Appl. Sci. **2015**, *5*, 122-135; doi:10.3390/app5020122

OPEN ACCESS

applied sciences

ISSN 2076-3417

www.mdpi.com/journal/applsci

Article

Adaptive Filtering Queueing for Improving Fairness

Jui-Pin Yang

Department of Information Technology and Communication, Shih-Chien University, Kaohsiung 84550, Taiwan; E-Mail: juipinyang@gmail.com; Tel.: +886-9-7535-7705

Academic Editor: Christos Verikoukis

Received: 20 March 2015 / Accepted: 27 May 2015 / Published: 5 June 2015

Abstract: In this paper, we propose a scalable and efficient Active Queue Management (AQM) scheme to provide fair bandwidth sharing when traffic is congested dubbed Adaptive Filtering Queueing (AFQ). First, AFQ identifies the filtering level of an arriving packet by comparing it with a flow label selected at random from the first level to an estimated level in the filtering level table. Based on the accepted traffic estimation and the previous fair filtering level, AFQ updates the fair filtering level. Next, AFQ uses a simple packet-dropping algorithm to determine whether arriving packets are accepted or discarded. To enhance AFQ's feasibility in high-speed networks, we propose a two-layer mapping mechanism to effectively simplify the packet comparison operations. Simulation results demonstrate that AFQ achieves optimal fairness when compared with Rotating Preference Queues (RPQ), Core-Stateless Fair Queueing (CSFQ), CHOose and Keep for responsive flows, CHOose and Kill for unresponsive flows (CHOKe) and First-In First-Out (FIFO) schemes under a variety of traffic conditions.

Keywords: adaptive filtering; active queue management; fairness; bandwidth

1. Introduction

Random Early Detection (RED) detects incipient congestion by computing the average queue size [1]. When the average queue size exceeds a threshold, RED drops or marks each arriving packet with a probability, where the probability is a function of the average queue size. RED not only keeps queuing delays low but also maintains high overall throughput because it can prevent current connections from global synchronization. RED should cooperate with transport-layer protocols capable of congestion control, such as TCP; unfortunately, it currently does not. Without congestion control, RED has poor fairness, especially for heterogeneous traffic environments. To improve the fairness of RED, a CHOose and Keep for responsive flows, CHOose and Kill for unresponsive flows (CHOKe) scheme was proposed [2]. When a packet arrives at a router, CHOKe compares it with a packet at random from the buffer. If both packets come from the same flow, both are discarded at the same time; otherwise, the arriving packet may be discarded with a probability depending on the current degree of congestion. By using additional discrimination on flows with heavy traffic, CHOKe demonstrates improved fairness. XCHOKe is a revised version of CHOKe [3]. XCHOKe maintains a lookup table to record CHOKe hits. Accordingly, it further identifies possible malicious flows. If a flow has many CHOKe hits, this flow has a higher probability of being identified as a malicious flow. Therefore, XCHOKe applies a higher dropping probability to the arriving packets of this flow. Although XCHOKe achieves better fairness than CHOKe and RED, it lacks scalability, making XCHOKe too complicated to be deployed in high-speed networks.

Considering the trade-off between scalability and fairness, Core-Stateless Fair Queueing (CSFQ) [4] and Rainbow Fair Queueing (RFQ) [5] were proposed. In particular, all routers in CSFQ are classified as edge or core routers. Edge routers need to maintain per-flow state because they have to estimate the flow rate of each arriving packet. Next, the information is inserted into the corresponding packet headers. Core routers estimate the fair share rate and then use a simple dropping algorithm to determine whether an arriving packet is accepted or discarded. RFQ is similar to CSFQ but with one significant difference: the state information that is inserted into the packet headers is the color layers, not the explicit flow rate. The operations of the core routers are further simplified, and the application can assign differentiated preferences to certain packets. The fairness of both core-stateless schemes could be degraded along with the increasing number of traversing nodes. Rotating Preference Queuing (RPQ) consists of a set of FIFO output queues that are dynamically rotated [6]. RPQ dispatches qualified arriving packets to adequate output queues based on packet distribution and preferences. RPQ has excellent fairness, but it needs a large buffer size, which means that RPQ may have a high implementation cost and result in a large queueing delay. Compared with RPQ, CSFQ, CHOKe and FIFO, the proposed AFQ scheme in this paper is scalable because it is easy to implement. Furthermore, AFQ is efficient because it provides approximately perfect fairness under various traffic conditions.

An interesting research question is whether a scheme can achieve fairness without requiring per-flow state. Providing fairness is important because it also contributes to congestion avoidance. In this paper, the objective is to achieve fair bandwidth sharing with a simple and scalable AFQ approach. The rest of the paper is organized as follows: Section II reviews the related work; Section III describes the details of the AFQ scheme with a two-layer mapping mechanism that can simplify the packet comparisons; Section IV presents the simulation results that demonstrate the fairness of different schemes under various network topologies and traffic conditions; and Section V presents our conclusions.

2. Related Work

In general, two types of schemes are used to address fairness among competing flows: AQM and packet scheduling. Compared with packet scheduling, AQM has attracted more attention due to its simplicity and efficacy. Moreover, AQM can enhance the performance of congestion control algorithms. Deficit Round Robin (DRR) is a packet scheduler that can achieve approximately perfect fairness [7]. DRR allocates a virtual queue dedicated to each active flow, which accommodates its arriving packets. When the packets are enqueued into particular queues, they will be served in a round-robin fashion according to the available quantum sizes. DRR needs to maintain per-flow state. In addition, DRR must work with a pushout (PO) buffer management scheme that avoids buffer shortages on certain flows [8]. When a packet arrives at a router with a full buffer, PO will push out one or more residing packets from the longest virtual queue. In this manner, PO can make room for the new arrival. Otherwise, the arriving packet will be accepted without any constraints. PO can make buffer utilization high and packet loss low under various traffic conditions. However, PO has two main drawbacks. First, it has to find the longest virtual queue out when arriving packets encounter a full buffer. Second, it has to execute frequent pushout operations under congested traffic conditions. As a result, it is questionable whether DRR should be implemented due to the abundance of active flows in routers [9].

DRR may suffer from a large delay and jitter due to a long cycle of round robin operations, so several variants have been developed to address these issues [10–12]. A customized deficit round robin (CDRR) takes care of real-time flows by adding a new queue to schedule real-time traffic just prior to the deadline [10]. However, the extra queue increases the delay for non-real-time traffic, particularly when the traffic load is heavy. Moreover, assigning weights to the queues may enlarge overall unfairness, especially for non-real-time traffic. Another variant, fuzzy-based Adaptive Deficit Round Robin (FADRR), uses expert systems based on fuzzy logic to adjust the weights of service queues for real-time and non-real-time traffic [11]. Additionally, this scheme sacrifices fairness because it favors real-time traffic that may result in bandwidth starvation for non-real-time traffic.

RED is a well-known AQM scheme employing a single FIFO buffer to accommodate arriving packets from all active flows [1]. The arriving packets encounter different drop probabilities according to average queue sizes and other parameters. RED discards packets before the buffer is full, so it can prevent the TCP connections from global synchronization. Unfortunately, RED is unable to provide fairness, especially for heterogeneous traffic. Based on RED, several variants have been proposed that enhance the fairness or robustness of RED parameters [13–18]. Self-Configuring RED changes dropping probabilities according to the variations in average queue sizes [13]. If the average queue size oscillates around the minimum threshold, then the current dropping probability is too high. On the other hand, if it oscillates around maximum threshold, then the current dropping probability is too low. Based on the dynamics of queue sizes, the scheme adjusts packet dropping probabilities that reduce packet loss while maintaining high link utilizations. Another RED variant, weighted Destination-Based Fair Dropping (wDBFD), only needs a single queue, and packets are probabilistically dropped before they are enqueued instead of tail dropping. Furthermore, by adding weights to the drop probabilities of different destination stations, this scheme realizes destination differentiation. Additionally, wDBFD is more robust in terms of fairness when subjected to different packet arriving rates. The idea of wDBFD is similar to that of the RED, but it relies not only on past measurements of queue size but also on recent observed rates of flows. By using this additional information, wDBFD improves RED’s fairness. However, this scheme also increases the complexity.

In CSFQ [4], an edge router has to maintain per-flow state and is in charge of state (flow arrive rate) insertion into packet headers. Whenever a core router receives a packet, CSFQ has to estimate the fair share rate and uses a simple probabilistic model to accept or discard the new arrival. CSFQ achieves reasonable fairness; moreover, it pushes complexity toward the edge routers, which simplifies the sophisticated implementation in the core routers. In general, the number of active flows in core routers is relatively larger than in edge routers. Therefore, CSFQ can be deployed in network environments consisting of high-speed core routers and moderate-speed edge routers. The architecture of RFQ is similar to CSFQ, which mainly consists of packet coloring and buffer management [5]. RFQ transfers the flow arriving rate into a set of layers, with a globally consistent color per layer. Next, the edge routers insert the color into packet headers. When a packet arrives at a core router, the arrival will be discarded only if its color level is over the color threshold. The color threshold dynamically changes in accordance with traffic variations. Compared with CSFQ, RFQ has approximate fairness, but it only carries a simple color level rather than an explicit flow arriving rate. Furthermore, it wards off exponential averaging estimation when a packet is generated. In summary, both CSFQ and RFQ schemes classify the routers as edge or core routers, and only the edge routers maintain per-flow state.

The fairness of RPQ tends to approach that of DRR, and it outperforms several schemes, such as CSFQ, DDE, CHOKe and FIFO [6]. In addition, RPQ has a complexity of O(1) and is simple to implement in high-speed networks. However, RPQ has an expensive implementation cost and a large queue delay. Currently, several TCP variants are adopted by end users, and heterogeneous congestion control schemes have thus become a characteristic of newly emerging networks. In contrast to pure TCP connections, the fairness of several well-known AQM schemes, such as RED and CHOKe, among heterogeneous TCP connections is discussed [19]. We do not consider the effect of TCP variants here, but it is an interesting topic as an extension of AFQ applications.

3. Adaptive Filtering Queueing

In Figure 1, we depict four main components of AFQ, including accepted traffic estimation, fair filtering level estimation, a filtering level table and a packet-dropping algorithm. At the end of this section, we propose a mechanism that permits the scalability of AFQ. If the mean arriving rate of a flow is larger than the max-min fair rate, such a flow is defined as an aggressive flow; otherwise, this flow is defined as a non-aggressive flow. In addition, the flow label is composed of a pair of IP source-destination addresses related to a flow. When a packet arrives at a router, AFQ identifies the filtering level of the arriving packet by randomly comparing with a flow label from the first level to an estimated level in the filtering level table. Based on the accepted traffic estimation and the previous fair filtering level, AFQ updates estimates of the fair filtering level. Finally, AFQ uses a simple packet-dropping algorithm to determine whether the packet is qualified to be accepted or discarded according to estimates of the fair filtering level and the filtering level of the arrival.

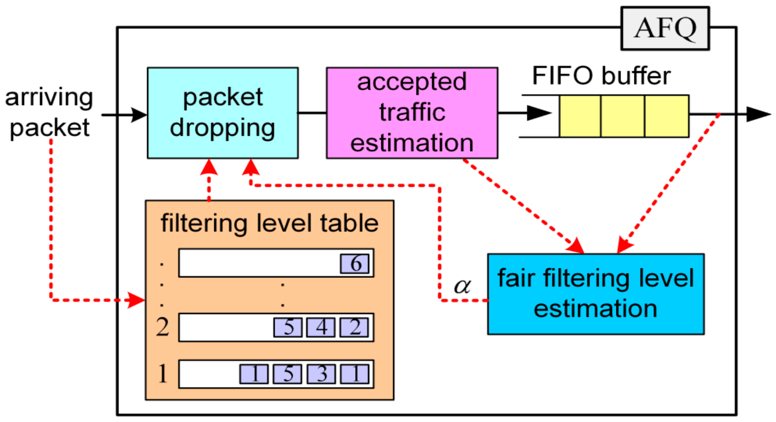

Figure 1. Adaptive Filtering Queueing (AFQ) scheme.

In a router, AFQ estimates accepted traffic $\hat{A}_{new}$ passing through the packet dropping component in a time interval using use Equation (1). The unit of $\hat{A}_{new}$ is bits, and T_d denotes the length of such a time interval.

$$\hat{A}_{new} = k_a A_{now} + (1 - k_a)\hat{A}_{old} \tag{1}$$

$\hat{A}_{old}$ is the value prior to the updating of $\hat{A}_{new}$, and A_{now} is the amount of traffic accepted in the current time interval. In addition, k_a is a coefficient used to balance the short-term and long-term estimates of $\hat{A}_{new}$. The estimation of the fair filtering level $\hat{\alpha}$ is proportional to $CT_d/\hat{A}_{new}$, as described in Equation (2). Similarly, $\hat{\alpha}_{old}$ is the value before the updating of $\hat{\alpha}_{new}$, and C denotes the router's link capacity (bits per second). If $\hat{A}_{new}$ is smaller than the idea output traffic CT_d, then $\hat{\alpha}_{new}$ increases. A larger $\hat{\alpha}_{new}$ will allow more arriving packets to be accepted. By observing the dynamics of accepted traffic, AFQ can produce a precise estimate of the fair filtering level.

$$\hat{\alpha}_{new} = \hat{\alpha}_{old} CT_d/\hat{A}_{new} \tag{2}$$

AFQ compares the flow label of each arriving packet with a flow label in a filtering level table at random from filtering level 1 to $\hat{\alpha}_{new}$ until there is a hit. The filtering level table consists of multiple filtering levels, and each filtering level only keeps flow labels, not whole packets, whose utility is to discriminate the dropping probabilities of arriving packets. If both own the same flow label (*i.e.*, are coming from the same flow), then a hit occurs. In AFQ, multiple filtering levels work as a hierarchical filter that filters out the unqualified arriving packets. AFQ may have insufficient discriminability because of traffic dynamics, which leads to fairness degradation; hence, we should enlarge the $\hat{\alpha}_{new}$. We readjust $\hat{\alpha}_{new}$ by k_b times the $\hat{\alpha}_{new}$ in Equation (3), denoted by $\hat{F}_{max}$. k_b is a coefficient whose function is to enhance the discriminability of AFQ. Finally, the range of flow label comparisons is altered from filtering level 1 to $\hat{F}_{max}$. In other words, $\hat{F}_{max}$ contributes to the realization of sufficient discriminability; the principle to determine whether arriving packets are accepted or dropped still depends on $\hat{\alpha}_{new}$. As a result, the complexity of AFQ is of O($\hat{F}_{max}$).

$$\hat{F}_{max} = k_b \hat{\alpha}_{new}, k_b \geq 1 \tag{3}$$

Assuming that packet i has a first hit at filtering level m_i, we use Equation (4) to calculate filtering level v_i. A larger m_i implies that the flow of packet i has fewer residing packets in the buffer, which means that the flow label of packet i will be enrolled into filtering level v_i. There are two supplementary rules. First, once packet i encounters an empty filtering level at v_{emp}, then $v_i = v_{emp}$. Second, if there is no hit until $\hat{F}_{max}$, then $v_i = \hat{F}_{max}$.

$$v_i = \begin{cases} m_i - 1 & 2 < m_i \leq \hat{F}_{max} \\ 1 & 1 \leq m_i \leq 2 \end{cases} \tag{4}$$

After determining the filtering level of packet i, AFQ needs to decide its location in filtering level v_i according to Equation (5), denoted by $index_i$. A circular replacement method is used to update the flow labels in each filtering level. Furthermore, we assume that each filtering level has the same capacity, denoted by L. AFQ updates the filtering level table according to traffic conditions. When the traffic is heavier, AFQ updates the filtering level table with a higher frequency.

$$index_i = max((index_i + 1)\, mod\, (L + 1), 1) \tag{5}$$

AFQ utilizes a simple packet-dropping algorithm to decide the treatment of packet i, where $prob_i$ denotes the dropping probability of packet i. Once packet i is accepted, it will be enqueued into the FIFO buffer; otherwise, it will be discarded immediately. Thus, only the arriving packets whose filtering levels are equal to or larger than the fair filtering level can be admitted to enter the buffer.

$$prob_i = \min(floor(\hat{\alpha}_{new}/m_i), 1) \tag{6}$$

We propose a mechanism to add to AFQ to simplify the packet comparisons (*i.e.*, the flow labels) while reducing memory consumption. In this design, the routers are classified as edge or core routers, similar to CSFQ. Edge routers use hash algorithms, such as SHA-1, to transfer the flow label of each packet into a key a_j that alleviates the IP dependency. Next, the key a_j is inserted into the packet's header. Figure 2 shows that the core routers have to maintain two tables; one is a first-layer table with size (b^n, L), and the other is a second-layer table with size (n, p). When a packet arrives at the core router, AFQ extracts key a_j with size s from the packet header of the arrival and compares it with the keys from the first-layer table instead of the flow labels. Next, we use the same circular replacement method to maintain the keys in the first-layer table. If there is a hit at filtering level z where $z \in [b^0, b^n]$, AFQ has to compare its flow label with a flow label at $(index_j, a_j)$ in the second-layer table. We use Equation (7) to calculate the $index_j$.

$$index_j = \{min(k)\,|(z < b^k), k \in [1, n]\} \tag{7}$$

If a packet encounters two hits, a real hit happened; otherwise, no real hit happened. A real hit corresponds to previous mentioned hit in the AFQ algorithm. If there is only a hit in the first table, the flow label of the arriving packet will replace the flow label at $(index_j, a_j)$ in the second-layer table; If different flows have the same key, the flow with more arriving packets has a higher chance to be captured in the second-layer table. In other words, its arriving packets will have a higher probability to be constrained. Consequently, AFQ can achieve sufficient discriminability. The updating of the second-layer table differs from the first-layer table in the use of the frequency-based replacement method. The two-layer mapping mechanism imposes additional burden on edge routers, but it can efficiently speed up the packet comparisons while reducing memory consumption on core routers. For instance, if s = 8 bits (get first 8 bits from160 bits producing by SHA-1), L = 32, b = 2, n = 7 and p = 128, the original AFQ requires 32 × 8 × 128 = 32,768 Bytes memory. When using the mechanism, AFQ requires approximately 32 × 1 × 128 + 8 × 128 × 7 = 11,264 Bytes. In other words, memory can be approximately reduced to two-thirds.

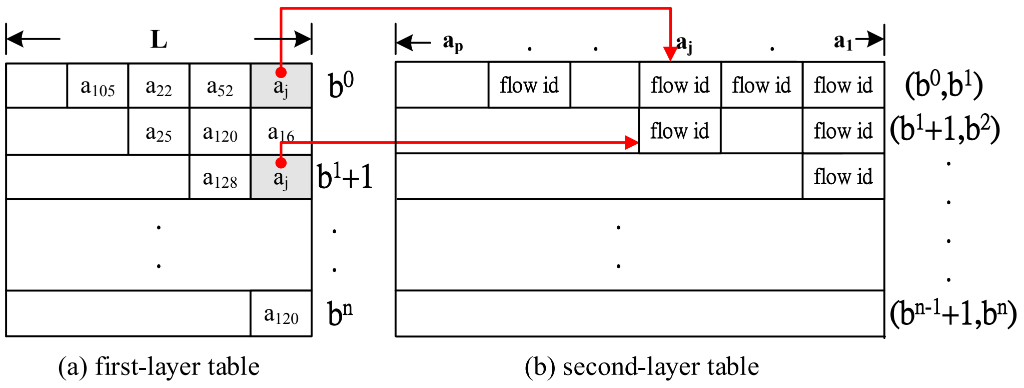

Figure 2. Two-layer mapping mechanism.

To compare the fairness, we define the Normalized Bandwidth Ratio (NBR) for a specific flow based on Equation (8). $\mathrm{NBR_j}$ denotes NBR; $\mathrm{D_j}$ denotes the mean departure rate; and $\mathrm{r_j}$ denotes the Mean Arrival Rate (MAR), all related to flow *j*. Additionally, f denotes the Max-Min Fair Share Rate (MMFSR).

$$\mathrm{NBR_j} = \mathrm{D_j}/\min(\mathrm{r_j}, \mathrm{f}) \tag{8}$$

If $\sum_{j=1}^{n} r_j > C$, f can be derived from $\sum_{j=1}^{n} \min(r_j, f) = C$, where n denotes the number of active flows. Otherwise, $f = \max(r_i), i \in n$. If a scheme achieves the optimal fairness, then the NBR of each flow is equal to 1.

4. Simulation Results

We simulated four well-known schemes (RPQ, CSFQ, CHOKe and FIFO) and compared their fairness by analyzing the NBR behaviors. We developed a software simulator to perform all simulations, which has been used in our previous study [6]. The traffic types of generating packets in each case are described in their respective figures. In Figure 3a,b, we consider two categories of network topologies: those with a single congested link and those with multiple congested links. Unless otherwise specified, we use the following parameters in the simulations. Each link capacity is of 10 Mbps, and the packet size is fixed at 1 KB. The buffer size for all schemes is set to 256 KB. In addition, we neglect the propagation delay of each link. In AFQ, the initial value of α_{new} is set to 32, and the other parameters are set to T_d = 200 ms, k_a = 0.8 and k_b = 1.5. In RPQ, each output queue is set to 32 KB, and the other parameters are set as follows: Δ = 0.8 ms, α = 0.8, K_d = 200 ms and *N* = 129. With respect to CSFQ, *K* and K_α are both set to 200 ms. CHOKe's parameters are set to the following values: max_{th} = 120 KB, min_{th} = 40 KB, w_q = 0.002 and max_p = 0.02. Finally, the duration on each simulation is 200 s.

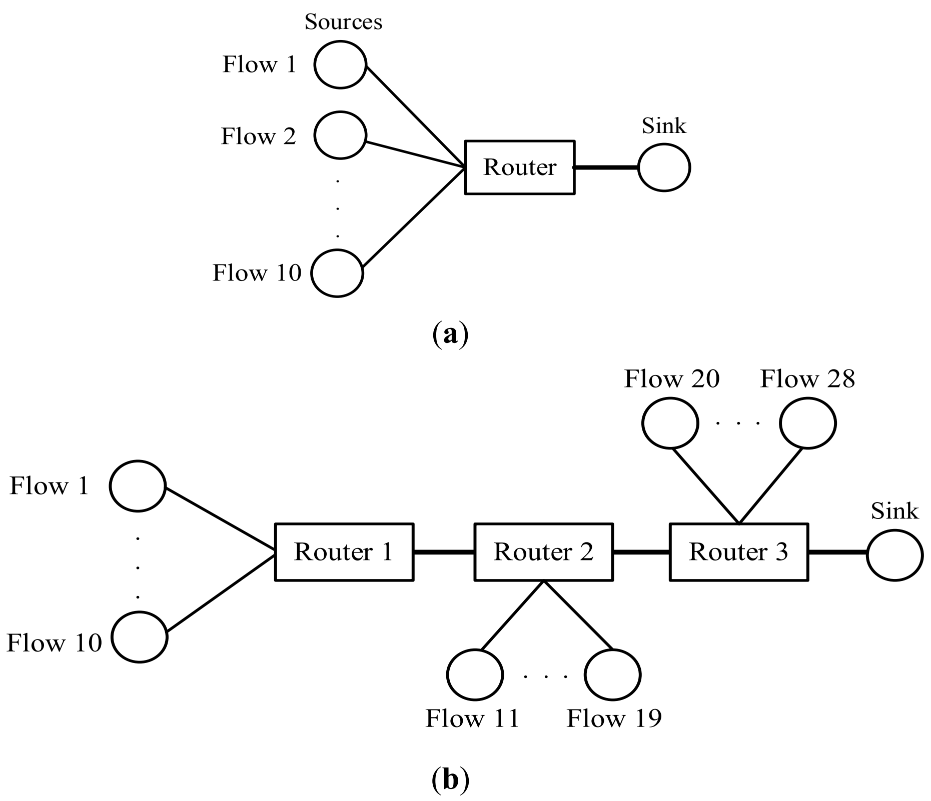

Figure 3. Network topologies. (**a**) A single congested link. (**b**) Multiple congested links.

4.1. A Single Congested Link

We consider four cases where a single congested link is shared by 10 flows, as depicted in Figure 3a. In the first case, the flows are indexed from 1 to 10; hence, the MMFSR is 1.4 Mbps. The simulation results are depicted in Figure 4. In CHOKe and FIFO, the shared bandwidth of flows 1 to 5 is roughly proportional to their MARs. Accordingly, their NBRs are close to constant. Flows 6 to 10 have the same MBR at 10 Mbps, so they fairly share the grabbed bandwidth from flows 1 to 5. Again, their NBRs maintain a constant value. FIFO is the simplest scheme, but it shows the worst fairness. Although CHOKe performs better than the FIFO, it can provide only limited fairness. In AFQ, the NBRs of flows 1 to 5 reach the largest values, and the NBRs of flows 6 to 10 reach the lowest values. In other words, AFQ achieves the best fairness. The NBRs of flows 1 to 5 in AFQ, RPQ and CSFQ all decrease when MAR is close to MMFSR because those flows are more likely to be discarded because of traffic burstiness. From the simulations, AFQ outperforms RPQ, and RPQ outperforms CSFQ in fairness. Therefore, ARQ can effectively protect the fairness of non-aggressive flows from damage due to aggressive flows.

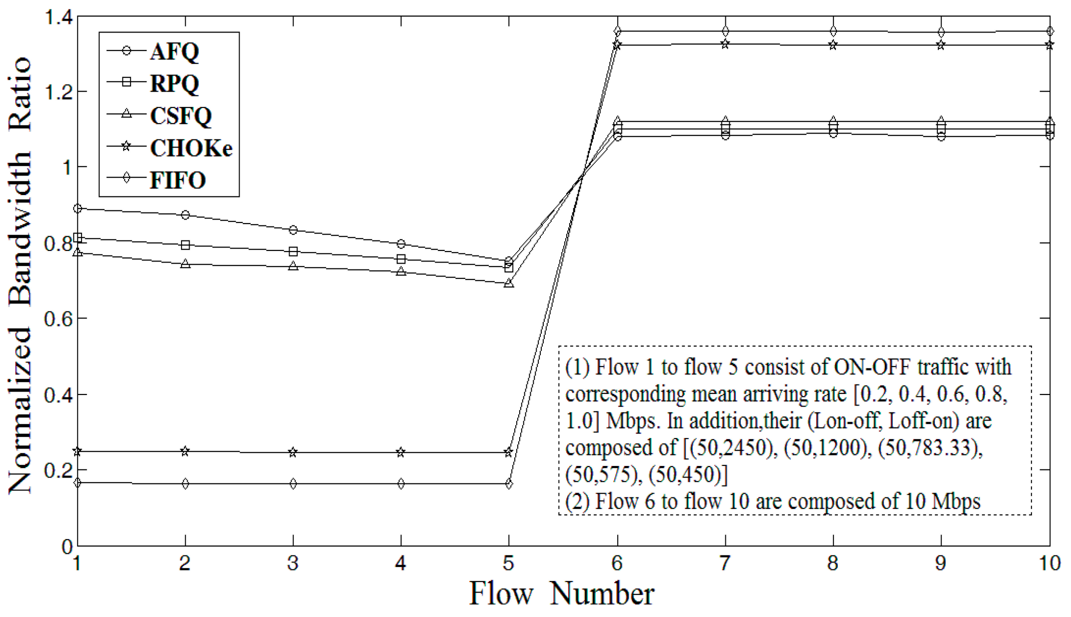

Figure 4. The Normalized Bandwidth Ratio (NBR) achieved by each of the ten flows sharing a bottleneck link; flows 1 to 5 belong to non-aggressive flows, and flows 6 to 10 belong to aggressive flows.

In the second case, we consider that all flows are composed of aggressive flows. The MAR of each flow is larger than the MMFSR (1 Mbps). The simulation results are depicted in Figure 5. Compared with the first case, flows 6 to 10 can obtain less additional bandwidth from flows 1 to 5 in CHOKe and FIFO. However, they have larger NBRs because the MMFSR is smaller in this case. CHOKe and FIFO have better fairness for two reasons: flows have approximate MARs, and the MBR of each flow is larger than the MMFSR. Similarly, RPQ and CSFQ both demonstrate better fairness; in addition, they both demonstrate approaching fairness. Repeatedly, AFQ achieves the best fairness. Consequently, RPQ can fairly allocate bandwidth under extreme traffic conditions where all flows are aggressive flows.

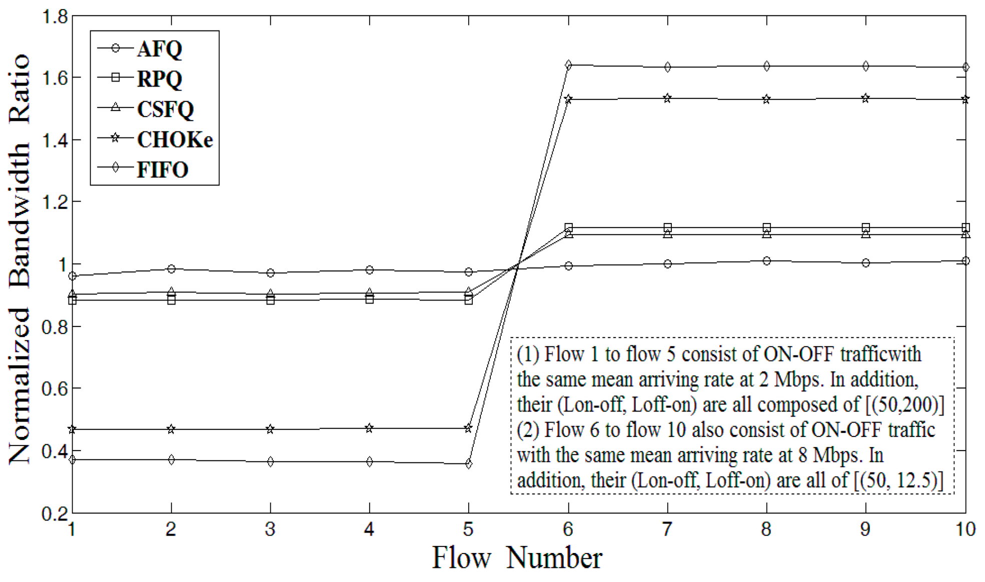

Figure 5. The NBR achieved by each of the ten flows sharing a bottleneck link, with all flows belonging to aggressive flows.

In the third case, we consider flows consisting of three magnitudes of MARs, where the MMFSR is 1.375 Mbps. The MARs of flows 1 to 3 and flows 4 to 6 equal 0.5 Mbps and 1 Mbps, respectively. In addition, the MAR of flows 7 to 10 is 6 Mbps. The simulation results are depicted in Figure 6. The MARs of flows 1 to 6 is smaller than the MMFSR, so their NBRs are constant in CHOKe and FIFO. In RPQ and CSFQ, the NBRs of flows 1 to 3 are larger than those of flows 4 to 6. Flows 1 to 3 have lower MARs, so they have less chance of being mistakenly discarded. The MARs of flows 4 to 6 are close to the MMFSR; hence, their arriving packets are more likely to be discarded during traffic burstiness. The NBRs of AFQ are better than those of RPQ and CSFQ. As a result, RPQ still keeps the best fairness under various traffic conditions.

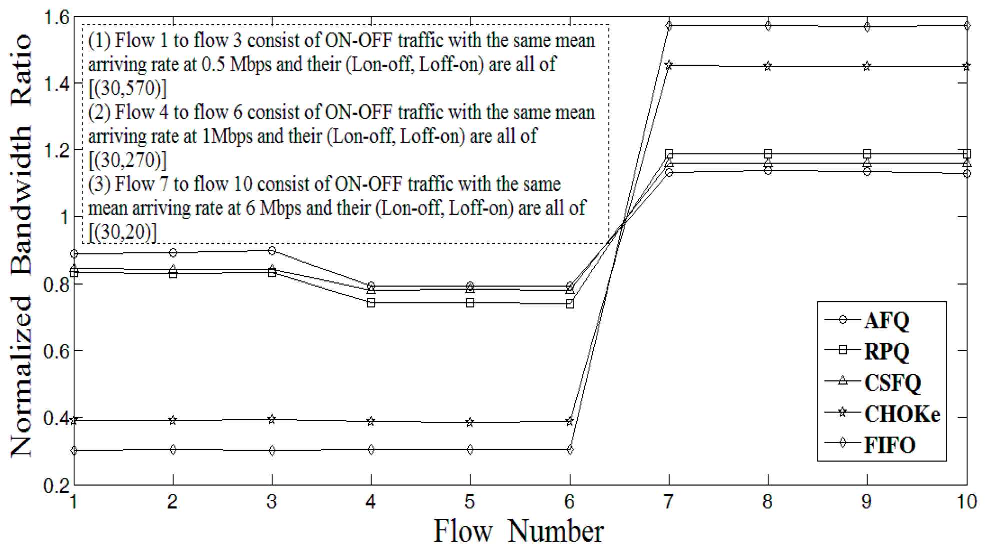

Figure 6. The NBR achieved by each of the ten flows sharing a bottleneck link, with flows consisting of three magnitudes of Mean Arrival Rate (MAR)s.

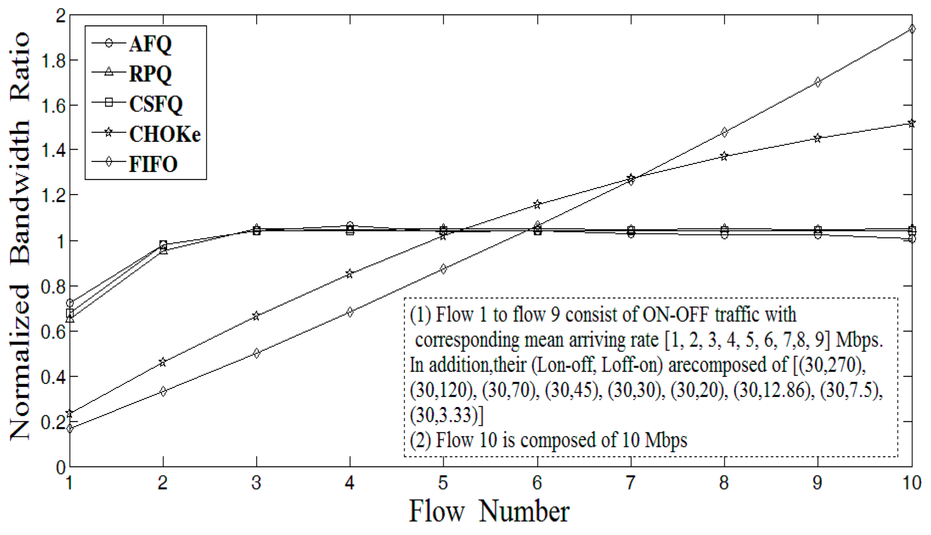

Figure 7. The NBR achieved by each of the ten flows sharing a bottleneck link, with diverse MARs.

In the fourth case, we consider that all flows belong to aggressive flows but that their MARs are diverse. The simulation results are depicted in Figure 7, and the MMFSR is 1 Mbps. Flow 1 is the only non-aggressive flow, which has a lower NBR due to larger traffic burstiness. FIFO still keeps the worst NBRs, whose trends are proportional to the MAR of each flow. We find that traffic load dominates the fairness of FIFO. Additionally, CHOKe is affected due to the same reason, but its fairness is better than that of FIFO. CSFQ has to estimate the MAR of each flow; therefore, traffic variations can influence fairness. However, exponential flow rate estimation methods can relieve such an effect. The fairness of RPQ is close to that of CSFQ, but it can also enhance fairness by increasing the number of output queues. As mentioned previously, this also reduces larger queueing delays. AFQ repeatedly outperforms the other schemes. In Figures 4–7, we conclude that AFQ is able to achieve robust and optimal fairness under a single congested link, where various traffic conditions are considered.

4.2. Multiple Congested Links

We now analyze how the NBRs of five flows with smaller MARs are affected by other flows with larger MARs under three congested links. We performed two cases based on the topology shown in Figure 3b. Flows 1 to 5 send at 0.2, 0.4, 0.6, 0.8 and 1 Mbps, and the others send at 10 Mbps. The capacity of each link is 10 Mbps so that each link between routers is congested. We consider the NBRs of flows 1 to 5 here because their fairness can be damaged by other aggressive flows. The simulation results are illustrated in Figure 8, and the MMFSR is 0.363 Mbps at the last router. The NBRs of flows 1 to 5 are all close to 0 in CHOKe and FIFO. In other words, they both fail to provide fairness. When the number of traversing routers increases, CSFQ gradually loses precision on estimates of the fair share rate and MAR of each flow. As a result, AFQ performs better than RPQ, while RPQ performs much better than CSFQ. AFQ achieves the best fairness because it can dynamically filter the arriving packets according to traffic variations.

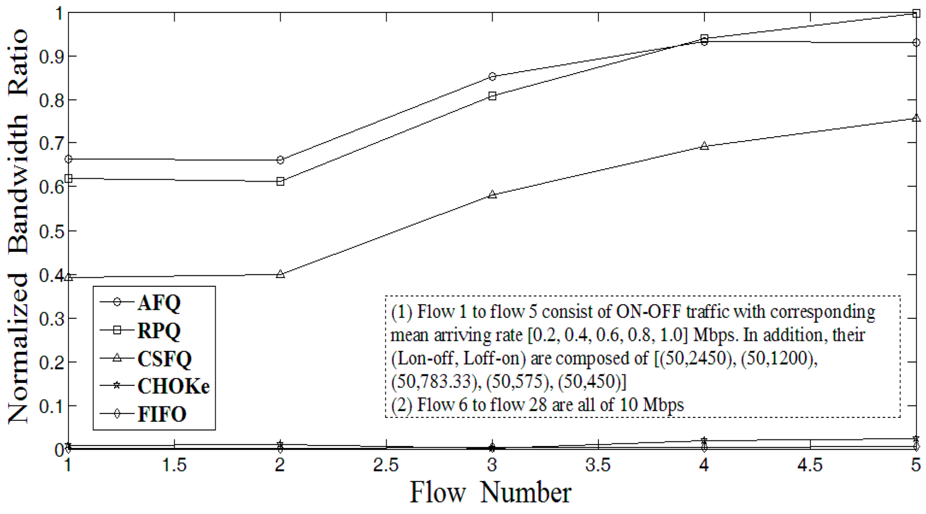

Figure 8. The NBR achieved by flows 1 to 5 with lower ON-OFF burstiness, with other flows all consisting of 10 Mbps.

In the second case, we increase double traffic burstiness to flows 1 to 5, and the MBR of flows 6 to 28 change to 8 Mbps. Figure 9 illustrates the NBRs of flows 1 to 5, and the MMFSRs are the same as in the first case. Repeatedly, AFQ proves to be the scheme with the best fairness. Although CSFQ is worse than RPQ, it is still much better than CHOKe and FIFO. AFQ, RPQ and CSFQ degrade their fairness compared with the first case because the traffic burstiness of flows 1 to 5 is double. According to our simulation results, we conclude that AFQ can achieve the best degree of fairness under different network topologies and various traffic conditions.

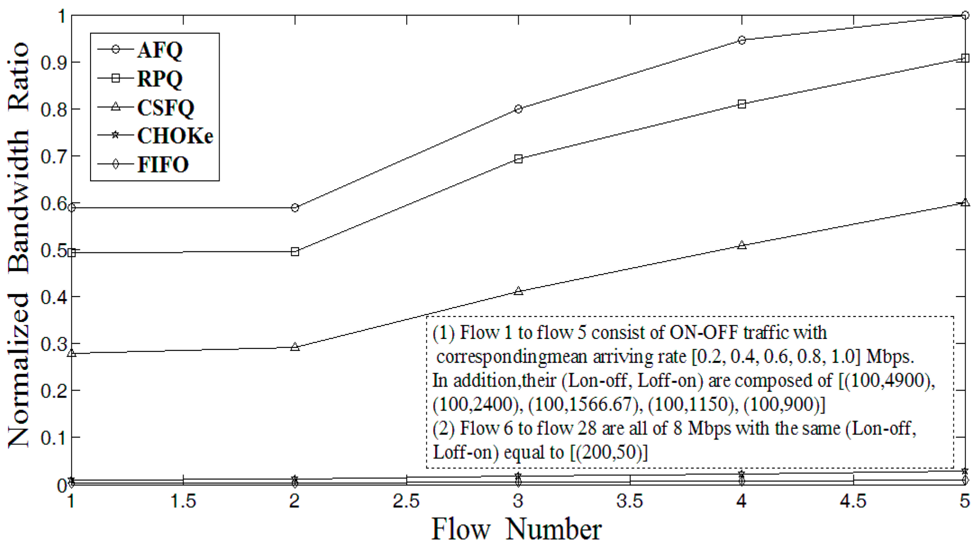

Figure 9. The NBR achieved by flows 1 to 5 with double ON-OFF burstiness, with other flows all consisting of 8 Mbps.

5. Conclusions

In this paper, we present a scalable and efficient AFQ scheme that achieves fair bandwidth sharing under various traffic conditions. The routers employ a simple packet-dropping algorithm to determine whether arriving packets are accepted or dropped according to the filtering levels of arriving packets and the estimates of fair filtering levels. In addition, we propose a mechanism that can effectively simplify packet comparisons while reducing memory consumption. Accordingly, AFQ is suitable for deployment in high-speed networks. The mechanism works under the same router environments as CSFQ. We analyzed the fairness of AFQ and four other schemes under different network topologies and different traffic conditions. The simulation results demonstrate that AFQ is superior to RPQ and CSFQ and performs much better than CHOKe and FIFO. In the future, we aim to study the effect of various TCP variants on AFQ. Moreover, we plan to study an enhanced AFQ version to ensure that it can support real-time applications while keeping fairness using dynamic bandwidth readjustment.

Acknowledgments

The author acknowledges financial support from the Shih-Chien University and the Ministry of Science and Technology in Taiwan, under the grant numbers USC 103-05-05012 and MOST 103-2221-E-158-001, respectively.

Conflicts of Interest

The author declares no conflict of interest.

References

1. Floyd, S.; Jacobson, V. Random early detection gateways for congestion avoidance. *IEEE/ACM Trans. Netw.* **1993**, *1*, 397–413.
2. Pan, R.; Prabhakar, B.; Psounis, K. CHOKe: A stateless active queue management scheme for approximating fair bandwidth allocation. In Proceedings of the Nineteenth IEEE Computer and Communications Societies (INFOCOM 2000), Tel Aviv, Israel, 26–30 March 2000; pp. 942–951.
3. Chhabra, P.; John, A.; Saran, H.; Shorey, R. Controlling malicious sources at Internet gateways. In Proceedings of the IEEE International Conference (ICC'03), Anchorage, AL, USA, 11–15 May 2003; pp. 1636–1640.
4. Stoica, I.; Shenker, S.; Zhang, H. Core-stateless fair queueing: A scalable architecture to approximate fair bandwidth allocations in high-speed networks. *IEEE/ACM Trans. Netw.* **2003**, *11*, 33–46.
5. Cao, Z.; Wang, Z.; Zegura, E. Rainbow fair queueing: Fair bandwidth sharing without per-flow state. In Proceedings of the Nineteenth IEEE Computer and Communications Societies (INFOCOM 2000), Tel Aviv, Israel, 26–30 March 2000; pp. 922–931.
6. Yang, J.P. Rotating preference queues: An efficient queue management scheme for fair bandwidth allocation. *IEEE Commun. Lett.* **2013**, *17*, 420–423.
7. Shreedhar, M.; Varghese, G. Efficient fair queuing using deficit round-robin. *IEEE/ACM Trans. Netw.* **1996**, *4*, 375–385.
8. Yang, J.P. Dynamic detection and expulsion buffer management scheme for high-speed networks. *IEICE Trans. Commun.* **2011**, *E94-B*, 2243–2246.
9. Zhou, X.; Ippoliti, D.; Zhang, L. Fair bandwidth sharing and delay differentiation: Joint packet scheduling with buffer management. *Comput. Commun.* **2008**, *31*, 4072–4080.
10. Laias, E.; Awan, I. An interactive QoS framework for fixed WiMAX networks. *Simul. Model. Pract. Theory* **2010**, *18*, 291–303.
11. Alsahag, A.M.; Ali, B.M.; Noordin, N.K.; Mohamad, H. Fair uplink bandwidth allocation and latency guarantee for mobile WiMAX using fuzzy adaptive deficit round robin. *J. Netw. Comput. Appl.* **2014**, *39*, 17–25.
12. Valente, P. Reducing the execution time of fair-queueing packet schedulers. *Comput. Commun.* **2014**, *47*, 16–33.
13. Feng, W.C.; Kandlur, D.D.; Saha, D.; Shin, K.G. A self-configuring RED gateway. In Proceedings of the Eighteenth IEEE Computer and Communications Societies (INFOCOM'99), New York, NY, USA, 21–25 March 1999; pp. 1320–1328.
14. Alasem, R.; Abu-Mansour, H. EF-AQM: Efficient and fair bandwidth allocation AQM scheme for wireless networks. In Proceedings of the Computational Intelligence, Communication Systems and Networks (CICSyN), Liverpool, UK, 28–30 July 2010; pp. 169–172.

15. Aldabbagh, G.; Rio, M.; Darwazeh, I. Fair early drop: An active queue management scheme for the control of unresponsive flows. In Proceedings of the 10th Computer and Information Technology (CIT), Bradford, UK, 29 June–1 July 2010; pp. 2668–2675.
16. Yilmaza, M.; Ansari, N. Achieving destination differentiation in ingress aggregated fairness for resilient packet rings by weighted destination based fair dropping. *Comput. Netw.* **2014**, *67*, 43–54.
17. Khosroshahi, M. UARA in edge routers: An effective approach to user fairness and traffic shaping. *Int. J. Commun. Syst.* **2012**, *25*, 169–184.
18. Abdelhafid, A.; Mohamed-el-Amine, B.; Pascal, L. A radio-aware worst-case fair weighted fair queuing scheduler for WiMAX networks. *Int. J. Commun. Syst.* **2014**, *27*, 13–30.
19. Xue, L.; Kumar, S.; Cui, C.; Park, S.-J. A study of fairness among heterogeneous TCP variants over 10 Gbps high-speed optical networks. *Opt. Switch. Netw.* **2014**, *13*, 124–134.

© 2015 by the authors; licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution license (http://creativecommons.org/licenses/by/4.0/).