a new algorithm for determining automatically all parameters which are used by the proposed approach to develop a fully automated system for QoS adaptation. We suggest also developing a modified version of online k-means algorithm to improve the performance of proposed system by decreasing the time complexity.

Acknowledgment

The authors gratefully acknowledge staff of Distributed Real-Time Systems Lab in computer engineering department at KFUPM, with special thanks to Mr. Anas AI-Roubaiey for their supports. The authors are grateful for useful advice and suggestions from Mr. Richard Williamson in RTI community. Moreover, authors thank RTI Company for their support with all tools that have been used in this work.

References

[1] Al-Zoubi M., Hudaib A., Huneiti A., and Hammo B., "New Efficient Strategy to Accelerate K-Means Clustering Algorithm," *American Journal of Applied Sciences*, vol. 5, no. 9, pp. 1247-1250, 2008.

[2] Bu X., Rao J., and Xu C., "A Reinforcement Learning Approach to Online Web Systems Auto-Configuration," *in Proceeding of 29th IEEE International Conference on Distributed Computing Systems*, Washington, pp. 2-11, 2009.

[3] Object Management Group, *Data Distribution Service for Real-time Systems*, Object Management Group, 2007.

[4] Gan G., Ma Ch., and Wu J., *Data Clustering: Theory, Algorithms, and Applications*, Society for Industrial and Applied Mathematics, 2007.

[5] Hoffert J., Schmidt D., and Gokhale A., "DQML: a Modeling Language for Configuring Distributed Publish/Subscribe Quality of Service Policies," *in Proceeding of 10th International Symposium on Distributed Objects, Middleware, and Applications*, Monterrey, pp. 515-534, 2008.

[6] Hoffert J., Mack D., and Schmidt D., "Integrating Machine Learning Techniques to Adapt Protocols for QoS-Enabled Distributed Real-Time and Embedded Publish/Subscribe Middleware," *International Journal of Network Protocols and Algorithms:Special Issue on Data Dissemination for Large-scale Complex Critical Infrastructures*, vol. 2, no. 3, pp. 1-33, 2010.

[7] Hoffert J. and Schmidt D., "Adapting Distributed Real-Time and Embedded Publish/Subscribe

Middleware for Cloud-Computing Environments," *in Proceeding of ACM/IFIP/USENIX 11th International Middleware Conference*, Bangalore, pp. 21-41, 2010.

[8] Ijaz S., Munir E., Anwar W., and Nasir W., "Efficient Scheduling Strategy for Task Graphs in Heterogeneous Computing Environment," *The International Arab Journal of Information Technology*, vol. 10, no. 5, pp. 486-494, 2013.

[9] King A., *Online K-Means Clustering of Nonstationary Data*, Prediction: Machine Learning and Statistics, 2012.

[10] Kotsiantis S., "Supervised Machine Learning: a Review of Classification Techniques," *Informatica*, vol. 31, no. 3, pp. 249-268, 2007.

[11] Mabrouk N., Beauche S., Kuznetsova E., Georgantas N., and Issarny V., "QoS-Aware Service Composition in Dynamic Service Oriented Environments," *in Proceeding of ACM/IFIP/USENIX International Conference on Distributed Systems Platforms and Open Distributed Processing*, pp. 123-142, 2009.

[12] Object Management Group, www.omg.org/spec, Last Visited 2014.

[13] Platania M., *Ordering, Timeliness and Reliability for Publish/Subscribe Systems Over WAN*, Sapienza University of Rome, 2011.

[14] Object Management Group, http://www.rti.com/products/index.html, Last Visited 2015.

[15] Schulzrinne H., State R., and Niccolini S., *Principles, Systems and Applications of IP Telecommunications. Services and Security for Next Generation Networks, chapter Automatic Adaptation and Analysis of SIP Headers Using Decision Trees*, Springer, 2008.

[16] Zhai L., Guo L., Cui X., and Li S., "Research on Real-Time Publish/Subscribe System Supported by Data-Integration," *Journal of Software*, vol. 6, no. 6, pp. 1133-1139, 2011.

Basem Almadani joined KFUPM in 2007 and Chaired Computer Engineering Department between 2009 and 2014. Before that, he was an assistant professor in Automation Institute at Montan University of Austria after receiving his PhD from the same university. Dr. Basem spent 12 years in the industry locally and internationally. He served in SABIC for four years and in several companies in Austria for another 8 years. His specialization is in Real-Time Mission Critical Systems and Middleware software. His areas of interest are Command and Control Systems (C4I-SR), WSN, and E-Government.

Shadi Abudalfa received the BSc and MSc Degrees both in Computer Engineering from the Islamic University of Gaza (IUG), Palestine in 2003 and 2010 respectively. He is a lecturer at the University Collage of Applied Sciences, Palestine. He is currently a PhD candidate in Computer Science and Engineering at King Fahd University of Petroleum and Minerals, Saudi Arabia. From July 2003 to August 2004, he worked as a research assistant at Projects and Research Lab in IUG. From February 2004 to August 2004, he worked as a Teaching Assistant at Faculty of Engineering in IUG.

Shuang-Hua Yang is a Professor in Computer Science at Loughborough University in the UK and an adjunct professor at King Fahd University of Petroleum and Minerals in Saudi Arabia. He is an associate editor for four academic journals including International Journal of Systems Science, Arabian Journal for Science and Engineering and a member of the editorial and advisory board for another two academic Journals. He is a fellow of the Institute of Measurement and Control since 2005, and a senior member of IEEE since 2003. He was the guest editor of six special issues on wireless sensor networks and Internet based control in various journals. He was awarded 2010 Honeywell Prize by the Institute of Measurement and Control in the UK to recognize his contribution to home automation research. He authored two research monographs published by Springer: Wireless sensor networks – principles, design and applications (2014), and Internet-based control systems design (2011).