SCIENCE CHINA
Information Sciences

• LETTER •

June 2016, Vol. 59 069301:1–069301:3
doi: 10.1007/s11432-016-5538-4

# Flow-based queue management for fairness control in high-bandwidth network

Wonjun CHOI[1], Ramneek SEKHON[1] & Woojin SEOK[1,2*]

[1]*Department of Science and Technology Information Science, Korea University of Science & Technology, Daejeon* 305-350*, Republic of Korea;*
[2]*Department of Advanced KREONET Application Support, Korea Institute of Science and Technology Information, Daejeon* 305-806*, Republic of Korea*

Received November 13, 2015; accepted January 8, 2016; published online February 17, 2016

**Citation** Choi W J, Sekhon R, Seok W J. Flow-based queue management for fairness control in high-bandwidth network. Sci China Inf Sci, 2016, 59(6): 069301, doi: 10.1007/s11432-016-5538-4

**Dear editor,**

With the development of information equipment and growth of broadband network, the progress of Internet has changed from the network with modem speed in the past to the high-bandwidth. The essential problem in the use of the network in a variety of fields is to use the network resource efficiently distributing the limited bandwidth [1]. If congestion occurs in high-bandwidth network, slow rate flow and fast rate flow will experience a lot of packet loss. When the fast flow occupies the current bandwidth, the available bandwidth for slow flow can be reduced. Therefore, it does not use the bandwidth range properly and results in the reduction of throughput. If fast flow adjusts the transmission rate for slow flow, fast flow cannot use the given bandwidth 100%. There are many ideas for efficient use of the bandwidth of the network, the examples of which are as follows. Comparison for the performance of TCP protocol and Parallel TCP study for improving TCP fairness and improvement in the loss of high-bandwidth [1]. Cubic TCP which is famous for Linux system is used for fairness evaluation at different delay time and RTT fairness [1]. The enhanced TCP congestion control is proposed to solve bandwidth utilization problem. It is from sharing bandwidth connection between high-speed TCP and traditional TCP. The multipath congestion control based on AIMD (additive increase and multiplicative decrease) is also proposed to share the bandwidth [2]. If the number of flow is high, interference between flows will increase and delayed ACK also increases, so RTT estimation will be uncertain. In order to address this problem, adaptive TCP congestion control is proposed [3] and also there are new network traffic algorithm [4] and traffic detection algorithm [5]. In this paper, we will consider how to maintain fairness by controlling the flow in the bottleneck of the network by efficiently using the structure of the intermediate queue. The proposed method will handle fast rate flow and slow rate flow separately to improve total throughput. It compares IPv4 layer of the arrived packet in the queue in order to separate the packets of a slow flow. When congestion occurs in the bottleneck node, the proportion of fast rate of packet flow and slow rate of packet flow will be set to 1:1.

Figure 1(a) shows the proposed algorithm in the intermediate queue. It is based on RED (random early detection) algorithm which is well known for common basic queue mechanism. If the buffer is

* Corresponding author (email: wjseok@kisti.re.kr)

The authors declare that they have no conflict of interest.

 

**Algorithm 1**

**Ensure:** [EnQueue] Arriving packet is not dummy
**for** each arriving packets **do**
  calculate the average of queue size
  avg = $(1 - w_{\mathrm{q}})$avg + $w_{\mathrm{q}}q$
  **if** the arrived packet is fast rate flow **then**
    assign the packet to the front of queue list
  **else if** the arrived packet is slow rate flow **then**
    assign the packet to the back of queue list
  **end if**
  **if** avg $\geqslant \min_{\mathrm{th}}$ **then**
    **if** avg $< \max_{\mathrm{th}}$ **then**
      calculate probablity $P_a$
      $p_b = \max_p(\mathrm{avg} - \min_{\mathrm{th}}/(\max_{\mathrm{th}} - \min_{\mathrm{th}})$
      $p_a = p_b/(1 - \mathrm{count}\ p_b)$
    **end if**
  **else if** avg $> \max_{\mathrm{th}}$ **then**
    mark the arriving packet
  **end if**
**end for**
Where:
- avg: average queue size
- count: packets since last marked packet
- $w_{\mathrm{q}}$: queue weight
- $\min_{\mathrm{th}}$: minimum threshold for queue
- $\max_{\mathrm{th}}$: maximum threshold for queue
- $\max_p$: maximum value for $p_a$
- $p_a$: current packet-marking probability
- $q$: current queue size

**Ensure:** [DeQueue] The proportion between fast rate flow and slow rate flow= 1:1
**if** queue list is not empty and front = true **then**
  pop the front packet of queue list
  front = false
**else**
  pop the back packet of queue list
  front = true
**end if**

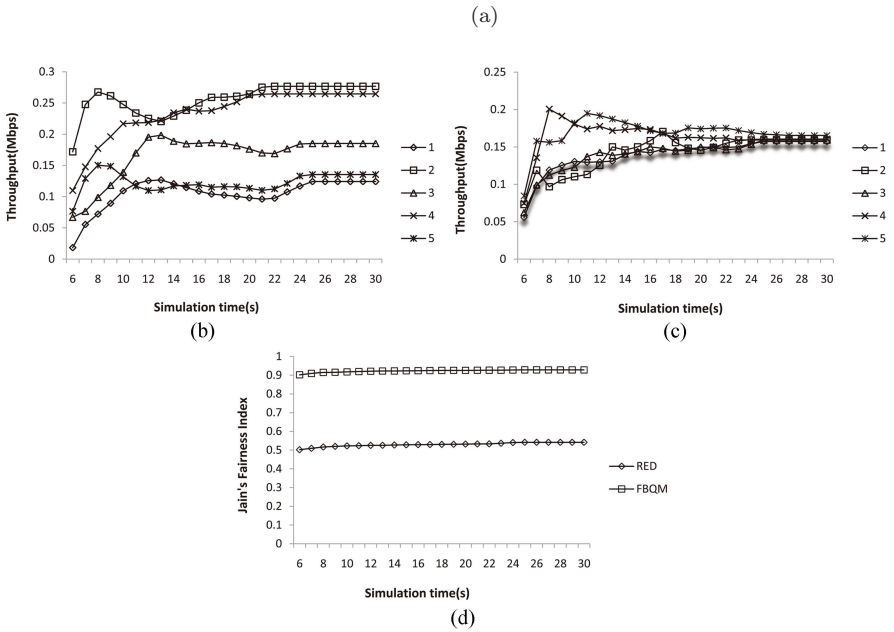


**Figure 1** (a) Algorithm EnQueue and DeQueue; (b) throughput for each flow with RED; (c) throughput for each flow with FBQM; (d) the comparison of Jain's fairness index.

almost empty, all incoming packets are accepted. As the queue grows, the probability for dropping an incoming packet grows too. When the buffer is full, the probability has reached 1 and all incoming packets are dropped. RED is more fair than tail drop mechanism. When the packet arrives in the bottleneck node, the algorithm checks the packet type whether it is fast or slow flow.

If the arrived packet is fast flow packet, it will be assigned to the front of the queue. If the arrived packet is slow flow packet, it will be assigned in the back of the queue. Because packets are stacked at the incoming time, the assignment mechanism will be done at the poping packet time. Queue utilization is high when the average size of the queue is between the minimum threshold and the maximum threshold. Therefore at this time, the proportion for fast flow and slow flow to pop the packet stacked in the queue is 1:1. This allows the fairness in the congested bandwidth. We used NS simulator for performance test of the proposed method. The packet size is 1040 byte. The available packet number in bottleneck queue is 100 packets. The amount of packet size to transmit is set randomly depending on the given simulation time. The simulation time is set to 30 s. The topology is dumbbell shape and the link rate is 100 Mbps. The transmission rate of each flow is set to 100 Mbps. The intermediate bottleneck interval is set to 1 Mbps, and the delay is 50 ms. Transmission layer of each flow is set to TCP protocol.

Random Early Detection algorithm adapts statistical way when the queue utilization is high. We compared FBQM (flow-based queue management) with RED type. The minimum threshold is set to 50 packet count in RED type, and the maximum packet count for threshold is set to 80 packets. The start time of 1 flow is 1 s and the interval of each flow is 1 s so the start time of 5 flow is 5 s. Figure 1(b) shows the throughput for each flow with RED as previous method in the intermediate bottleneck node's queue. We can find that the throughputs are different at congested time. Each flow starts to check current bandwidth after its start time. Although the first flow (which is started 1 s) is faster flow than the other flows, it is affected by the other slow flows (which is started later). The slow flow can also be affected by fast flow at congested time. Therefore, the throughput of fast flow can be slow in the queue of intermediate node. Figure 1(c) shows throughput for each flow with FBQM. When the network condition is bad in the bottleneck, if we run the proposed method, we can get more fair throughput than RED type. The throughput of each flow at start time is different, however, it converges to 0.16 at 25 s. This is because when the queue utilization is high, we change the proportion rate for fast flow and slow flow at poping packet time. Figure 1(d) describes the comparison of Jain's fairness index for each method. Fast flow is affected by slow flow when network condition is bad. The queue structure of current network is serial type, the fast flow packet and the slow flow packet are mixed in the bottleneck queue. Therefore both throughputs can be low or only fast flow can occupy the bandwidth. It causes irregular fairness. The proposed method tried to solve the problem by changing the proportion between fast flow and slow flow at congested time in bottleneck queue. In this way, the proposed queue mechanism can solve the problem in congestion state of the network. The enhanced FBQM mechanism with various situation will be the next research.

**Acknowledgements** This work was in part supported by Korea Institute of Science and Technology Information (KISTI) (User-based research network services) (Grant No. K-14-L01-C03-S03).

## References

1 Kozu T, Akiyama Y, Yamaguchi S. Improving RTT Fairness on CUBIC TCP. In: the 1st International Symposium on Computing and Networking (CANDAR), Matsuyama, 2013. 162–167
2 Zhou D Z, Wei S, Yu C. A study of fair bandwidth sharing with AIMD-based multipath congestion control. IEEE Wirel Commun Lett, 2013, 2: 299–302
3 Nemoto Y, Ogura K, Katto J. An adaptive TCP congestion control having RTT-fairness and inter-protocol friendliness. In: IEEE Consumer Communications and Networking Conference (CCNC), Las Vegas, 2013. 178–183
4 Jiang D D, Xu Z Z, Xu H W. A novel hybrid prediction algorithm to network traffic. Ann Telecommun, 2015, 70: 427–439
5 Jiang D D, Xu Z Z, Zhang P, et al. A transform domain-based anomaly detection approach to network-wide traffic. J Netw Comput Appl, 2014, 40: 292–306