International Journal of Future Generation Communication and Networking
Vol.7, No.4 (2014), pp.81-90
http://dx.doi.org/10.14257/ijfgcn.2014.7.4.08

The Metadata Quality of Service Differentiation for Multi-priority in Cloud Computing

Yang Zhiyong[1, 2], Li Chunlin[1, 3], Li Layuan[1] and Liu Yanpei[1]

[1]Department of Computer Science, Wuhan University of Technology, China
[2]Key Laboratory of Fiber Optic Sensing Technology and Information Processing, Ministry of Education, Wuhan University of Technology, China
[3]State Key Laboratory for Novel Software Technology, Nanjing University, China
yzywhlgdx@163.com, chulin74@aliyun.com, jwtu@public.wh.hb.cn, mantianxingok@163.com

Abstract

To ensure the Quality of Service (QoS) of the high-priority requests, the paper proposes a Congestion Control Algorithm of Multi-Priority (CCAMP) which differentiates the QoS according to the priorities, CCAMP reserves queue buffer for the high-priority packets by dropping the low-priority packets in a certain probability. Moreover, the paper presents an improved EDF (Earliest Deadline First) schedule of multi-priority with the sliding window which sorts the packets according to the estimated completion time, the sliding window defines the time similarity, and the high-priority packets are scheduled in priority in the sliding window. The results of the experiments prove that CCAMP and the improved EDF schedule reduces the delay and the loss rate of the high-priority packages, they realize the QoS differentiation by priorities effectively.

Keywords: *congestion control, queue schedule, QoS*

1. Introduction

Cloud computing has become popular for its economical, reliable, convenient services in recent years [1]. With the popularity of the cloud services, a variety of cloud platforms have emerged, such as the Google's GFS, Microsoft's Windows azure, Amazon's EC2 *etc.*

Currently, the master-slave structure is used in most of the cloud computing systems such as Hadoop Distribute File System (HDFS) [2]. The master node (called Name Node) is the metadata server that manages the file system's metadata; the slave nodes (called Data Node) are the numerous data servers which store the actual data. All the requests from the clients must be processed as the metadata request in the name node firstly [3]. Therefore, Name Node may become the bottleneck in the case of surging requests.

When the metadata requests are submitted to the name node, the congestion controller determines whether the requests are accepted to enter the buffer queue or discarded. The purpose of congestion control is to meet the QoS of the high-priority requests by discarding the low-priority packets. The schedule flow chart is as Figure 1. The schedule strategy sorts the requests according to estimated completion time, and schedules the high-priority requests in the sliding window in priority.

Copyright ⓒ 2014 SERSC

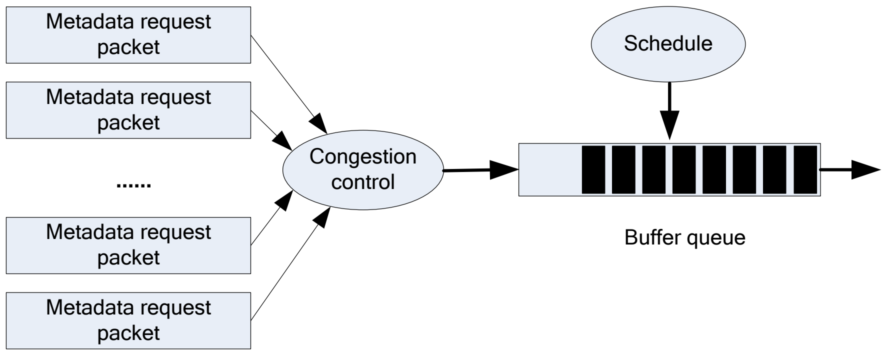

Figure 1. The Congestion Control and Schedule Procedure

The rest of the paper is organized as follows. Section 2 describes the related work. In Section 3, we present the congestion control algorithm of multi-priority and Section 4 proposes an improved EDF schedule of multi-priority with the sliding window. In Section 5, we evaluate the QoS differentiation effect by a series of experiments. Finally, we conclude in Section 6.

2. Related Work

Drop-tail [4-6] drops the arrival packets when queue overflow occurs. Drop-tail incurs large queue length and high packet loss rate at congested links. Especially, Drop-tail results in a phenomenon, called global synchronization, when a lot of data flows compete in a bottleneck. However, Drop-tail is the most widespread queue management scheme due to its simplicity.

Active Queue Management (AQM) [7, 8] is a proactive congestion control mechanism, AQM is different from drop-tail and it drops packets before the queue is full and effectively solves the global synchronization problem.

Random Early Detection (RED) [9-11] is a queue management scheme that is intended to remedy the shortcomings of Drop-tail. The dropping probability of RED is decided by the queue length, it is an early congestion notification, and the dropping probability increases in order to provide enough early congestion notifications.

When a packet arrivals at the buffer queue, the new average queue length is updated by using the current queue length q_{real} and the old value of the average queue length q_{avg}.

$$q_{avg} = (1 - w_q) * q_{avg} + w_q * q_{real}$$

The parameter w_q determines the weight given to the new queue measurement, q_{avg} is the average queue length, and q_{real} is the actual queue length.

$$P_{drop} = p_{\max} * (q_{avg} - \min_{th}) / (\max_{th} - \min_{th})$$

The dropping probability P_{drop} is calculated based on the minimum and maximum queue thresholds $\min_{th}$ $\min_{th}$ and $\max_{th}$ respectively which are usually set as some percentage of the maximum buffer capacity, as shown in Figure 2, the parameter $p_{\max}$ determines how aggressively packets are dropped as the queue builds.

RIO (RED with In/Out) [12, 13] differentiates the arriving packets into two classes: In packets and out packets. If the arriving rate of the packets is bigger than the subscription rate, it will be marked with in packet; else it will be marked with out packet. RIO expends RED to two set of parameters: ($\min_{in}, \max_{in}, p\max_{in}$) and ($\min_{out}, \max_{out}, p\max_{out}$), it is shown as Figure 3. They present the minimum, maximum queue thresholds and the maximum dropping probability of the In and Out packets. When congestion occurs, the Out packets will be dropped earlier than in packets to reserve more buffers for In packets.

The gentle RED [14] modification extended maximum threshold to twice value. However, although we may exploit fully the buffer space in this way, packet-drops do not have always desirable effects. Some applications that generate a small amount of critical data do not exit from slow start and may delay by packet drops.

Adaptive RED [15] avoids link underutilization by maintaining the average queue length among the two thresholds by adjusting $p_{\max}$.

BLUE [16] manages dropping based on packet loss and link idle events, if the queue drops packets due to buffer overflows, BLUE increases the dropping probability, whereas if the queue becomes empty or idle, BLUE decreases the dropping probability.

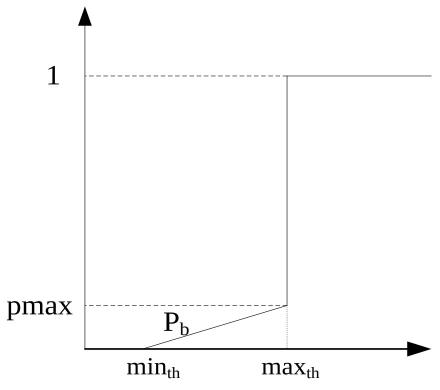

Figure 2. The Parameter of the RED

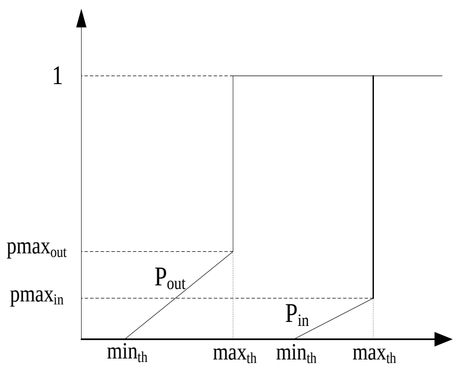

Figure 3. The Parameter of the RIO

3. The Congestion Control Algorithm of Multi-Priority (CCAMP)

Although the classical RED can avoid congestion by dropping part of the packets previously, it can’t provide differentiated QoS for no classification to the requests. RIO differentiates the arriving packets into Out and In packets, which can’t provide multi-priority QoS differentiation.

The paper proposes a multi-priority multi-threshold congestion control method which overcomes the above disadvantages. The higher the priority is, the bigger the threshold is. CCAMP reserves more buffers for the requests of high priority by dropping the packets of low priority.

Because the data processed by the metadata server is the metadata, it does not involve in the real data, so the request packets in the metadata server are with the same size in the buffer queue, that is, the processing speed to the request metadata is a constant value.

Before computing the dropping probability, we defined several parameters:

N : N is the number of the priorities.

PR_i: PR_i is the i^{th} request priority, $i \in [1, N]$, the bigger i is, the higher the priority is.

$\min_i$: $\min_i$ is the minimum threshold of the i^{th} priority.

$\max_i$: $\max_i$ is the maximum threshold of the i^{th} priority.

$p\max_i$: $p\max_i$ is the maximum dropping probability of the i^{th} priority.

Pb_i: Pb_i is the real dropping probability of the i^{th} priority.

$L_{\max}$: $L_{\max}$ is the maximum length of the buffer queue.

L_{avg}: L_{avg} is the average length of queue.

L_{real}: L_{real} is the real length of the queue.

α_i: α_i is the adjusting parameter of the minimum threshold. $\alpha_i \in [0,0.5]$.

β_i: β_i is the adjusting parameter of the maximum threshold. $\beta_i \in [0,0.5]$.

V_{in}: V_{in} is the arriving speed of the packets.

V_{out}: V_{out} is the processing speed of the metadata server.

If V_{in} is less than V_{out}, the metadata server can handle the request packets in time, so all request packets should be allowed to go into the buffer queue in the condition. Otherwise, the congestion control procedures should be started to achieve QoS differentiation.

When a request packet arrives, the average queue length is calculated as follow:

$$L_{avg} = (1-w) * L_{avg} + w * L_{real}$$

w is the adjusting weight, and the initial average queue length is the real queue length.

The minimum threshold of the queue length is set as follow:

$$\min_i = \gamma * L_{\max} + \frac{(i-\alpha_i) * L_{\max}}{2 * N} \qquad i < N$$

And the maximum threshold of the queue length is set as follow:

$$\max_i = \gamma * L_{\max} + \frac{(i+\beta_i) * L_{\max}}{2 * N} \qquad i < N$$

γ is the initial parameter for the buffer queue congestion control, if $\gamma = 0.5$, the congestion control procedures will be started when the buffer queue is in near half-full condition.

The dropping probability is calculated as follow:

if $i = N, Pb_i = 0$;

if $1 \le i \le N-1$, then:

$$Pb_i = \begin{cases} 0 & L_{avg} < \min_i \\ \dfrac{L_{avg} - \min_i}{\max_i - \min_i} * p\max_i & \min_i \le L_{avg} \le \max_i \\ 1 & L_{avg} > max_i \end{cases}$$

4. The Improved EDF Schedule of Multi-priority with the Sliding Window

In HDFS, The scheduler processes the request packets from the buffer queue in a strict FIFO order, the queue schedule mechanism does not differentiate the requests, which will lead to a long time delay when the congestion occurs. Because the main parameter of the QoS is the delay time, and the processing rate of the metadata is a constant value, so the processing delay mainly depends on the schedule way in the buffer queue. The higher the priority is, the shorter the expected delay is, so the EDF strategy is good solution to the schedule.

EDF [17, 18] algorithm is a well-known real-time schedule algorithm, the priority of the task is dynamically allocated according to the deadline of the task and the task of the highest priority will be processed firstly.

The paper proposes an improved schedule algorithm on EDF with sliding window. The procedure of the strategy in the paper is as follow:

(1) The metadata packet goes into the buffer queue according to its arriving time, such as Figure 4.

(2) Sort the packets in the queue according to EDF, that is, the estimated completion time is earlier, the packet in the queue is sorted in the front of the queue. Because the higher the priority is, the shorter estimated delay is, so if the packets with different priorities arrive in close time, the packet with higher priority will be put more forward than the packet with lower priority, such as Figure 5.

(3) A sliding window is applied to present the similarity of the estimated completion time, the packets in the same window have the similar estimated completion time, the packet with higher priority is scheduled with priority in the sliding window, such as Figure 6.

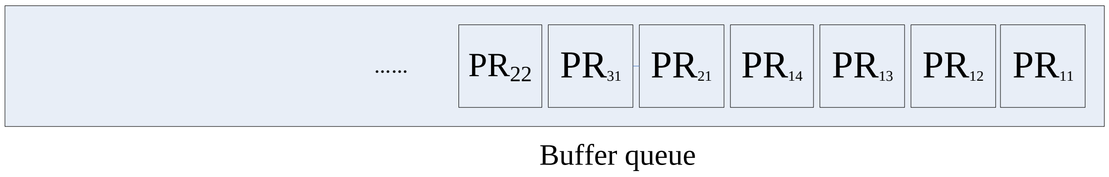

Figure 4. The Buffer Queue Sequence before Schedule

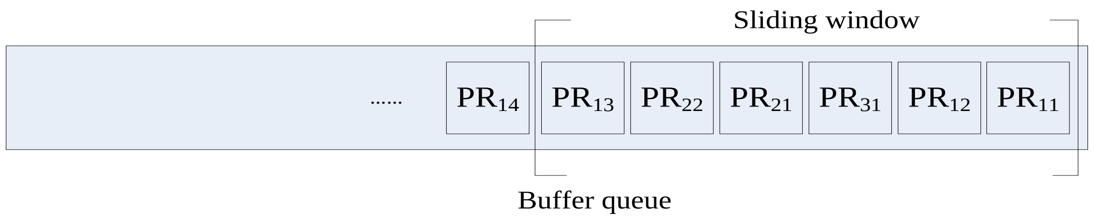

Figure 5. The Buffer Queue Sequence in EDF Schedule

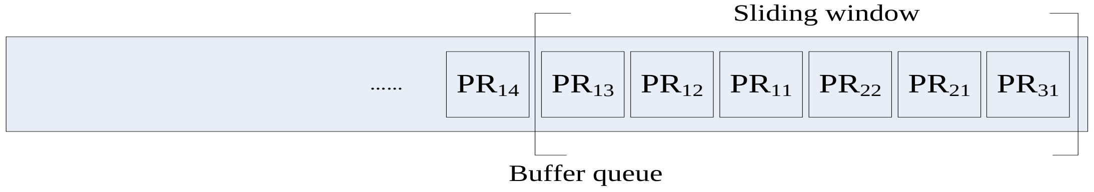

Figure 6. The Buffer Queue Sequence in EDF Schedule with Sliding Window

5. Experiment and Analysis

The metadata are with the same size, and the processing speed of the metadata in the metadata server is a constant value, therefore, CCAMP can be simplified as a problem to enter the queue and schedule in the queue, the experiment is simulated by the visual C++ program. It is assumed that the length of the buffer queue is 8000, the initial length of the buffer queue is 4000, there are four priorities, the adjusting parameters of the minimum and maximum thresholds are all set as 0.4, that is, $\alpha = \beta = 0.4$, the initial parameter of the congestion control is set as 0.5, that is, $\gamma = 0.5$, the parameters are set as Figure 7.

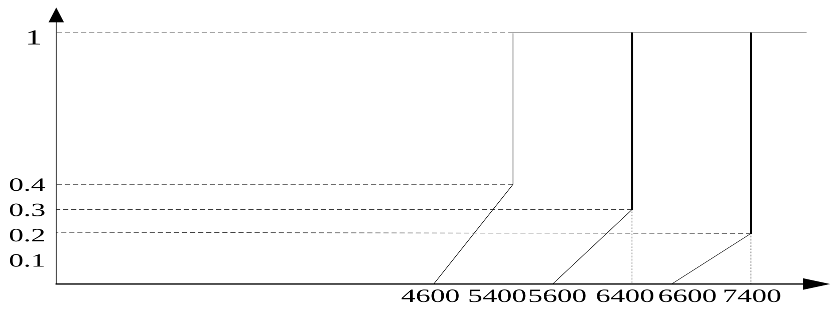

Figure 7. The Parameters for Multi-priority

Experiment 1: the reception contrast to the packets with different priority

The generation rate of the request packets for each priority is set as 100/s, the accepted amount of the request packets with different priority is as Figure 8.

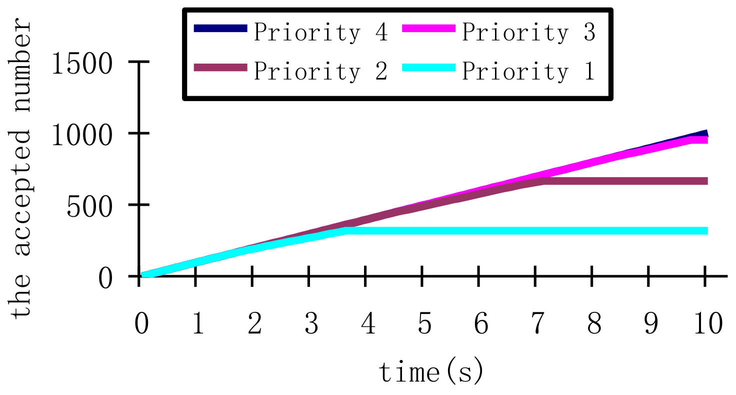

Figure 8. The Reception Contrast to the Packets with Different Priority

Copyright ⓒ 2014 SERSC

As shown in Figure 7 and Figure 8, at first, the average length of the queue is less than the minimum threshold of the lowest priority, all request packets are accepted to enter the queue, but with the growth of the queue length, the request packets with the lowest priority are dropped in a certain probability, when the average length of the queue is bigger than the maximum threshold, all request packets with the lowest priority are dropped, so the number of the accepted packets keeps constant after some time. Similar with the lowest priority, the packets with different priorities are controlled to enter the queue from low to high, and the packets of the highest priority are all accepted to go into the queue. In the experiment, the accepted number of the fourth priority is nearly three times than the number of the first priority. So the CCAMP algorithm realizes the QoS differentiation of multi-priority when the congestion occurs, CCAMP reserves buffer for the request packets of the higher priority by dropping the request packets of the lower priority.

Experiment 2: the delay contrast between the request packets with different priority.

Six groups of the delay experiments are designed. In the experiments, the generation rate of the request packets with every priority is set as 100/s, 200/s, 300/s, 400/s, 500/s, 600/s.The average delay of the request packets is counted as Figure 9.

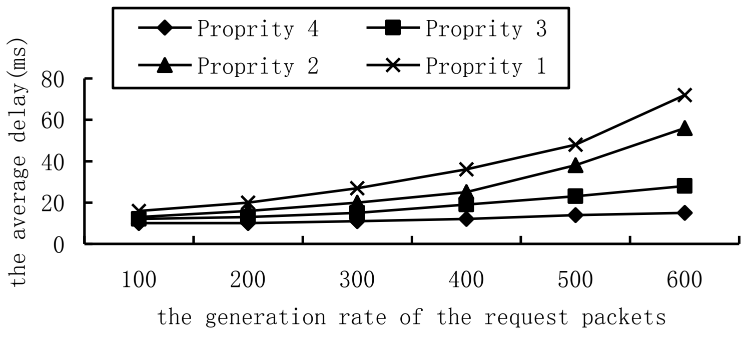

Figure 9. The Delay Contrast in the Different Generation Rate of the Request Packets

As shown in Figure 9, when the generation rate of the request packets is a fixed value, the average delay of the request packets of the high priority is shorter than the low priority. The main reason is the application of the EDF with sliding window. Because the request packets with high priority have shorter estimated completion time, so if the request packets with different priority come in near time, the request packets with higher priority will be sorted in front of the queue. At the same time, the request packets of the high priority in the sliding window are scheduled in priority, so the mechanism can shorten the average delay for the request packets of the higher priority. When the generation rate is low, the difference between the different priorities is little, but with the growth of the generation rate, the delay difference will increase dramatically.

Experiment 3: the average delay in different sizes of the sliding window

Six groups of the delay experiments are designed. In the experiments, the size of the sliding window is set as 400, 800, 1200, 1600, 2000, 2400.The average delay of the request packets is counted as Figure 10.

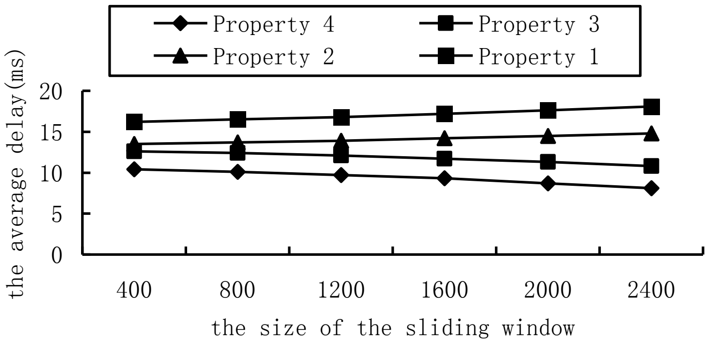

Figure 10. The Delay vs the Size of the Sliding Window

As shown in Figure 10, the average delay with higher priority decreases with the growth of the sliding window size, however, the average delay with lower priority increases with the growth of the sliding window size. The amount of the request packets with higher priority will increase with the growth of the sliding window, the request packets with higher priority is scheduled in priority, so it shortens the delay of the packets with higher priority and increases the delay of the packets with lower priority.

6. Conclusion

The paper proposes a multi-priority congestion control method and the EDF schedule with a sliding window. It reserves buffers for the request packets with higher priority by dropping the packets with lower packets, so the request packets with higher priority has bigger probability to enter the buffer queue; for the request packets with higher priority have less estimated completion time, so the EDF strategy will schedule the request packets with higher priority firstly in similar condition, and the schedule strategy in the sliding window decreases the delay for the request packets with higher priority further. So the comprehensive mechanism can differentiate the QoS for multi-priority.

Acknowledgements

The work was supported by the National Natural Science Foundation (NSF) under grants (No.61171075), Special Fund for Fast Sharing of Science Paper in Net Era by CSTD (FSSP) No.2013014311021, Specialized Research Fund for the Doctoral Program of Higher Education under Grant No.20120143110014, Program for the High-end Talents of Hubei Province, and the Open Fund of the State Key Laboratory of Software Development Environment (SKLSDE-2013KF). Any opinions, findings, and conclusions are those of the authors and do not necessarily reflect the views of the above agencies.

References

[1] L. Junzhou, J. Jiahui, S. Aibo and D. Fang, “Cloud computing: architecture and key technologies”, Journal on Communications, vol. 7, no. 32, **(2001)**.
[2] K. Shvachko, H. Kuang, S. Radia and R. Chansler, “The hadoop distributed file system, In Mass Storage Systems and Technologies (MSST)”, 2010 IEEE 26th Symposium on IEEE, **(2010),** pp. 1-10.
[3] W. Yijie, S. Weidong, Z. Song, P. Xiaoqiang and L. Xiaoyong, “Key Technologies of Distributed Storage for Cloud Computing”, Journal of Software, vol. 4, no. 23, **(2012)**.

[4] H. J. Kim, P. K. Park, H. S. Yoon and S. G. Choi, "QoS-aware Active Queue Management Scheme for es", 13th International Conference on Advanced Communication Technology, ICACT, 042.
[5] D. Vleeschauwer, S. Wittevrongel and H. Bruneel, "Dimensioning Drop-tail and AQM access networks for optimal performance with bulk data TCP traffic", Computer uppl., vol. 33, **(2010)**.
[6] ng, F. Wirth and D. Leith, "Modeling TCP congestion control dynamics in drop-tail tomatica, vol. 3, no. 43, **(2007)**.
[7] -GREEN: An active queue management mechanism for multi-QoS classes", Computer Standards & Interfaces, vol. 1, no. 36, **(2013)**.
[8] P. Wang, H. Chen, X. Yang and Y. Ma, "Design and analysis of a model predictive controller for active queue management", ISA Transactions, vol. 1, no. 51, **(2012)**.
[9] M. Hosamo, S. P. Singh and A. Mohan, "Random Early Detection method for ABR service", Computers & Electrical Engineering, vol. 4, no. 34, **(2008)**.
[10] A. A. Abouzeid and S. Roy, "Modeling random early detection in a differentiated services network", Computer Networks, vol. 4, no. 40, **(2002)**.
[11] J. V. Chen, F. C. Chen, J. M. Tarn and D. C. Yen, "Improving network congestion: a RED-based Fuzzy PID approach", Computer Standards & Interfaces, vol. 5, no. 34, **(2012)**.
[12] X. Yang and K. Kiseon, "Congestion Control of Differentiated Service Network", Chinese Journal of Electronics, vol. 1, no. 19, **(2010)**.
[13] C. Yu-bao, C. Yan, H. Zhi-Guo, K. Ya-nan and F. Qiang, "An Active Queue Management Algorithm Based on DiffServ Model", 2nd International Workshop on Education Technology and Computer Science, ETCS, **(2010),** pp. 455-458.
[14] S. Dimitriou and V. Tsaoussidis, "Adaptive Head-to-Tail: Active Queue Management based on implicit congestion signals", Computer Communications, vol. 2, no. 32, **(2009)**.
[15] Z. Jingjun, X. Wenlong and W. Liguo, "An Improved Adaptive Active Queue Management Algorithm Based on Nonlinear Smoothing", 2011 International Conference on Advanced in Control Engineering and Information Science, **(2011),** pp. 2369-2373.
[16] W. C. Feng, K. G. Shin, D. D. Kandlur and D. Saha, "The BLUE active queue management algorithms", IEEE/ACM Transactions on Networking, vol. 4, no. 10, **(2002)**.
[17] M. Bertogna and S. Baruah, "Tests for global EDF schedulability analysis", Journal of Systems Architecture, vol. 5, no. 57, **(2011)**.
[18] S. Kato and N. Yamasaki, "Global EDF-based scheduling with laxity-driven priority promotion", Journal of Systems Architecture, vol. 5, no. 57, **(2011)**.

Authors

Yang Zhiyong, he is currently a PhD student in the Department of Computer and Science at Wuhan University of Technology. He received his B.S. from Wuhan University of Technology, China, in 2001 and M.S. from Wuhan University of Technology in 2007. His research interests are in cloud computing, smart and Intelligent Computing and computer network.

Li Chunlin, she is a Professor of Computer Science in Wuhan University of Technology. She received the M.S in Computer Science from Wuhan Transportation University in 2000, and PhD in Computer Software and Theory from Huazhong University of Science and Technology in 2003. Her research interests include cloud computing and distributed computing.

Li Layuan, he is a Professor of Computer Science in Wuhan University of Technology. He received his B.E. in communication engineering from Harbin Institute of Military Engineering, China in 1970 and M.E. in communication and electrical systems from Huazhong University of Science and Technology, China in 1982. Her research interests include high speed computer network, protocol engineering, and image processing.

Liu Yanpei, she is a currently a PhD Student in the Department of Computer and Science at Wuhan university of technology. She received her B.S. from Luoyang Normal University, Luoyang, in 2006 and M.S. from Nanchang Hangkong University, China, in 2009. She has worked at Henan Institute of Science and Technology since 2009. Her research interests are in cloud computing.

Copyright ⓒ 2014 SERSC