TELKOMNIKA Indonesian Journal of Electrical Engineering
Vol.12, No.2, February 2014, pp. 885 ~ 893
DOI: http://dx.doi.org/10.11591/telkomnika.v12i2.4009

A Multi-objective QoS Mechanism for Web Applications: Improved EFXCP

Hangxing Wu*[1,2], Xiaolong Yang[1], Min Zhang[1]
[1]School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing, China, 100083, Ph.: +86-01062332929
[2]Beijing Key Laboratory of Knowledge Engineering for Materials Science, Beijing, China, 100083,
*Corresponding author, e-mail: whx9711@163.com

Abstract

The research of QoS is concerned with multiple QoS requirements (multi-QR): queue delay, delay jitter and guaranteed throughput etc. As different QoS requirements have different attributes, it is hard to ensure multi-QR in one QoS mechanism. A common method to ensure multi-QR is normalizing multi-QR into one value range by a utility function and then finding the optimal solution of the utility function. However, although multi-QR are considered in such QoS mechanisms, in fact, no concrete QoS requirement can been strictly ensured, which may be embarrassing because web application usually requires one or more concrete QoS requirements to be satisfied strictly. Therefore, it is significant to design a real multi-objective QoS mechanism to satisfy different QoS requirements of various web applications. Our previous works proposed an efficient and fair explicit congestion control protocol (EFXCP) which can achieve excellent performance in terms of high link utilization, low queue delay, low delay jitter, etc. Because different web applications have different throughput requirements, to further satisfy throughput requirements of web applications, we extend EFXCP in this paper to implement an improved EFXCP (IEFXCP). Firstly, ToS field in IP header is utilized to classify different web applications, and then the relative fair bandwidth allocation is proposed between different types of web applications to preferentially ensure throughput requirements of high prior web applications. Therefore, IEFXCP can simultaneously satisfy multi-QR: queue delay, delay jitter and guaranteed throughput. The performance of IEFXCP is validated by extensive NS2 simulations over a wide range of network scenarios, the results show that IEFXCP is a real multi-objective QoS mechanism.

***Keywords**: multi-object, QoS, relative fairness*

1. Introduction

With the dramatic increase in the amount of data and the rapid development of extensive web applications, the research of QoS is always a hot spot. Because different web applications have different QoS requirements, the research of QoS focuses on various QoS requirements, such as queue delay, delay jitter, guaranteed throughput, etc. However, different QoS requirements have different attribute, it is hard to design a QoS mechanism which can satisfy multi-QR simultaneously. Hence many proposed QoS mechanisms aimed to ensure the most important QoS requirement according to QoS requirements of concrete web application, for example, the mechanisms in [1-4] aimed to satisfy low delay requirement, the mechanisms in [5-8] aimed to satisfy throughput requirement.

In order to design a mechanism to satisfy multi-QR, a common method is normalizing multi-QR into one value range by a utility function and then finding the optimal solution of the utility function, for example, the work in [9] proposed a multi-objective QoS mechanism for wireless sensor networks, which normalizing energy, delay and reliability into one value range by a function. The works in [10, 11] also adopted the same method. However, although multi-QR are considered in such QoS mechanisms, in fact, no concrete QoS requirement can been strictly ensured, which may be embarrassing because web application usually requires one or more concrete QoS requirements to be satisfied strictly. So, it is probable that such a QoS mechanism does not adapt to any web applications.

Received June 25, 2013; Revised August 22, 2013; Accepted September 18, 2013

This paper aim to address above challenge, our main goal is to propose a multi-objective QoS mechanism for various web applications which can satisfy simultaneously guaranteed throughput, low queue delay, low delay jitter, high link utilization.

In this paper, we implement a multi-objective QoS mechanism by extending EFXCP [12]. In our previous work, we have proposed EFXCP which can achieve low queue delay, low delay jitter, and high link utilization. But the fairness in EFXCP means absolute fair bandwidth allocation, every data flows will obtain the same amount of bottleneck bandwidth, which does not adapt to real web applications, because different web applications have different throughput requirements, for example: the multimedia application may requires strictly high throughput to ensure QoS, while the QoS of file transfer application is not sensitive to throughput and does not need strictly to ensure throughput. Therefore, in order to efficiently utilize limited web resources to provide better QoS to various web applications, relative fair bandwidth allocation is a good choice.

By extending EFXCP, We design a multi-objective QoS mechanism: improved EFXCP (IEFXCP). IEFXCP utilizes ToS field in IP header to classify different web applications, and then provides relative fair bandwidth allocation between different types of web applications to preferentially ensure the throughput requirement of web applications which need high transport rate. Therefore, IEFXCP can simultaneously satisfy multi-QR: queue delay, delay jitter and guaranteed throughput. The performance of IEFXCP is validated by extensive NS2 simulations over a wide range of network scenarios, the results show that IEFXCP is a real Multi-Objective QoS Mechanism.

The rest of the paper is organized as follows. We first give an overview of the EFXCP in Section 2. Then we propose IEFXCP algorithms in section 3. Next, we present simulation results in section 4. In the end, conclusion and next work are given.

2. An Overview of the EFXCP

EFXCP adds a congestion header to packet, as shown in Figure 1. Congestion header is used to communicate a flow's state to routers and feedback from the routers to the receivers. The field cur_thr_ is the sender's current throughput estimate and cur_rtt_ is the sender's current RTT estimate. These fields are filled by the sender and never modified in transit. The remaining field, exp_fair_thr_ and avg_rtt_ are the expected fair throughput and average RTT of the flows sharing the output link, which are calculated by router and initialized to zero by the sender. Routers along the path modify these two fields

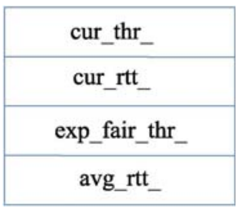

Figure 1. Congestion Header

EFXCP works as follows: before a packet is sent, Sender estimates its current throughput and RTT, and then communicates these values to the routers via the congestion header. Router monitors the degree of congestion in output link. If a link is overloaded, the router fills the exp_fair_thr_ field of congestion headers with a large number Num_, e.g. 10^{15}, which is far more than normal flows' throughput and means congestion occurs. Otherwise, based on the degree of congestion and all flow's state in congestion header, the router estimates the expected fair throughput of the flows sharing an output link and then fills the exp_fair_thr_ with the expected fair throughput. A more congested router along the path can further reduce the value in exp_fair_thr_ field by overwriting it. Note that a congestion header that exp_fair_thr_ field is filled with Num_ cannot be overwritten. Ultimately, when packet reaches the receiver, the value in exp_fair_thr_ field will be taking as the feedback to return to the sender by an ACK packet. The sender updates its cwnd to make its throughput converge to the feedback. Eventually, all flows will converge at the expected fair throughput.

The estimations of key parameters are given next, the more detailed descriptions can be found in [12].

2.1. Key Parameters at the Router

1) Average throughput

Average throughput $\bar{\gamma}$ is defined as the average throughput of the flows sharing an output link. $\bar{\gamma}$ is calculated only based on information in congestion header to avoid keeping per-flow state. $\bar{\gamma}$ is estimated as:

$$\bar{\gamma} = \frac{\sum r_i \times \frac{1}{n_i}}{\sum \frac{1}{n_i}} = \frac{\sum r_i \times \frac{s_i}{r_i \times t_s}}{\sum \frac{s_i}{r_i \times t_s}} = \frac{\sum s_i}{\sum \frac{s_i}{r_i}} \tag{1}$$

Where r_i is rate of flow i, which can be obtained from congestion header, n_i stands for the number of packets from flow i in the sample interval t_s, s_i is packet size of flow i which can be obtained from IP header.

2) The degree of congestion

The degree of congestion ρ_l of the output link in each sample interval is calculated as:

$$\rho_l = \frac{\lambda_l + k_q \times q_l}{C_l \times t_s} \tag{2}$$

Where λ_l is the amount of input traffic in the sample interval, q_l is the persistent queue length in t_s, k_q controls how fast the persistent queue drains, we set $k_q = 0.5$, C_l is the capacity of link l.

3) The expected fair throughput

When all flows sharing the output link send data with the expected fair throughput γ_o, the utilization of output link is 100% just right. Hence γ_o is estimated as:

$$\gamma_o = \frac{1 \times C_l \times t_s - k_q \times q_l}{\rho_l \times C_l \times t_s - k_q \times q_l} \times \bar{\gamma} \tag{3}$$

Then, γ_o will be filled in the exp_fair_thr_ field of congestion header and provided to the senders.

2.2. Key Algrithms at the Sender

When a sender i receives feedback γ_o, it will calculate target congestion window as:

$$target\_Cwnd\_ = \frac{\gamma_o \times rtt_i}{s_i} \tag{4}$$

Where rtt_i is current RTT of sender i. Then, sender i will adjust its congestion window $Cwnd$ as:

$$Cwnd \leftarrow \begin{cases} Cwnd + \alpha \times (target\_Cwnd - Cwnd) \times rtt / t_s; & \text{others.} \\ Cwnd \times \beta; & \text{congestion occurs} \end{cases} \tag{5}$$

Where $\beta = 0.875$.

3. Improved EFXCP

To satisfy different throughput requirements of web applications, in this section, we extend EFXCP to implement relative fair bandwidth allocation to preferentially ensure throughput of web applications which require high transport rate, such as Video on Demand, video conference, etc.

To implement relative fair bandwidth allocation, we should firstly classify web applications into different ranks. There are two ways which can be adopted to classify web applications. One is directly utilizing the ToS field in IP header, the other is adding a field in congestion header which is similar to the ToS field. The advantage of the former is simple to implement, the advantage of the latter is not interfering with standard ToS fields in IP header. In this paper, we adopted the former method to classify web applications, and then implemented improved EFXCP (IEFXCP) to provide relative fair bandwidth allocation.

IEFXCP works as follow: before a packet is sent, sender not only fills the congestion header, but also fills the ToS fields in IP header, which means sender should choose its rank of throughput requirement, a large value in ToS field means high rank. Then, expected fair throughput is estimated in Router and ToS field must be considered in the estimation. Ultimately, expected fair throughput is returned to senders, then, senders adjust their congestion window according to expected fair throughput and the value in their ToS field.

In IEFXCP, the value in ToS field is set to an integer, the value of the integer can be taken as the number of virtual data flow, for example, for a data flow with ToS field in which the value is 5, it can be taken as the aggregation of 5 virtual data flows. Relative fair bandwidth allocation in IEFXCP means that the ratio of allocated bandwidth between two data flows of which the value in ToS fields is 5, 1 respectively, will be 5 to 1.

To implement IEFXCP, we need to make some simple modifications to algorithms in sender and router of EFXCP.

3.1. Modification in Router

Instead of calculating per flow expected fair throughput in EFXCP, it is per virtual flow expected fair throughput that is calculating in IEFXCP. Considering the value in ToS field, we modify Equation (1) as:

$$\bar{\gamma} = \frac{\sum r_i \times \frac{1}{n_i}}{\sum(\frac{1}{n_i} \times ToS_i)} = \frac{\sum r_i \times \frac{s_i}{r_i \times t_s}}{\sum(\frac{s_i}{r_i \times t_s} \times ToS_i)} = \frac{\sum s_i}{\sum(\frac{s_i}{r_i} \times ToS_i)} \qquad (6)$$

Where $\bar{\gamma}$ in Equation (6) is per virtual flow average throughput. Then, the estimation of per virtual flow expected fair throughput γ_o is the same as the calculation in Equation (2). Ultimately, γ_o will be returned to senders.

3.2. Modifications in Sender

Firstly, before a packet is sent, sender not only fills the congestion header, but also fills the ToS fields in IP header.

Secondly, when receiving the feedback γ_o, sender will calculate its target congestion window as:

$$t\arg et\_Cwnd\_ = \frac{\gamma_o \times rtt_i}{s_i} \times ToS_i \qquad (7)$$

In Equation (7), $\gamma_o \times rtt_i / s_i$ means per virtual data flow expected congestion window, hence, the expected congestion window of a real data flow is ToS_i times of that of per virtual data flow.

4. Performance Evaluation

We make extensive ns2 simulations to evaluate the performance of IEFXCP. Our simulations cover capacities in [25Mbps, 4Gbps], RTT in [12ms, 1200ms], and arrival rates of short-lived, web-like flows in the range [1 s^{-1}, 1000 s^{-1}]. Simulations topology is shown in Figure 2, which is a typical single bottleneck link topology. We always use two-way traffic in all simulations. The bottleneck buffer size is set to the bandwidth-delay product, or two packets per-flow. The data packet size is 1000 bytes. All simulations are run long enough to ensure that the system has reached its steady state. All statistic data are collected in steady state.

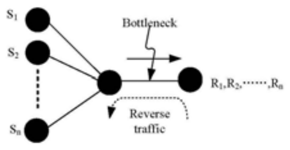

Figure 2. Single Bottleneck Link Topology

The basic setting is a 300Mbps link with 60ms RTT where the forward and reverse path each has 50 FTP flows. These forward 50 FTP flows are divided into 5 groups, the value in ToS field in data header is set to 1, 2, 4, 8, 10 respectively. We will choose randomly one flow in each group respectively to trace its throughput.

In all simulations, we will observe throughputs of chosen flows to evaluate relative fair bandwidth allocation, observe the average queue length of bottleneck link to evaluate queue delay, observe real time queue length of bottleneck link to evaluate queue delay jitter. Moreover, every QoS mechanism should ensure high utilization of bottleneck link while maintaining QoS requirements, so bottleneck link utilization is also be observed.

4.1. Evaluations under Stable Environment

Firstly, we evaluate IEFXCP under stable environment. We study the effect of varying the link capacity and the RTT on the performance of IEFXCP. we evaluate the impact of each network parameter in isolation while retaining the others as the basic setting.

1) Impact of bottleneck capacity

In this group of simulations, we vary the bottleneck capacity between [25Mbps, 4Gbps]. To depict results in all simulations within a figure, the unit of vertical axis in Figure 3(a) is set as the ratio of senders' practical throughput to the ideal throughput of flow with $ToS_i = 1$, which is adopted in next simulations. Flow_i in legend means the flow with $ToS_i = i$.

It can been seen from Figure 3(a) that relative fair bandwidth allocation is maintained excellently between flows with different values of ToS, which means throughputs of high prior web applications can be guaranteed. When capacities are 25Mbps and 50Mbps respectively, because the target congestion window of the flow with $ToS_i = 1$ is a small real number, but effective congestion window is an integer number of packets, the difference between the two values results in decreases in the throughput of flow with $ToS_i = 1$.

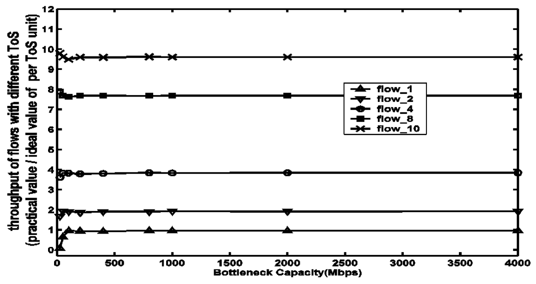

Figure 3(a). Throughput of Senders at Various Capacities

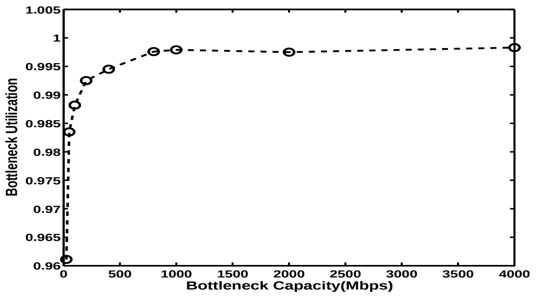

Figure 3(b). Link Utilization at Various Capacities

Figure 3(c) showed that average queue lengths in all simulations are almost below 1% buffer size, which means low queue delay. The real time queue lengths are shown in Figure 3(d), we can see that queue lengths are stable, the maximal variation range is from 1% to 3% buffer size when link capacity is 100Mbps. Hence, low queue delay jitter can be satisfied. Moreover, IEFXCP maintained high link utilizations in all simulations, as shown in Figure 3(b).

Figure 3(c). Average Queue Length at Various Capacities

Figure 3(d). Real Time Queue Length at Various Capacities

2) Impact of RTT

In this group of simulations, we vary the round-trip propagation delay between [12ms, 1200ms]. The results are shown from Figure 4(a) to Figure 4(e). We notice that the IEFXCP scheme always maintained excellent relative fair bandwidth allocation, high link utilization and low average queue lengths. When RTT is varied from 12ms to 800ms, the real time queue lengths are stable after the initial stage, the low queue delay jitter can be satisfied. However, because too large delay will make control system keep stable difficultly, it is hard for IEFXCP to keep queue length stable when RTT is larger than 1000 ms, as shown in Figure 4(e), but it is not a serious problem because the variation of the real time queue length is from 1% to 3% buffer size at the stable state.

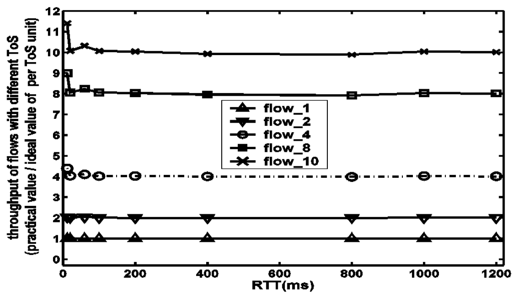

Figure 4(a). Throughput of Senders at Various RTTs

Figure 4(b). Utilization of Link at Various RTTs

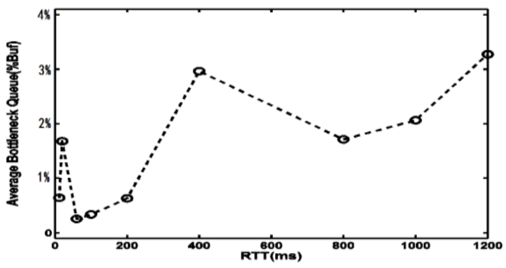

Figure 4(c). Average Queue Length at Various RTTs

Figure 4(d). Real Time Queue Length at Various RTTs

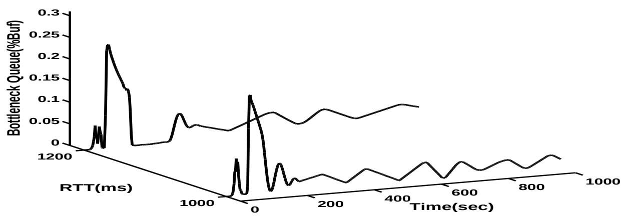

Figure 4(e). Real Time Queue Length at Large RTTs

4.2. Evaluations under Dynamic environment

Then, we evaluate IEFXCP under dynamic environment.

1) Impact of web-like traffic

In this group of simulations, we investigate IEFXCP's performance in the presence of variability and burstiness caused by the short-lived, web-like flows arrivals, which arrive according to a Poisson process and whose transfer size is derived from a Pareto distribution with an average of 30 packets (ns implementation with shape = 1.35), which complies with real web traffic. We can see from Figure 5(a) to Figure 5(d) that IEFXCP still maintains excellent relative bandwidth allocation, high utilization and low average queue length.

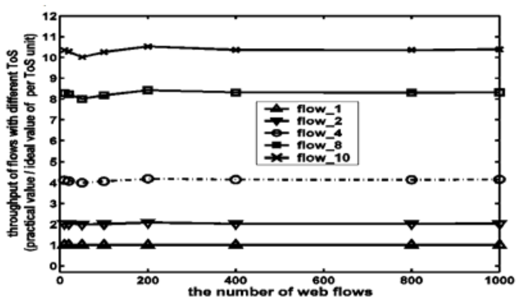

Figure 5(a). Throughput of Senders at Various Arrival Rates of of Web-Like Flow

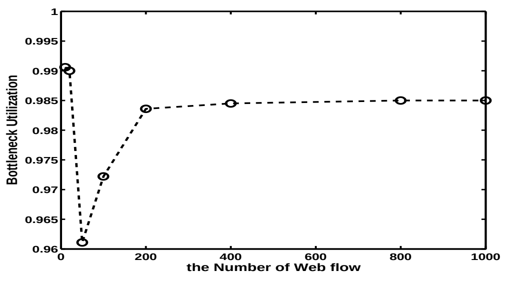

Figure 5(b). Link Utilization at Various Arrival Rates of Web-Like Flow

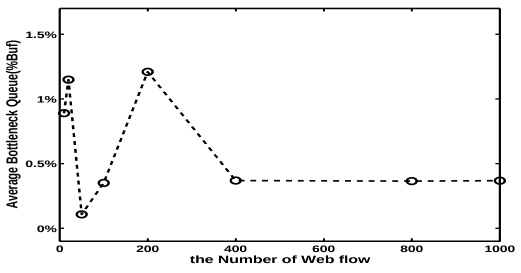

Figure 5(c). Average Queue Length at Various Arrival Rates of Web-Like Flow

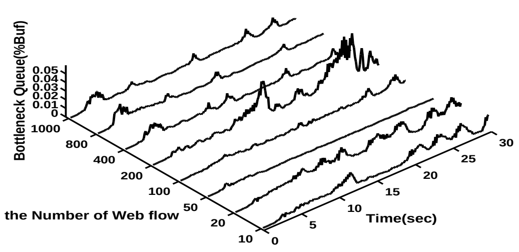

Figure 5(d). Real Time Queue Length at Various Arrival Rates of Web-Like Flow

An interesting phenomena is that when arrival rate of web-like flows is more than 200s^{-1}, the higher the arrival rate of web-like flows, the more stable the real time queue length are. The reason is that the aggregation data is more like a long-lived flow when the arrival rate is very high. In contrast to higher arrival rate, lower arrival rate is more likely to induce vibration of the queue length. But it is also not a serious problem because the maximal vibration of queue length is between 1% to 5% buffer size when arrival rate is 200s^{-1}. So, IEFXCP can maintain low queue delay jitter under most of the situations.

2) Impact of Sudden Change in Traffic Demand

Next, we illustrate how IEFXCP reacts to sudden changes in traffic demand. In the experiment, we keep basic setting unchanged. Moreover at t = 20s, we start 50 new forward FTP flows which are the same as the flows in basic setting, then stop them at t = 40s.

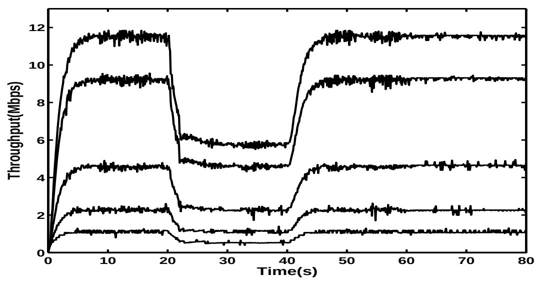

Figure 6(a). Throughput of Senders at Sudden Change

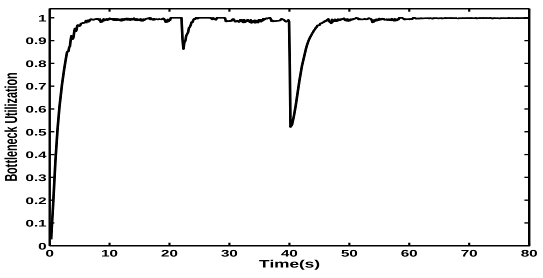

Figure 6(b). Real Time Utilization at Sudden Change

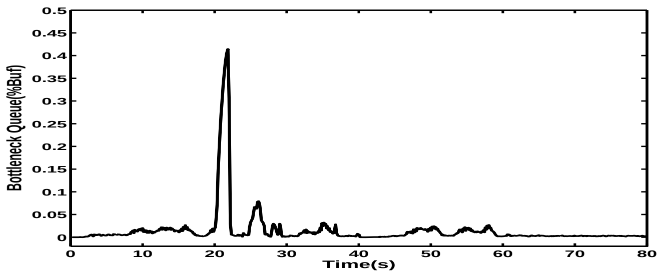

Figure 6(c). Real Time Queue Length at Sudden Change

The results are shown from Figure 6(a) to Figure 6(c). It can be seen that IEFXCP can adapt quickly to a sudden fluctuation in the traffic demand, and still maintains excellent relative bandwidth allocation, high utilization and a low and stable real time queue length. Although the queue length increases suddenly at the t=20s, which will lead to larger queue delay jitter, it can be drained quickly.

In summary, we can see from experiment results that IEFXCP can simultaneously maintain excellent relative fair bandwidth allocation, high utilization, low queue delay and low delay jitter. Therefore, IEFXCP can satisfy simultaneously multi-QR of various web applications, it is a real multi-objective QoS mechanism.

5. Conclusion and Next Work

To satisfy multi-QR of various web application, we proposed IEFXCP by extending EFXCP which can achieve high link utilization, low queue delay and low delay jitter. In IEFXCP, ToS field of IP header is utilized to classify web applications according to their throughput requirements, then IEFXCP provides relative fair bandwidth allocation to preferentially ensure the throughput requirements of web applications which need high transport rate. Extensive simulations in NS2 have shown that IEFXCP can simultaneously satisfy multi-QR: guaranteed throughput, low queue delay, low delay jitter and maintain high link utilization, it is a real multi-objective QoS mechanism.

However, IEFXCP in this paper assumes that all senders are honest, if a malicious sender fills a very large value in ToS fields to obtain oversubscribe throughput, the performance of IEFXCP will be damaged. Hence how to discriminate and restrict the malicious senders is our next work.

Acknowledgement

This work is supported in part by the Scientific Research Foundation of Central-affiliated University under Grant No. FRF-TP-12-090A, and the 2012 Ladder Plan Project of Beijing Key Laboratory of Knowledge Engineering for Materials Science, No. Z121101002812005.

References

[1] Jae-Han Jeon, Hee-Jung Byun, and Jong-Tae Lim. End-to-end delay guar-anteed proportional fair scheduling for wireless networks. *Communications Letters.* IEEE. 2011; 15(4): 401–403.
[2] K Karenos, V Kalogeraki. in RTSS. *Real-time traffic management in sensor networks*. Real-Time Systems Symposium. 2006: 422-434.
[3] Xiaodong Wang, Xiaorui Wang, Guoliang Xing, Yanjun Yao. *Dynamic duty cycle control for end-to-end delay guarantees in wireless sensor networks*. Quality of Service (IWQoS). 18th International Workshop. 2010.
[4] Xu Xiaole, Huang Wei, Chen Shengyong, Gao Lixin. Consensus of multi-agent systems with time delays and measurement noises. *Telkomnika Indonesia Journal of Electrical Engineering*. 2012; 10(6): 1370-1380.
[5] N Buchbinder, J Naor. *Fair online load balancing*. In SPAA Proceedings of the eighteenth annual ACM symposium on Parallelism in algorithms and architectures. 2006; 291–298.
[6] J Chou, B Lin. *Optimal multi-path routing and bandwidth allocation under utility max-min fairness*. Quality of Service. IWQoS. 17thInternational Workshop on. 2009; 1–9.
[7] Kong Yiquan. One method of cloud computing bandwidth allocation based on fairness. *Telkomnika Indonesia Journal of Electrical Engineering*. 2013: 954-959.
[8] JJ Jaramillo, R Srikant. Optimal scheduling for fair resource allocation in ad hoc networks with elastic and inelastic traffic. *Networking. IEEE/ACM Transactions on*. 2011; (99):1.
[9] E Jin, W Wang, M Palaniswami. Utility max-min fair flow control for multipath communication networks. In ICSPCS. 2007.
[10] Alwan H, Agarwal, A. *multi-objective qos routing for wireless sensor networks*. Computing, Networking and Communications (ICNC). 2013.
[11] D Yiltas, H Perros. Quality of service-based multi-domain routing under multiple quality of service metrics. *Communications. IET.* 2011; 5(3): 327–336.
[12] Chen, Wei-Neng; Zhang, Jun. An ant colony optimization approach to a grid workflow scheduling problem with various QoS requirements. *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*. 2009; 39(12): 9-43.
[13] Hangxing Wu, Fengyuan Ren, Dejun Mu, Xianwu Gong, An efficient and fair explicit congestion control protocol for high bandwidth-delay product networks. *Computer Communications.* 2009; 32: 1138-1147.