Poster: Flow-Based Buffer Management Scheme Suitable for Broadband Router

Yusuke Shinohara, Hideki Tode, Koso Murakami
Department of Information Networking,
Graduate School of Information Science and Technology, Osaka University,
Yamadaoka 1-5, Suita-shi, Osaka 565–0871, Japan
E-mail: {shinohara.yusuke, tode, murakami}@ist.osaka-u.ac.jp

Abstract— **In today's IP network, the Best-Effort service is provided mainly. However, it has trouble realizing QoS guarantee for each flow. Therefore, routers need to equip control mechanism which can provide high-speed and high-scalability. So far, we have proposed a new buffer management scheme, called DMFQ (Dual Metrics Fair Queueing), whose throughput is 6.8 Gbps. However, DMFQ will be required to adapt to 10 Gbps Ethernet. Moreover, DMFQ has not performed experiments in real networks yet and it is unclear its behavior on operations. In this paper, we extend DMFQ to achieve 10 Gbps throughput and verify its feasibility in real networks.**

I. Introduction

Recently, the realization of QoS guarantees becomes a very important technical issue on IP network. As one of the solutions, congestion controls in a router, called Active Queue Management (AQM), have received some attention.

So far, we have proposed a flow-based AQM, called Dual Metrics Fair Queueing (DMFQ)[1], [2], to improve fairness and to guarantee QoS. DMFQ discards arrival packets considering both the arrival rate and the flow succession time as instantaneous and historical conditions, respectively.

DMFQ calculates a packet discard probability by using the coefficient of the arrival rate, W_{rate}, and the one of the flow succession time, W_{con}. Simultaneously, W'_{rate} and W'_{con} are used as their average value within the class. DMFQ calculates W_{rate} by using Time Sliding Window (TSW)[3] and W_{con} by using the value of the counter C_S which indicates flow succession time as Eq. (1).

$$W_{rate} \leftarrow \frac{W_{rate} \cdot R_c + pkt\_size}{now - T_f + R_c}, \quad W_{con} = C_S \qquad (1)$$

where R_c is the time window, pkt_size is the arrival packet size, now is the arrival time of the packet, and T_f is the arrival time of the last packet. Applying these coefficients, DMFQ calculates the flow coefficient W and its average value W_{avr}.

$$W = W_{rate} \cdot W_{con}, \qquad W_{avr} = W'_{rate} \cdot W'_{con} \qquad (2)$$

The packet discard probability is calculated by comparing W with W_{avr}, which enables DMFQ to discard an arrival packet considering both the amount of instantaneous and historical condition of the flow. DMFQ calculates P by Eq. (3).

$$P = \begin{cases} 0 & (W_{rate} < W'_{rate}) \\ \max\left[0, 1 - W_{base}/W\right] & (W_{rate} \geq W'_{rate}) \end{cases} \qquad (3)$$

Note that W_{base} is the standard value for comparison and calculated as Eq. (4).

$$W_{base} = W_{avr} \cdot th/q_{avr} \qquad (4)$$

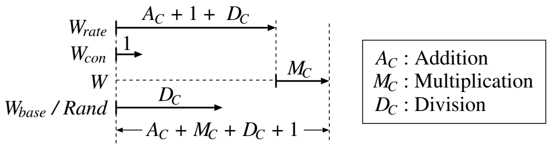

Fig. 1. The packet arrival process in DMFQ

where q_{avr} is the average queue length and th is the buffer threshold. By calculating P with not W_{avr} directly but W_{base}, DMFQ can use the buffer effectively.

To implement DMFQ with hardware, DMFQ has some technical issues and solutions.

The calculations related to the average values, W'_{rate}, W'_{con}, W_{avr}, W_{base} and q_{avr}, are carried out independently of the normal processing of arrival packets. Moreover, in the DMFQ, R_C is set so as to calculate W_{rate} by shift operation. Thus, DMFQ can calculate W_{rate} with not a multiplication operation but a shift operation that usually requires only 1 clock. In addition, DMFQ compares W with $W_{base}/Rand$ to decide packet discard, where $Rand$ means random number in [0,1]. This corresponds to Eq.(3), $W_{base}/Rand$ and W can be calculated in parallel.

It is assumed that the computing of addition, multiplication and division requires A_C(=1), M_C(=2) and D_C(=4) clock periods, respectively. With the above solutions, the operation time required is $(A_C + M_C + D_C + 1)$(=8) clocks, as shown in Fig. 1. This is very effective for high-speed operation.

DMFQ has already been implemented and synthesized by the Synopsys Design Compiler[4]. As a result, DMFQ realizes the maximum link speed of approximately 6.8 Gbps. However, DMFQ does not have the throughput adaptable to 10 Gbps Ethernet, which will spread in the future. Thus, DMFQ will be bottleneck in the future network.

II. The extended DMFQ

In this paper, we extend DMFQ to achieve 10 Gbps throughput by simplifying bottleneck process, calculation of W_{rate}. In the extended DMFQ, we introduce the calculation interval of W_{rate}, T_R. The extended DMFQ calculates W_{rate} every T_R and uses lastly calculated W_{rate} value (the latest value) at packet arrival. Thus, the extended DMFQ can read out W_{rate} as the calculated value and can make the calculation process easier. Moreover, not only T_R but also $service\_rate$ and R_C are set-up parameters. Thus, the extended DMFQ calculates W'_{rate} with these value as a shift operation. It is efficient to alleviate the process on updating W_{rate}.

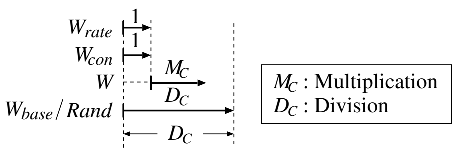

Fig. 2. The packet arrival process in the extended DMFQ

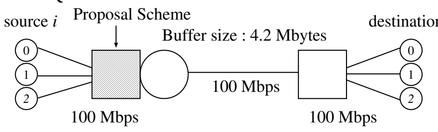

Fig. 3. Experiment environment

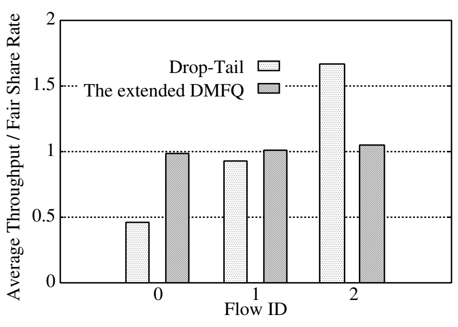

Fig. 4. Normalized throughput (Scenario1)

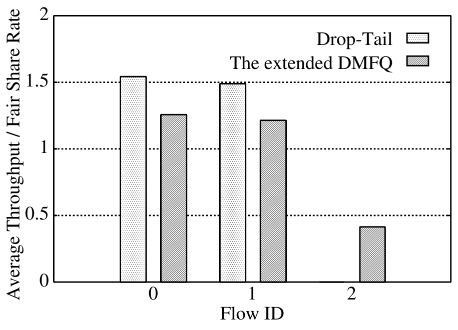

Fig. 5. Normalized throughput (Scenario2)

Concretely, the extended DMFQ calculates W_{rate} every T_R as Eq. (5), and calculates W'_{rate} as Eq.(6).

$$W_{rate} \leftarrow W_{rate} \times R_c + total\_pkt\_size \tag{5}$$

$$W'_{rate} = service\_rate \times (R_c + T_R)/Fact \tag{6}$$

where $total\_packet\_size$ is the total size of arrival packets of each flow during T_R interval. With the above calculation, the extended DMFQ can achieve both the quick process at packet arrival and the performance equivalent to DMFQ.

The extended DMFQ should calculate exact W_{rate} quickly. However, we assume that multiple management tables are implemented in the memory, and the extended DMFQ can access the element of only one flow per access of each table. We introduce "slot" which is the fixed size division of T_R. The extended DMFQ calculates W_{rate} for one flow in one slot and always calculates W_{rate} for the flow at the corresponding slot location. Then, W_{rate} for each flow is calculated every T_R exactly, because the extended DMFQ can always calculate W_{rate} of the flow with T_R. To calculate more exact W_{rate}, T_R should be set shorter value. However, the extended DMFQ can not calculate W_{rate} of many flows in short T_R. Thus, the extended DMFQ calculates W_{rate} for fixed number of flows in one slot with parallel arithmetic units and memories. This leads to both exact W_{rate} calculation and the management of many flows.

According to the above solution, the extended DMFQ can perform packet arrival process for D_C(=4) clocks as shown in Fig.2. Moreover, the extended DMFQ achieves both the exact derivation of W_{rate} and the management of many flows.

We implemented and synthesized the extended DMFQ. As a result, the extended DMFQ realized the maximum link speed of approximately 10.2 Gbps. This is adaptable to 10 Gbps Ethernet, which will spread in the future.

III. Experimental evaluation in the real network

In this section, we evaluate the performance of the extended DMFQ experimentally in the real network. We use the extended DMFQ implemented on FPGA[5] chip as the experimental production.

In the network shown in Fig. 3, there are 3 source-destination pairs. We use 100BASE-TX as each link. Each source generates UDP or TCP traffic with Greedy Model. UDP packet size is set to 1.5 Kbytes (constant). We use File Transfer Protocol (FTP) to generate TCP traffic. The experiment results are the sample mean of 10 experiments whose experiment time is 60 s.

We compare the performance of the extended DMFQ with Drop-Tail. Parameters are set as follows: $R_C = 100$ [ms], $T_C = 5.0$ [s], $C_{Din} = 2$, $T_R = 2^{-10}$ [s]. We evaluate the performance in the following scenarios.

<**Scenario1**> : The source i ($i = 0, \cdots, 2$) generate traffic at Constant Bit Rate (CBR) of 40, 70, 100 Mbps, respectively.

<**Scenario2**> : The source i ($i = 0, 1$) generate UDP traffic at CBR of 100 Mbps. The source2 generates TCP traffic.

Figure 4 shows normalized throughput in Scenario1. From Fig.4, Drop-Tail can not assign the fair share rate. In contrast, the extended DMFQ can assign the bandwidth fairly.

Figure 5 shows normalized throughput in Scenario2. In the case of Drop-Tail, Flow0 and Flow1 which generate burst UDP traffic occupy all bandwidth. On the other hand, the extended DMFQ can assign some bandwidth to Flow2. The extended DMFQ still assigns higher throughput to Flow0 and Flow1 than fair share rate. Because, when there are no packets belonging to Flow2 (TCP flow) in the buffer, the extended DMFQ calculates fair share rate as 50 Mbps for Flow0 and Flow1, and works for the rate.

The above results show that the extended DMFQ can provide the fairness while achieving the high throughput.

IV. Conclusion

In this paper, we extended DMFQ, to achieve 10 Gbps by eliminating its bottleneck process. This leads to the adaptation to future broadband networks e.g., 10 Gbps Ethernet. Moreover, we confirmed its behavior and effectiveness from the viewpoint of fairness in the real network. Future work includes that the buffer management scheme considering not only the TCP layer but higher layer.

References

[1] N. Yamagaki, H. Tode and K. Murakami: "Dual Metrics Fair Queueing: Improving Fairness and File Transfer Time" Proc. of The 2005 International Symposium on Applications and the Internet (SAINT2005), pp.150-156, Feb. 2005.
[2] Y. Shinohara, N. Yamagaki, H. Tode and K. Murakami: "Queue Management Scheme Stabilizing Buffer Utilization in the IP Router" Proc. of 4th International Conference on Networking (ICN'05), partII, pp.307-317, Apr. 2005.
[3] D. D. Clark and W. Fang,: "Explicit Allocation of Best-Effort Packet Delivery Service," IEEE/ACM Trans. Networking, vol.6, no.4, pp.362-373, Aug. 1998.
[4] Design Compiler, http://www.synopsys.com/, Synopsys, Inc.
[5] Virtex-II FPGA, http://www.xilinx.com/, Xilinx, Inc.