International Journal of Control and Automation
Vol.7, No.5 (2014), pp.9-16
http://dx.doi.org/10.14257/ijca.2014.7.5.02

An Study on Threshold Tuning for Demand Queue Utilizing M/M/1 Model

XiaoChun Yin[1], ZengGuang Liu[2] and Hoon Jae Lee[3]

[1]*Dept of Ubiquitous IT, Graduate School of Dongseo University, Sasang-Gu, Busan 617-716, Korea*
WeiFang University of Science & Technology ShouGuang 262700, China

[2]*IP Platform Dept of MGC, Alcatel-Lucent, QingDao SongLing Road 169, China*

[3]*Division of Computer and Engineering of Dongseo University, Sasang-Gu, Busan 617-716, Korea*

yinspring2012@gmail.com, sterling.liu@alcatel-lucent.com, hjlee@dongseo.ac.kr

Abstract

In the telecom core network, one of the challenges is the unpredictable mass calling events, which suspend the switch operation because of over-loading function, in some worst case, even crash the local network. Overload Control Module, with a set of well-tuned thresholds, is the key point for Media Gateway Controller to survive under unpredictable mass calling events. In this paper, we propose a scheme to efficiently get a set of well- tuned thresholds. We did the experiments and provided the analysis, in order to verify our scheme and make it work well under different platforms.

Keywords: *Overload Control; Demand Queue; Threshold Tuning; M/M/1 Queue*

1. Introduction

In the telecom network, one of the challenges is the mass calling events, which suspend the switch operation because of over-loading function, in some worst case, even crash the local network. How to manage the system survive under mass calling events has become one of the most important issues. Overload Control Module (OCM) provides one solution. Using OCM, the system can automatically reduce the workload and maintain it in a normal state in the case of overloading. As a core network element, OCM plays a critical role in managing the system survive under mass calling events. The OCM is aiming at protecting the system against multiple overload conditions, including both external events and internal events. The external events are sent from the external network, like stimulated mass calling events; the internal events are triggered by the system's internal components, like special algorithm calculation, inter-process communication, fault recovery, etc. In this paper, we propose a threshold tuning scheme, which is aiming at setting the thresholds and make it work well under different platforms.

The rest of the paper is organized as following. We first provided preliminaries in Section 2. Then Section 3 discussed the proposed scheme. Section 4 provided the trial results and analysis. At last, in Section 5, we concluded our work and described the future research.

ISSN: 2005-4297 IJCA

2. Preliminaries

To facilitate of our proposed scheme, the following articles about M/M/1 Queue model are briefly introduced.

In queue theory, M/M/1 model is the special case of Kendall's notation which is the most basic and important foundation of queue theory. An M/M/1 queuing system is the simplest non-trivial queue where the requests arrive according to a Poisson process with rate λ, in which the inter arrival times are independent and exponentially distributed random variables with parameter λ. The service times are also assumed to be independent and exponentially distributed with parameter μ . Furthermore, all the involved random variables are supposed to be independent of each other. There are several metrics about the M/M/1 Queue, which are important for evaluating the system using the M/M/1 model and also we need to apply them in the Section 3.

- The time interval between two arrivals: $E[T] = 1/\lambda$
- Steady state probability of finding n customers in the system: $P(T < t) = 1 - e^{-\lambda t}$
- The traffic rate(the server utilization) : $\rho = \lambda/\mu$

When we are judging whether a system is an application of M/M/1 model, we must consider the following conditions. Firstly, the customers arrive should be in a passion process, it means the inter arrival time should be in exponential distribution. Secondly, the service time should also be in exponential distribution. Thirdly, there is only one server in the system and the number of the buffer slot should be infinite. At last, the system should be in stable fashion, it means the input rate λ should be smaller than the output rate μ, so that the server can work in a steady state.

3. Proposed Scheme

In order to investigate the method of setting well-tuned thresholds and make it work well under different platforms, we propose the threshold tuning scheme. In this scheme, we firstly analysis the work flow of MGC system and design the Demand Queue based OCM Model, secondly we apply the M/M/1 Queue in the analysis of the system. Thirdly, based on the model, we classify the thresholds into 4 static overload levels and propose the method of setting thresholds with the detailed analysis.

3.1 Demand queue based OCM modeling

Based on the function analysis of Media Gateway Controller (MGC) based OCM system, we design the model of MGC Demand Queue based on OCM, in the Figure 1, we can clearly get the idea how the system works. The Distributer Controller distributes the system's events into the Demand Queue (DQ) which maintains the system's internal and external events. The OCM samples the system's status from the DQ, and calculate the overload level, so that OCM can act on the system accordingly, different system can deploy different sample and OCM

samples the DQ depth for overload level calculation. According to the M/M/1 Queue theory, we can conclude that the Media Gateway Controller (MGC) based OCM system is an M/M/1 model, based on the following description:

- Call events are Poisson arrival process.
- MGC has exponential service times for a single server.
- DQ is a FIFO queuing discipline.

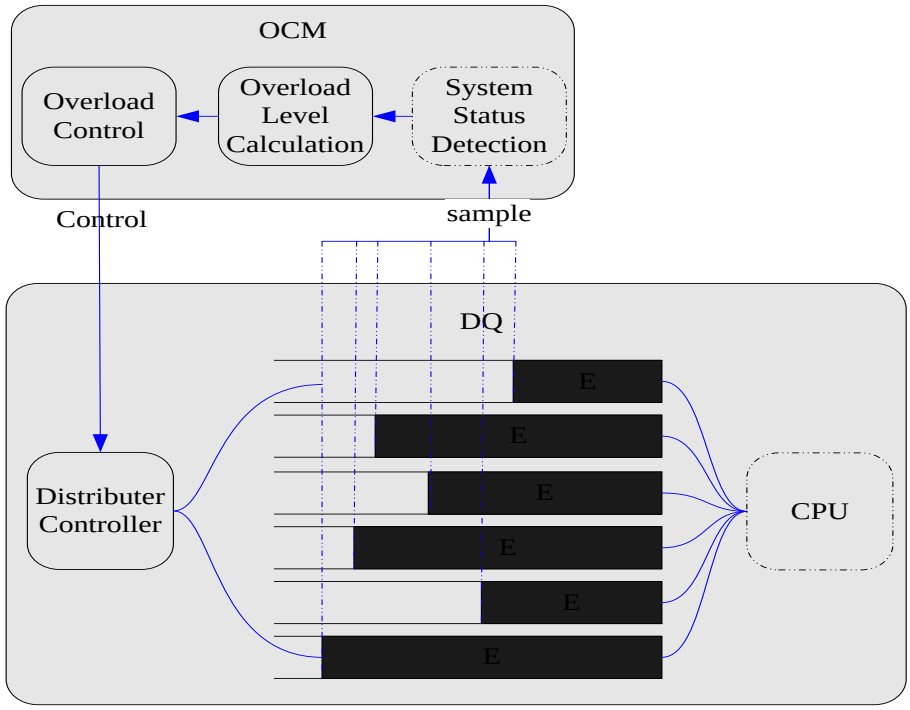

Figure 1. Demand Queue depth under different enqueue rate

3.2 Overloading level classification

In MGC, both internal events and external events are processed into the DQ. An event is out of the DQ when CPU time is allocated to process the DQ. The MGC OCM monitors the depth of the DQ and then acts accordingly. According to the length of DQ, we define 4 overload levels, and each level has two thresholds. One threshold is for onset and another threshold is for abating. Correspondingly, the OCM takes different action under different overload level:

- No Overload: All incoming calls are processed normally.
- Overload Level 1: Reject 20% normal calls, special calls (*e.g.*, priority calls, emergency calls, GETS calls, *etc.*) can pass through.
- Overload Level 2: Reject 80% normal calls, special calls (*e.g.*, priority calls, emergency calls, GETS calls, *etc.*) can pass through.

- Overload Level 3: Reject 100% normal calls, special calls (*e.g.*, priority calls, emergency calls, GETS calls, *etc.*) can pass through.
- Overload Level 4: Reject all calls including special calls.

3.3 Threshold tuning method

So far, we design the system model and predefine the levels of overloading which is the foundation of our threshold tuning method. Since the designed Model is an M/M/1 model and in order to clearly describe the method of setting threshold, we define the following parameters:

- λ : The enqueue rate (events/second)
- μ : The dequeue rate (events/second)
- $\rho = \lambda / \mu$:the utilization of the server
- The overload thresholds: $Abate_1 < Onset_1 < Abate_2 < Onset_2 < Abate_3 < Onset_3 < Abate_4 < Onset_4$
- Under overload level i (0<i<4), the dequeue rate can be looked on as constant value μ_i, and $\mu_0 < \mu_1 < \mu_2 < \mu_3 < \mu_4$

Before reaching overload level 1, when $\lambda < \mu_0 (\rho < 1)$, the DQ can be looked on as an M/M/1 queuing system. The possibility of which the queue depth N is less than $Onset_1$ is $p(N <= Onset_1) = 1 - \rho^{Onset_1}$. When $Onset_1$ is big enough, p is only related to ρ which is not supposed to change on different platforms, therefore the $Onset_1$ can be set to a common value across different platforms. As the enqueue rate increases until $\lambda > \mu_0 (\rho > 1)$, the queue depth will increase quickly until the DQ depth exceeds the $Onset_1$, then the OCM will start rejecting calls and it will result in the increasing of the dequeue rate. When the dequeue rate is up to μ_1, the DQ depth will decrease. when $\mu_0 < \lambda < \mu_1$, the overload level will bounce between level 1 and no overload. Similarly, when $\mu_1 < \lambda < \mu_2$, the overload level will bounce between level 2 and level 1, *etc.* The DQ depth under different enqueue rate is shown in Figure 2.

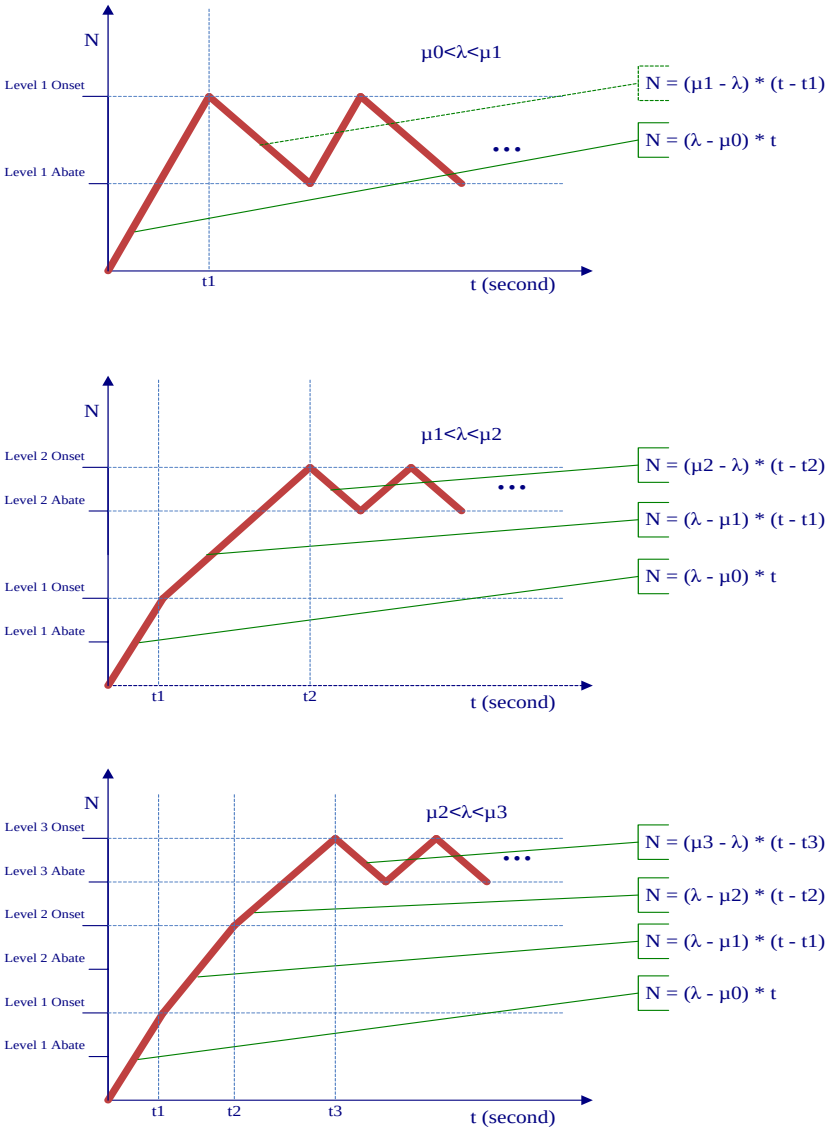

Figure 2. Demand Queue depth under different enqueue rate

4. Experiments and Analysis

Based on the scheme discussed in the above section, we did the experiments which are done through running a typical call load with the following threshold ($Onset_1 = 500, Abate_1 = 300, Onset_2 = 5000, Abate_2 = 4000$). The results are provided in the Figure 3, the DQ depth bounces between $Onset_2$ & $Abate_2$. As analyzed in above section, the DQ thresholds can be set to a common value across different platforms, regardless of the system's capacity. While under the real test bed, it shows that the result is biased due to noise, the enqueue rate is not a stable value λ , but $\lambda \pm \Delta$. The value of Δ causes the old OCM threshold does not work well on the new hardware platform and the threshold tuning is required.

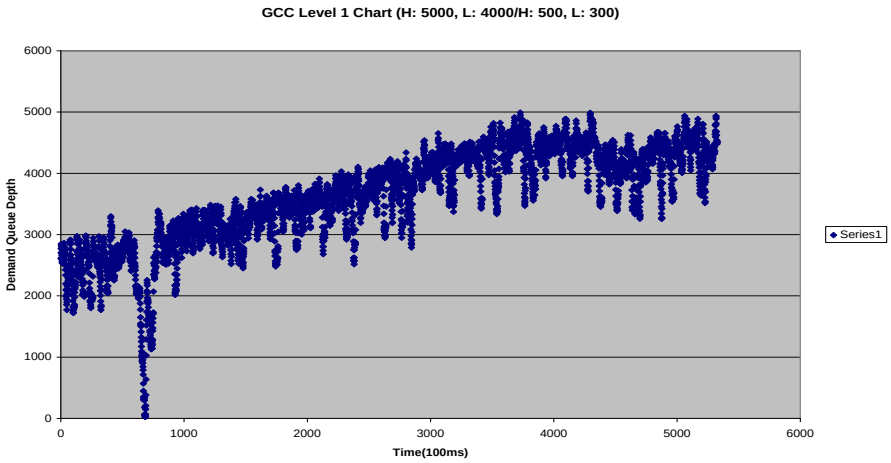

Figure 3. Test Result of Demand Queue depth

In order to avoid the overload level bouncing across multiple levels under a stable traffic load, we need to set the interval between thresholds large enough, *e.g.*, when $\mu_1 < \lambda < \mu_2$, it is expected that the system reports overload level 1 or level 2, but it should not hit level 3, nor no overload. It means: $Onset_2 + \Delta_2 < Onset_3$ and $Abate_2 - \Delta_2 > Abate_1$. Meanwhile, the thresholds should be small enough, so that the overload control can be triggered fast enough, then the DQ will not waste resources, nor cause processing delay. Based on the above analysis, following steps are taken in the MGC's threshold tuning:

Step 1: Collect $\mu_0, \mu_1, \mu_2, \mu_3$:

Firstly, we set the $Onset_1$ & $Abate_1$ and run the traffic to increase the traffic load. It will not report overload level 1 until the traffic is big enough up to per system's capacity requirement. When the system starts reporting overload level 1, record the traffic load λ_1, which should be very close to μ_0. Secondly, we set initial $Onset_2$ & $Abate_2$ to a much higher value than $Onset_1$ & $Abate_1$, so that we can get $Onset_1 + \Delta_1 < Onset_2$, and run the traffic to increase the traffic load until the system start reporting overload level 2, then record the traffic load λ_2, which should be very close to μ_1. At last, we repeat the above steps to collect the λ_3 & λ_4, which should be very close to μ_2 & μ_3.

Step 2: Monitor the DQ depth under $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ to calculate $\Delta_1, \Delta_2, \Delta_3, \Delta_4$

At this step, we first set initial $Onset_2$ & $Abate_2$ to a much higher value than $Onset_1$ & $Abate_1$ and run the traffic load λ_1, meanwhile monitor the DQ depth when the traffic is running and collect the highest DQ depth $High_1$, we can get $\Delta_1 = High_1 - Onset_1$. Secondly, we set initial $Onset_3$ & $Abate_3$ to a much higher value than $Onset_2$ & $Abate_2$ and run the traffic load λ_2, meanwhile monitor the DQ depth when the traffic is running, collect the highest DQ depth $High_2$ and the lowest DQ depth Low_2. Then we can get $\Delta_2 = MAX((High_2 - Onset_2), (Abate_2 - Low_2))$. At last, we repeat above steps to collect Δ_3, Δ_4.

Step 3: Set the thresholds accordingly:

$$Onset_2 = Onset_1 + \Delta_1 + 100$$

$$Abate_2 = Abate_1 + \Delta_2 + 100$$

$$Onset_3 = Onset_2 + \Delta_2 + 100$$

$$Abate_3 = Abate_2 + \Delta_3 + 100$$

$$Onset_4 = Onset_3 + \Delta_3 + 100$$

$$Abate_4 = Abate_3 + \Delta_4 + 100$$

Rerun the test with the adjusted thresholds; verify the system's behavior can meet the requirement on all platforms.

5. Conclusion and Future Research

In this paper, we proposed a threshold tuning scheme and investigated the method of setting well-tuned thresholds. We did the experiments and provided the analysis to verify our scheme and make sure it work well under different platforms. With the evolvement of the system's hardware and software, all the OCM's thresholds need to be tuned each time with the system's capacity growth. Is there any method that the OCM can be self-adaptive to avoid such changes? The dynamic overload control method is under investigation for the future enhancement. Unlike the static predefined thresholds for multiple overload levels, the dynamic overload controlling rejects the call events based on the actual demand queue depth. For example, the rejection rate can be 0 when DQ depth $N <= Onset_1$, and the rejection rate can be 100% when DQ depth $N >= Onset_2$. When $Onset_1 < N < Onset_2$, the rejection rate can be $(N - Onset_1)/(Onset_2 - Onset_1)$ or other kind of calculation formula based on the $Onset_1 \& Onset_2$. In this way, the $Onset_1 \& Onset_2$ can be set to a big enough value regardless of the system's capacity and this method can make the DQ demand queue depth changes more smoothly. More analysis and verification is required for this dynamic overload control method in the future.

Acknowledgements

This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education, Science and Technology. (Grant number: 2013-071188). And it also supported by the BB21 project of Bussan Metropolitan City.

References

[1] M. J. Whitehead and P. M. Williams, "Adaptive network overload controls", BT Technology Journal, vol. 20, no. 3, **(2002)** July, pp. 1-30.
[2] M. Andrews, "Maximizing Profit in Overloaded Networks", Bell Laboratories, Lucent TechnologiesMurray Hill, (**2002**), NJ 07974.
[3] S. Floyd and V. Jacobson, "Random early detection gateways", **(1993)**.
[4] S. Kunniyur and R. Srikant, "Analysis and design of an adaptive virtual queue (AVQ) algorithm for active queue management", in Proc. Of ACM SIGCOMM, San Diego, USA, (**2001**).
[5] C. V. Hollot, V. Misra, D. Towsley and W. Gong, "On designing improved controllers for AQM routers supporting TCP flows", in Proc.of INFOCOM, Anchorage, USA, (**2001**).

[6] S. Athuraliya, S. H. Low, V. H. Li and Q. Yin, "REM: active queue management", IEEE Network Magazine, vol. 15, no. 3, **(2001),** pp. 48-53.
[7] S. Floyd, R. Gummadi and S. Shenker, "Adaptive RED: an algorithm for increasing the robustness of RED's active queue management", **(2001)**, http://www.icir.org/floyd/papers/adaptiveRed.pdf.
[8] J. Aweya, M. Ouellette and D. Y. Montuno, "A control theoretic approach to active queue management", Computer Networks, vol. 36, no. 2-3, (**2001**), pp. 203-235.
[9] B. Abbasov and S. Korukoglu, "Effective RED: an algorithm to improve RED's performance by reducing packet loss rate", Journal of Network and Computer Applications, vol. 32, no. 3, **(2009)**, pp. 703–709.
[10] L. Hu and A. D. Kshemkalyani, "HRED: a simple and efficient active queue management algorithm", in Proc. of International Conference on Computer Communications and Networks (ICCCN'04), Chicago, USA, **(2004).**
[11] W. Chen and S. Yang, "The mechanism of adapting RED parameters to TCP traffic", Computer Communications, vol. 32, no. 13-14, **(2009)**, pp. 1525-1530.
[12] H. Wang, C. Liao and Z. Tian, "Effective adaptive virtual queue: a stabilising active queue management algorithm for improving responsiveness and robustness", IET Communications, vol. 5, no. 1, **(2011)**, pp. 99-109.
[13] X. C. Yin, H. J. Lee, "An efficient Threshold tuning Method of OCM based M/M/1 queue", Advanced Science and Technology Letters, Embedded Ubiquitous, vol. 38, **(2013),** pp. 44-47.

Authors

Xiao Chun Yin

She received the B.S. degree in education and technology from Qufu Normal University, Qufu, China in 2004, and received the M.S. degree in education and technology from Nanjing Normal University, Nanjing, China in 2007. She had been working as a lecturer in Weifang University of Science & Technology, China from 2008 to 2012. Currently she is a doctoral candidate in cryptography and network security at Dongseo University, Korea. Her research interests include network security, cloud security, authentication protocol and real-time communication.

ZengGuang Liu

He received the B.S. and M.S. from Dept. of computer engineering, University of ShangHai for Science and Technology, China in 2005 and 2008 respectively. He is a senior software engineer at IP platform dept. of Alcatel-Lucent, QingDao, China from 2008. His research interests include Operation System and real-time communication.

Hoon Jae Lee

He received the B.S., M.S. and Ph.D. degree in Electrical Engineering from Kyungpook national university in 1985, 1987 and 1998, respectively. He had been engaged in the research on cryptography and network security at Agency for Defense Development from 1987 to 1998. Since 2002 he has been working for Department of Computer Engineering of Dongseo University as an associate professor, and now he is a full professor. His current research interests are in security communication system, side-channel attack, USN & RFID security. He is a member of the Korea institute of Information security and cryptology, IEEE Computer Society, IEEE Information Theory Society and *etc.*

