Advanced Science and Technology Letters
Vol.38 (Embedded Ubiquitous 2013), pp.44-47
http://dx.doi.org/10.14257/astl.2013.38.10

# An efficient Threshold tuning Method of OCM based M/M/1 queue

Xiaochun Yin[1], Hoon Jae Lee[2]

[1]Department of Ubiquitous IT, Graduate School of Dongseo University,
Sasang-Gu, Busan 617-716, Korea
Weifang University of Science & Technology
Weifang 262700, China
yinspring2012@gmail.com

[2]Division of Computer and Engineering Dongseo University
Sasang-Gu, Busan 617-716, Korea
hjlee@dongseo.ac.kr

**Abstract.** Unpredictable mass calling events can suspend the switch operation because of over-loading function, sometimes 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 method to efficiently get a set of well – tuned thresholds. Additionally, we provided the analysis in detail in order to verify our method.

**Keywords:** Overload control; Demand queue; Threshold tuning; M/M/1 queue

## 1 Introduction

In the telecom network, one of the challenges is the unpredictable mass calling events, which suspend the switch operation because of overloading function, in some worst case, even crash the local network. How to make the system survive under mass calling events has become the most important issue. Overload Control Module (OCM) provides the solution, with OCM, the system can automatically reduce the workload and keep it in a normal state when it is overloading. As a core network element, OCM plays a critical role that the system can survive under mass calling events. The OCM is aiming at protecting the system against multiple overload conditions, including both external events and internal events. In this paper, we propose a threshold tuning method, which is aiming at setting the thresholds and make it work well under different platforms.

ISSN: 2287-1233 ASTL

# 2 Proposed method

## 2.1 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 fig. 1, from which we can clearly get the idea how the system works. The Distributer Controller distributes the system's events into the Demand Queue (DQ) which maintain 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 get that the Media Gateway Controller (MGC) based OCM system is an M/M/1 model.

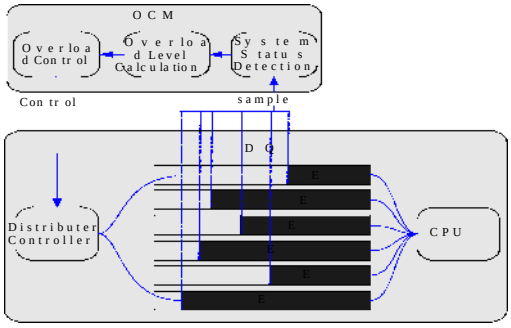


Fig. 1: Demand Queue depth under different enqueue rate

## 2.2 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 is for onset and another is for abating. Correspondingly, the OCM takes different action under different overload level. When there is no overload, all incoming calls are processed normally; at Overload level 1, the system will reject 20% normal calls, special calls (e.g. priority calls, emergency calls, GETS calls, etc.) can pass through; at level 2, it will reject 80% normal calls, special calls (e.g. priority calls, emergency calls, GETS calls, etc.) can pass through; at level 3, it will reject 100% normal calls, special calls (e.g. priority calls, emergency calls, GETS calls, etc.) can pass through; at level 4, it will reject all calls including special calls.

### 2.3 Threshold tuning method

Since the OCM model is an M/M/1 model, in order to describe the method of setting threshold clearly, we define the following arguments: $\lambda$ represents enqueue rate (events/second); $\mu$ denotes dequeue rate (events/second). The overload threshold should satisfy $Abate_1 < Onset_1 < Abate_2 < Onset_2 < Abate_3 < Onset_3 < Abate_4 < Onset_4$, under overload level i($0<i<4$) $\mu$ can be consider as constant value, 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 considered as an M/M/1 queuing system. The possibility of the queue depth $N$ is less than $Onset_1$ is $p(N <= Onset_1) = 1 - \rho^{Onset_1}$. When $Onset1$ is big enough, it is related to $\rho$ which is not supposed to change on different platforms. Therefore the $Onset1$ can be set to a common value across different platforms. As the enqueue rate increases until $\lambda > \mu_0 (\rho > 1)$, then the queue depth would increase quickly until the DQ depth exceeds the $Onset1$. The OCM will start rejecting calls and the dequeue rate is increased to $\mu_1$, then the DQ depth would decrease when $\mu_0 < \lambda < \mu_1$, the overload level would bounce between level 1 and no overload. Similarly, when $\mu_1 < \lambda < \mu_2$, the overload level would bounce between level 2 and level 1, etc..

## 3. 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, from which we can clearly get that 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 shows that the result is biased due to noise, the enqueue rate is not a stable value $\lambda$, but $\lambda \pm A$. The value of A causes the old OCM threshold does not work well on the new hardware platform and the threshold tuning is required.

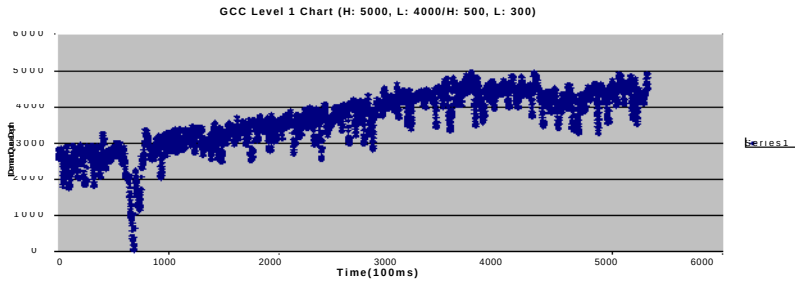


 

Fig. 3: Test Result of Demand Queue depth

In order to avoid the overload level bounces 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 should not hit level 3, nor no overload. It means: $Onset_2 + A_2 < Onset_3$ and $Abate_2 - A_2 > Abate_1$. Meanwhile, the thresholds should be small enough so that the overload control can be triggered fast enough, then the DQ would not waste resources, nor cause processing delay.

## 4 Conclusion and future research

With the system's hardware or software evolvement, 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 pre-defined thresholds for multiple overload levels, the dynamic overload control reject the call events based on the actual demand queue depth, and it 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.

**ACKNOWLEDGMENT**

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 in 2013.

## References

1. M J Whitehead, P M Williams, Adaptive network overload controls, BT Technology Journal, July 2002, Vol 20 No3, 1-30
2. Matthew Andrews, Maximizing Profit in Overloaded Networks, Bell Laboratories, Lucent TechnologiesMurray Hill, 2002, NJ 07974
3. S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance", IEEE/ACM Transactions on Networking, 1(4):pp. 397-413, 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, 15(3): pp. 48-53, 2001.
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.

 