Pamukkale Univ Muh Bilim Derg, 24(2), 246-254, 2018

Pamukkale Üniversitesi Mühendislik Bilimleri Dergisi

Pamukkale University Journal of Engineering Sciences

Non-preemptive priority scheduler with multiple thresholds for network routers

Ağ yönlendiricileri için çoklu eşik kullanan geçişsiz öncelikli zamanlayıcı

Tamer DAĞ[1*]

[1]Computer Engineering Department, Faculty of Engineering and Natural Sciences, Kadir Has University, Istanbul, Turkey.
tamer.dag@khas.edu.tr

Received/Geliş Tarihi: 24.10.2016, Accepted/Kabul Tarihi: 15.03.2017 doi: 10.5505/pajes.2017.74318
* Corresponding author/Yazışılan Yazar Research Article/Araştırma Makalesi

Abstract

The vast variety of applications available and being developed for computer networks have different quality of service requirements. One of the most significant ways to satisfy the needs of the applications is the packet scheduling algorithms employed by the network routers. By allocating router resources to the applications, packet schedulers try to improve the quality of service needs of the applications. Thus, the delays can be reduced or the reliability of the applications can be increased by reducing packet losses. Priority schedulers are able reduce the delay and losses for high priority applications. On the other hand, for low priority applications they introduce the starvation problem. Low priority application packets can face excessive delays and losses. In this paper, a non-preemptive priority scheduler with multiple thresholds (PRMT) is proposed. The PRMT scheduler needs only a single queue with predefined threshold levels for different priority applications. The PRMT scheduler eliminates the starvation problem of low priority applications without a significant impact on the high priority applications.

Keywords: Network routers, Quality of service, Priority scheduler, Multiple threshold

Öz

Bilgisayar ağları için mevcut ve geliştirilmekte olan çok çeşitli uygulamaların farklı hizmet kalitesi gereksinimleri vardır. Uygulamaların ihtiyaçlarını karşılamanın en önemli yollarından birisi ağ yönlendiricileri tarafından kullanlan paket zamanlama algoritmalarıdır. Uygulamalara yönlendirici kaynaklarını ayırarak, paket zamanlayıcıları uygulamaların hizmet kalitesi ihtiyaçlarını artırmaya çalışır. Bu nedenle, gecikmeler azaltılabilir ya da uygulamaların güvenilirliği paket kayıplarını azaltmak suretiyle arttırılabilir. Öncelikli zamanlayıcılar, yüksek öncelikli uygulamaların gecikme ve kayıplarını azaltabilirler. Öte yandan, düşük öncelikli uygulamalar için açlık problemi getirirler. Düşük öncelikli uygulama paketleri aşırı gecikmeler ve kayıplarla karşılaşabilirler. Bu makalede çoklu eşik kullanan geçişsiz öncelikli zamanlayıcı (PRMT) önerilmektedir. PRMT zamanlayıcı farklı öncelikli uygulamalar için önceden tanımlanmış eşik seviyeleri kullanan tek bir kuyruk ihtiyacı duyar. PRMT zamanlayıcı yüksek öncelikli uygulamaların üzerinde önemli bir etkiye yol açmadan, düşük öncelikli uygulamalardaki açlık sorunu ortadan kaldırır.

Anahtar kelimeler: Ağ yönlendiricileri, Servis kalitesi, Öncelikli zamanlayıcı, Çoklu eşik

1 Introduction

Computer networks support an enormous variety of applications from digital audio to instant messaging, from peer-to-peer file sharing to streaming video. Not only the number of these applications but also the number of users are rapidly increasing. The growth rate of applications and the number of users are expected to increase further with the spread of Internet of Things (IoT) [1],[2] in the next decade. According to the estimates given by [3], approximately 6.4 billion connected things are in use in 2016 and this number will reach to 20.8 billion by 2020.

The applications that are in use have different requirements from the network. Treating all traffic in the same manner will result in poor or unacceptable performance for some of the applications, as the requirements of those applications may be more critical compared to the others. Thus, different types of services should be provided for different applications according to their needs.

Quality of Service (QoS) refers to the capability of a network to provide differentiated and better service for distinct network applications. The basic QoS parameters [4] are specified as bandwidth, latency, jitter and loss ratio. These four parameters usually define the QoS requirements of an application.

The bandwidth refers the rate at which traffic is carried by the network. The amount of bandwidth required by an application varies significantly and depends on the application type. For example, video applications such as HD streaming video are in need of high bandwidth. However, for data applications such as e-mail or remote login lower bandwidth might suffice.

Latency is the delay in data transmission. Some applications are sensitive to delay such as interactive applications. For such applications, the users will find the connection unacceptable for high delay values. For example, VoIP applications cannot tolerate delay more than one sec. But, streaming audio or video applications do not have low delay requirements.

Jitter is the variation in the latency. Jitter requirement is also application specific. For example, real-time voice applications require low jitter values and they can suffer when the jitter increases. However, for a file sharing application jitter is not very important and does not impact the performance of the application. Thus, some applications may tolerate high jitter values, while some can not.

Loss ratio is the percentage of packets discarded by routers. Since the router resources are finite, under heavy traffic or when there is congestion in the network, some packets may not be accepted for service and they are discarded and lost. The amount of packet losses that an application can tolerate varies considerably. For example, while audio applications are able to

tolerate some losses, for some applications such as e-mail every single bit must be correctly delivered to the destination without any losses.

In order to satisfy the QoS requirements of the applications, several disciplines such as admission control and traffic control can be implemented. Admission control [5] determines which applications and users can access the resources provided by the network. After an application declares its QoS requirements, the network decides if it can accept the traffic through resource reservations, call admission or call setup signaling. Traffic control regulates the data flowing in the network by classifying, shaping, policing and scheduling the traffic. These mechanisms separate the traffic into service classes and control each service class.

Packet scheduling is one of the ways for the traffic control discipline. Packet scheduling algorithms allocate router resources to competing applications in terms of bandwidth, buffer space or CPU cycles. Thus, it is an effective way for providing better QoS to applications.

In this work, a non-preemptive priority packet scheduler with multiple thresholds is proposed. The priority schedulers usually suffer from the starvation problem. Low priority application traffic may never get serviced when there is high priority traffic present in the router. The delay and packet losses that low priority applications are imposed can reach excessive amounts. The proposed scheduler allows lower priority traffic to be serviced even with the presence of higher priority traffic, when the buffer is not congested. This way allows reduced packet drop ratios and latency values for lower priority traffic. High priority application QoS is not affected since when the buffer is congested, the proposed scheduler services high priority traffic first. Thus, with the proposed scheduler, the starvation problem commonly seen in priority schedulers can be reduced and solved.

The organization for the rest of the paper is as follows. Section 2 describes most common scheduling algorithms and packet drop policies in use. Section 3 introduces the proposed scheduling algorithm, the policies for accepting incoming packets and how they are serviced. Section 4 discusses the simulation environment used for simulating the proposed algorithm and the obtained results. Finally, the paper ends with the conclusions made in Section 5.

2 Packet scheduling

A packet scheduler is a traffic control discipline which determines the transmission order of packets at a router in order to respond to the application's QoS requirements. Packet scheduling algorithms have been a major research area [6],[7] as they are one of the most critical components in a network for increasing the level of QoS offered for applications.

Packet schedulers can be classified into non-work-conserving packet schedulers and work-conserving packet schedulers [8].

With a non-work-conserving scheduler, the scheduler can stay idle even when there are packets waiting to be serviced. Non-work-conserving schedulers have the benefit of reducing the jitter, thus making the downstream traffic more predictable and causing fewer and smaller traffic bursts.

For a non-work-conserving scheduling algorithm, the scheduler waits until a packet is eligible for transmission. The eligibility of the packets for transmission might be decided in different ways. To control the end-to-end jitter, it can be ensured that a packet always spends no more than a fixed time at a router. Another way to decide eligibility is to establish the notion of a fixed end-to-end delay for a packet. In this case, routers decide a packet is eligible based on the time budget that remains for each packet.

Non-work-conserving schedulers might reduce the buffering requirements at routers and end-systems. But, the end-systems can already fix this issue as they can de-jitter flows as needed. The cost of using such mechanisms is higher overall end-to-end delay. In addition, such systems can be complex to build in practise. Consequently, non-work-conserving disciplines are not used but considered as an active research area.

With a work-conserving scheduler, the router never stays idle when there are packets waiting to be serviced. A scheduler is work-conserving, if Kleinrock's conservation law [9] stated by Equation (1) holds. In this equation, ρ_n is the average link utilization for flow n, q_n is the average waiting time caused by the scheduler for flow n, and C is a constant for the N flows arriving at the scheduler. Based on the conservation law, providing a better QoS for one flow will have a negative impact on the QoS of one or more others. If the delay for a flow is lowered through a scheduling algorithm, then the delay for other flows will increase. If the loss ratio for a flow is lowered, the loss ratio of other flows will increase, in a similar sense. Thus, work-conserving schedulers need to be carefully designed without hurting the needs of some flows too much.

$$\sum_{n=1}^{N} \rho_n q_n = C \tag{1}$$

The rest of this section describes the most common scheduling algorithms and packet dropping policies.

2.1 First-Come first-served (FCFS) scheduling

Traditional routers use the first-come first served (FCFS) scheduling policy. The simple and easy to implement FCFS scheduler schedules the packets in the same order as they arrive at the router. With FCFS, there is no kind of service differentiation and all the packets are treated with the same manner. Thus FCFS is suitable for networks where no QoS guarantees are required. FCFS is a work-conserving scheduling disciple.

[10] studies on the performance analysis of FCFS. In [11], FCFS and EDF scheduling algorithms for real-time packet switching networks are compared in terms of packet loss, delay and buffer size.

2.2 Priority scheduling

Under priority scheduling, the packets arriving at a router are classified into several priority classes. A packet's priority class can depend on an explicit marking carried in the packet header, source or destination addresses, port number or other criteria.

The priority scheduling establishes a queue for each priority level and the queues are served in the order of their priority. With priority scheduling, high priority traffic always gets the best possible service quality as higher priority queues are serviced before lower priority queues. On the other hand, this approach will lead into excessive delays and losses for low priority traffic as lower priority queues will only be serviced if there are no packets awaiting service in a higher priority queue. As a consequence, priority scheduling can result in a situation called as starvation. If there is a constant flow of high priority

packets, low priority packets will be prevented from getting service.

Priority schedulers can be either preemptive or non-preemptive. A non-preemptive scheduler can never interrupt a packet in service. That is, if a packet is in service, its service will be completed even if a higher priority packet arrives in the meanwhile. On the other hand, a preemptive scheduler can interrupt a packet in service. If a higher priority packet arrives while a lower priority packet is in service, lower priority packet will be taken out of service and higher priority packet will start being serviced. Priority scheduling is a work-conserving scheduling disciple.

In [12], a fuzzy based dynamic multilevel priority packet scheduler is proposed with three types of priority queues. [13] proposes an energy efficient packet scheduling with delay and loss constraints. Every incoming data packet is analysed and prioritized based on delay and loss constraints. However, this type of approach introduces excessive processing delays. In [14], dynamic multi threshold priority scheduling algorithm is introduced. If a packet arrives with an urgent flag set, its priority is increased and the packet is placed forward in the queue. This algorithm may require shifting off all the packets present in the queue and therefore increases the time complexity significantly. In [15] and [16], the priority queues are organized in such a way that real time packets are stored in high priority queue, non-real time packets whose destinations are distant nodes are stored in medium priority queue and non-real time packets whose destinations are local nodes are stored in low priority queue. In these studies, the priority scheduler is preemptive. While non-real time packets are processing, an arriving real-time packet can preempt. [17] also proposes a preemptive scheduling algorithm that can be applied to wireless sensor networks. In this paper, not only the high priority packets can preempt low priority packets, but also low priority packets can preempt high priority packets given that they have waited for a long time. Thus, this scheduler needs to follow the waiting time of low priority packets in the queue, thus extra overhead is introduced with the algorithm.

2.3 Deadline based scheduling

Deadline based schedulers [11],[18] check arriving packet's delivery deadline and give priority to packets which have less remaining deadline. This approach enables packets with lower remaining deadlines to reach their destination faster, however similar to priority schedulers, packets with higher remaining deadlines may get stacked at the routers, leading into starvation. Deadline based schedulers are work-conserving schedulers.

In [19], a frame oriented round-robin earliest deadline first (EDF-RR) is introduced. [20] develops a modified earliest deadline first scheduling algorithm for real-time traffic in LTE networks by improving average throughput, delay, packet loss ratio and fairness index.

2.4 Fair queuing

Fair queuing relies on the General Processor Sharing (GPS) model. GPS is a work-conserving scheduler aiming to provide max-min fairness with probabilistic, statistical and deterministic performance bounds. However, it is only a theoretical model and some approximations have been presented such as Weighted Fair Queuing (WFQ) [21] and Packet based GPS (PGPS) [22].

2.5 Packet dropping policies

Under heavy traffic the routers may become congested and the router buffers start to fill up very quickly. When the router buffers are completely occupied, a new arrival might not be accepted in the buffer. In this situation, a packet will need to be dropped.

The most common way for dropping packets is drop-from-tail. In this method, the packets at the end of the queue are dropped. Drop-from-tail is easy to implement, however, the packets already in the queue and already delayed may expire. When such packets arrive at their destination, its usefulness for the application might be insignificant.

A solution for this problem might be to drop-from-head. The packets that have stayed in the network longest are dropped, giving way to new arrivals. But the processing time of drop-from-head is significantly larger, as all the remaining packets in the queue will need to be shifted.

Packets can also be dropped in random. With random drop packets from different buffer locations are selected at random and dropped. Under random drop, packets belonging to a flow using a higher bandwidth are affected more. It is also possible to see the following scenario under random drop. The dropped packets may belong to a high priority application traffic flow, while there are lower priority application packets in the queue.

Another way to drop packets is to flush the whole queue. Thus, in case of a congestion all the packets already waiting in the queue are dropped. This method immediately frees all the buffers for the new arriving packets. However, it is not efficient as for real-time applications having a big hole in their traffic flow as a result of flushing might not be acceptable.

Another alternative is intelligent dropping, where packets are dropped by knowledge of the traffic flow. In PRMT, this approach is used. If the latest arrival's priority causing a buffer overflow is higher than some packets priority waiting in the queue, it is more beneficial to pick lower priority packets as a candidate to drop instead of high priority packets. But if there are available lower priority traffic packets in the queue, then the latest arriving packet is dropped. This method needs more information from the router, but appears to have the potential of offering very fine-grained congestion control.

3 Non-preemptive priority scheduler with multiple thresholds

In this section, the proposed priority scheduler with multiple thresholds (PRMT) is described. The proposed scheduler aims to reduce the starvation problem commonly seen in priority schedulers.

PRMT defines a single queue with three levels of priorities for different application types. However, the number of priority levels can be increased or decreased based on the application needs. The considered priority levels are high priority (priority level 1), medium priority (priority level 2) and low priority (priority level 3). For each priority level, the buffer uses different threshold levels. While the threshold level for high priority applications is lower, the threshold levels for low priority applications is higher. An example buffer structure with the indicated threshold levels is shown in Figure 1.

For a buffer with capacity $\boldsymbol{C}$, high priority packets' threshold level is Th1, medium priority packets' threshold level is Th2 and low priority packets' threshold level is Th3. The reason for

defining low threshold levels to high priority applications is to place high priority application packets between the head of the queue and its threshold level whenever possible, so that the delay that they will face is reduced as well as the packet losses.

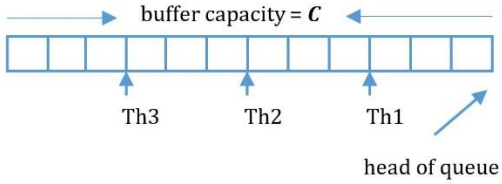

Figure 1: Threshold structure for PRMT.

3.1 Packet arrival with PRMT

When a new packet arrives at the buffer, the scheduler takes an action based on the incoming packet's priority level, the buffer occupancy and the threshold level defined for the arriving packet.

The following cases describe, what happens at the buffer with a new packet arrival.

- If the new arrival finds the buffer empty, the arriving packet is placed at the head of the buffer and immediately serviced as PRMT is a work-conserving scheduler. An example illustration depicting this case is shown in Figure 2. When a packet with priority level 2 arrives at an empty queue, it is placed at the head of the buffer and the PRMT scheduler immediately starts service for that packet.

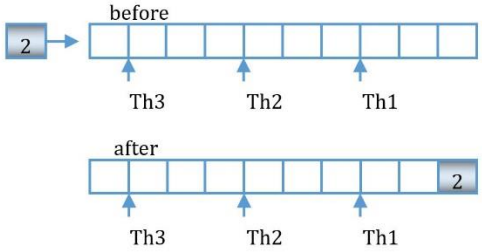

Figure 2: Arrival of packet with priority level 2 to an empty buffer.

- If the new arrival finds the buffer occupancy less than its corresponding threshold value, the arriving packet is placed at the end of the queue just like in FCFS scheduling. Higher priority packets will not prevent lower priority packets ahead of them from getting service. As a result, lower priority packets in front of the new arrival will be served before. By this way, low priority packets' starvation rate will be reduced. An example illustration depicting this case is shown in Figure 3.

  When a packet with priority level 2 arrives at a buffer with occupancy less than Th2, the packet is placed at the end of the queue. Even though there are priority level 3 packets in the front, they will still be served before the newly arriving packet. As the new arriving packet is placed ahead of its defined threshold level, its QoS needs can still be satisfied, without deteriorating low priority applications QoS needs.

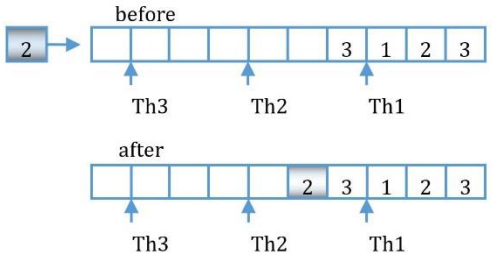

Figure 3: Arrival of packet with priority level 2 to a buffer with occupancy less than its corresponding threshold.

- If the new arrival finds the buffer occupancy more than its corresponding threshold value, scheduling decisions are made in favour of high priority packets, in order not to deteriorate their QoS requirements. Different from the previous case, the algorithm will not allow a lower priority packet to be serviced before the newly arriving packet. The newly arriving high priority packet is replaced with a lower priority packet between the threshold and the head of the queue. If there are multiple lower priority packets between the threshold level of the arriving packet and the head of the queue, the one which is closer to the threshold is chosen, as the others have already been waiting for a long time, and they will be serviced soon.

  An example illustration depicting this case is shown in Figure 4. When a packet with priority level 2 arrives at a buffer with occupancy greater than Th2, the newly arriving packet is replaced with the first priority level 3 packet before Th2.

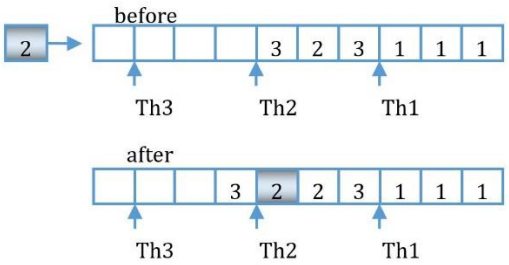

Figure 4: Arrival of packet with priority level 2 to a buffer with occupancy larger than its corresponding threshold.

  On the other hand, when new packet arrives, if there are no lower priority packets present in the queue, the new packet will be placed at the end of the queue. All the packets before have higher priority and the algorithm will not allow to deteriorate their performance. In this case, the newly arriving packet's delay requirements will be negatively affected because of the heavy high priority traffic present in the network.

  A disadvantage of the PRMT scheduler can be observed in Figure 5. When a packet with priority level 1 arrives at a buffer with occupancy greater than Th1, it is replaced with the priority level 2 packet before Th1. However, this replacement places priority level 2 packet after priority level 3 packets even though it has arrived before priority level 3 packets. Such a case might increase the waiting time of the

priority level 2 packet. This situation might be avoided by shifting the low priority packets, but shifting operation would increase the complexity of the algorithm. On the other hand, it is very likely that the lowest priority packets will be placed backwards in the queue with the later arrivals of higher priority packets. Thus, these type of occurrences are not expected to have a significant impact on the performance of the algorithm.

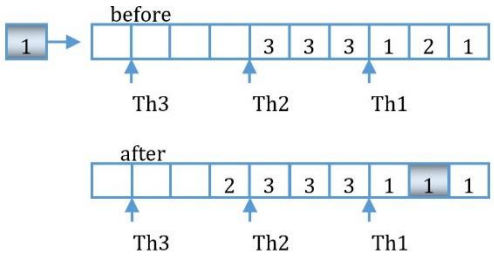

Figure 5: Arrival of packet with priority level 1 to a buffer with occupancy larger than its corresponding threshold.

- If the new arrival finds the buffer full at its maximum capacity ***C***, the PRMT scheduler drops a lower priority packet waiting in the queue, and places the new arriving packet in its place.

  An example illustration depicting this case is shown in Figure 6. When a packet with priority level 2 arrives at the full buffer, the newly arriving packet is placed in the place of the first priority level 3 packet, and the priority level 3 packet is dropped.

  If there are no lower priority packets, when the new packet arrives, then there is no choice left other than dropping the newly arriving packet.

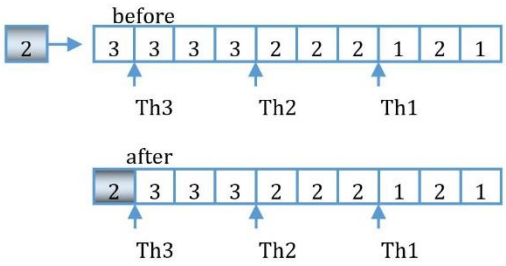

Figure 6: Arrival of packet with priority level 2 to a buffer with full occupancy.

The flowchart of PRMT at a packet arrival is shown in Figure 7. For every new packet arrival, the steps are repeated.

3.2 Packet service with PRMT

Since PRMT is work-conserving, the server will be busy serving a packet at the head of the queue whenever there are packets present in the queue. The server will always pick the first packet in the queue for service. Only when the buffer is completely empty, the server will sit idle.

If an arriving packet whether high priority or low priority finds the buffer empty, it will enter service immediately. Otherwise, the packet at the head of the queue will be picked up by the server for transmission. If there are no waiting packets the server will be idle.

Since PRMT is a non-preemptive scheduling algorithm, an arriving may never interfere the current transmission, even if the arriving packet is of highest priority and the packet under service is of lowest priority.

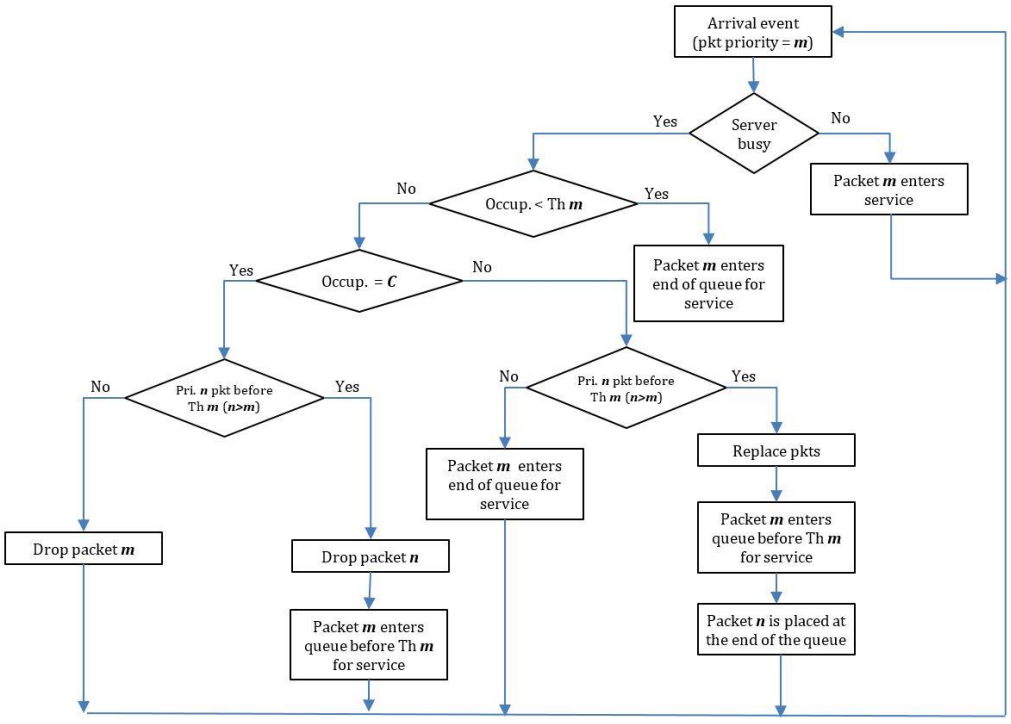

Figure 7: PRMT scheduler flowchart for packet arrivals.

4 Simulation and analysis of PRMT

In this section, the simulation method for the analysis of PRMT scheduler and the obtained results are discussed. For the simulations, the discrete event system simulation method [23] is used. Discrete event system simulation is based on event scheduling and time advance algorithms.

Two events are observed in a packet scheduling algorithm: Packet arrival event and packet service completion event. The packet arrival event deals with how a packet will be placed in the buffer queue upon its arrival. For example, for the FCFS packet scheduler, an arriving packet is placed in the first available buffer space or is dropped if no space is available. An arrival event always triggers a new arrival event and conditionally triggers a service completion event only if the new arrival enters service upon arrival. Packet service completion event deals with which waiting packet will be chosen for transmission when the transmission of the current packet is completed. For example, for the PRMT packet scheduler, the packet at the head of the queue is chosen for transmission. A packet service completion event is a conditional event, meaning that only when there are waiting packets, the next packet service completion event is triggered and scheduled.

The scheduled events are placed in the future event list (FEL) The purpose of the time advance algorithm is to execute the simulation by calling the imminent event from the FEL. As events can trigger new events, the length and the contents of the FEL constantly changes throughout the simulation.

Since PRMT can be considered as a priority scheduler, the performance of PRMT is compared with priority scheduler and the FCFS scheduler traditionally employed by routers. For each of these algorithms, packet arrival and packet service completion events as well as time advance and FEL are coded with C++. Each simulation run is repeated 100 times and the averages obtained from these runs are presented in the simulation results.

Three priority level packets (high priority, medium priority and low priority) are considered throughout the simulations. The packets from different priority levels arrive in accordance with a Poisson process where the total arrival rate is λ Thus, the number of arrivals N(t) in a finite interval of length t obeys the Poisson distribution given by Equation (2).

$$P\{N(t) = n\} = \frac{(\lambda t)^n}{n!} e^{-\lambda t} \tag{2}$$

For the simulation studies, it is assumed that the arrival rate of low priority application packets is $\lambda/2$, the arrival rate of medium priority application packets is $\lambda/3$ and the arrival rate of high priority application packets is $\lambda/6$.

Only a single queue with size $\boldsymbol{C}$ is implemented for the FCFS and PRMT schedulers. However, three queues are implemented for the priority scheduler due to its nature. The total sizes of the queues are kept the same for all schedulers, and the buffer sizes for the priority scheduler are adjusted according to the packet arrival rates of each priority to make a fair comparison.

The service times are independent exponential random variables with a mean value $1/\mu$. Thus, the utilization factor of the buffer can be represented by equation (3).

$$\rho = \frac{mean\ arrival\ rate}{mean\ service\ rate} = \frac{\lambda}{\mu} \tag{3}$$

In order to measure the performance of the scheduling algorithms, the simulations are conducted under heavy traffic when the utilization factor $\boldsymbol{\rho}$ = 0.99.

Figure 8 illustrates packet drop ratios for the three levels of priorities under the simulated scheduling algorithms when the buffer capacity $\boldsymbol{C}$=50. Since FCFS scheduler does not differentiate amongst different priority levels, the loss ratios for all three priority levels are at the same levels. Under priority scheduler, high priority packets do not incur any packet losses, however as the priority levels decrease, the loss ratios increase significantly. As a consequence, with priority scheduler low priority packets suffer excessive loss ratios. The proposed PRMT scheduler, on the other hand, improves the loss ratio of the low priority packets significantly when compared with the priority scheduler by allowing low priority packets for transmission if the queue occupancy is below the corresponding threshold level. For high priority packets, the loss ratio under the PRMT scheduler is still similar to the priority scheduler.

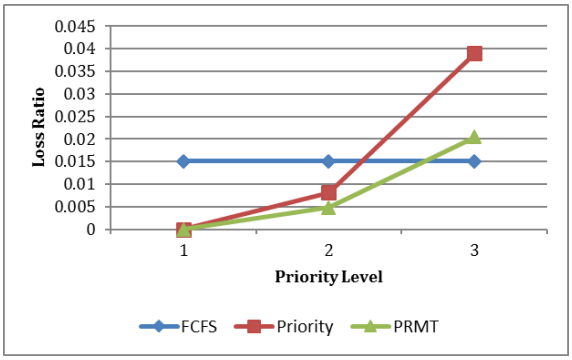

Figure 8: Loss ratio for different priorities.

Figure 9 shows the packet drop ratios for priority level 1 packets as a function of the buffer capacity *C*. 50% of the arriving packets belong to a low priority traffic, 33.3% belong to a medium priority traffic and 16.7% belong to a high priority traffic. As the buffer capacity increases, the loss ratio decreases. For priority and PRMT schedulers no losses are observed for the high priority packets as both schedulers aim to serve high priority packets first. The PRMT scheduler performs slightly worse than priority scheduler only for very low buffer capacities.

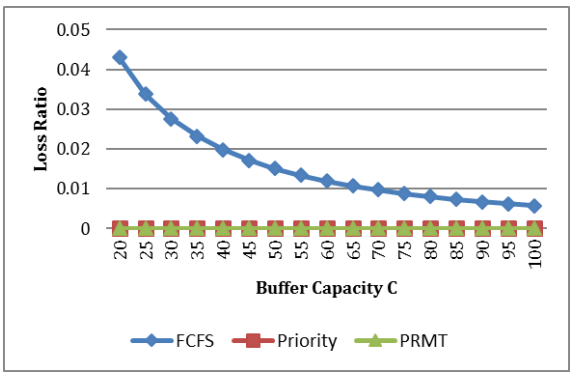

Figure 9: Loss ratio for priority 1 packets.

The difference in PRMT and priority scheduler can be explained as follows: Priority scheduler always serves high priority packets first without considering lower priority packets. But, PRMT scheduler can serve lower priority packets before high priority packets, if the occupancy of the buffer is low and consequently all incoming packets are serviced in order. In such a case, a new arriving high priority packet does not prevent the service of a lower priority packet in front of the queue. FCFS scheduler, on the other hand, observes losses, since for the FCFS scheduler a newly arriving packet does not have any impact at all on the service of the packets in the buffer on the contrary to the above discussion.

Figure 10 illustrates the packet drop ratio for priority level 2 packets as a function of the buffer capacity ***C***. As the buffer capacity increases, the loss ratios decrease. While FCFS has the worst performance, the best performance is obtained under the PRMT scheduler. PRMT scheduler allows service of medium priority level packets even if high priority packets are arriving, when the buffer occupancy is lower than the corresponding threshold value. However, priority scheduler prefers and allows the transmission of high priority packets whenever a high priority packet arrives. As a consequence, the loss ratio for priority level 2 packets under the priority scheduler is better than priority level 3 application flow but worse than priority level 1 application flow.

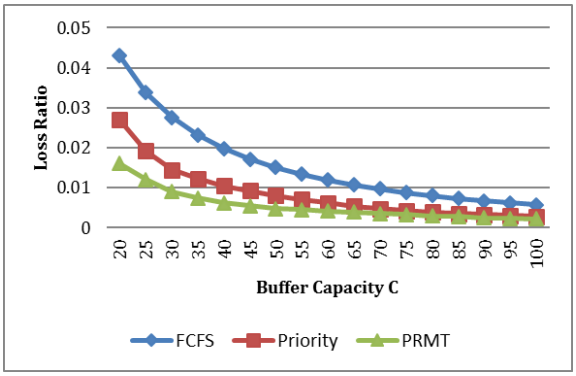

Figure 10: Loss ratio for priority 2 packets.

Figure 11 shows the packet drop ratio for priority level 3 packets as a function of the buffer capacity ***C***. As the buffer capacity increases, the loss ratios decrease. In this case, the priority scheduler has the worst performance and PRMT scheduler has the best performance slightly better than FCFS. Under the priority scheduler, low priority packets are always served last without considering the buffer occupancy. Because of this reason low priority application flows may have the starvation problem, resulting in an excessive amount of losses as can be seen from Figure 11. Since PRMT scheduler allows low priority packets to be served if the occupancy of the buffer is lower than the corresponding threshold, it reduces the starvation problem and decreases the starvation problem significantly. Only when the buffer starts to get congested, lower priority packets will be served last and might be dropped. As FCFS scheduler is indifferent to different priorities, similar loss ratio figures are observed. While this might seem beneficial for low priority applications, providing lower level of QoS for high priority applications can have severe inferences under FCFS. Thus, FCFS may not be a suitable choice for computer networks where a vast number of applications with different QoS requirements reside.

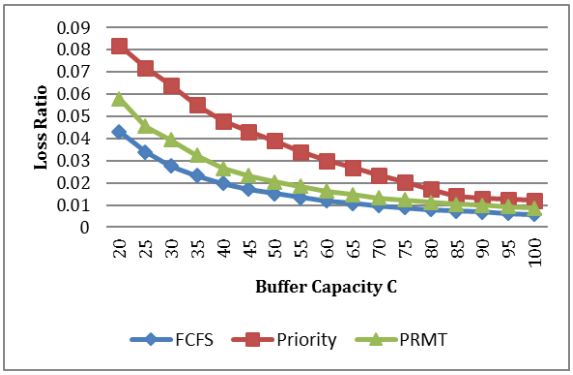

Figure 11: Loss ratio for priority 3 packets.

Figure 12 shows the latency of priority level 1 packets as a function of the buffer capacity ***C***. Similar to the previous simulations, 50% of the arriving packets belong to a low priority traffic, 33.3% belong to a medium priority traffic and 16.7% belong to a high priority traffic. As the buffer capacity increases, the latency increases. The buffer utilization is very high and the buffer tends to fill up very quickly. As a result, the waiting time in the buffer will increase with increased buffer capacity.

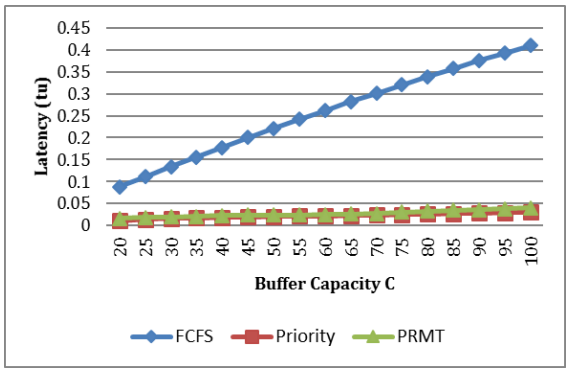

Figure 12: Average latency for priority level 1 packets.

FCFS scheduler has the worst performance for the latency of high priority packets as FCFS does not differentiate high priority application packets and provide them the same level of QoS as medium or low priority application packets. On the other hand, both priority and PRMT schedulers favor high priority packets for service and their latency values are lower compared to FCFS. The performance of the PRMT scheduler is slightly worse than the priority scheduler, as priority scheduler always selects high priority packets for transmission but PRMT scheduler allows the transmission of low priority packets in the presence of high priority packets only when the buffer occupancy is below the corresponding threshold.

In Figures 13 and 14, the latency of priority level 2 packets and priority level 3 packets as a function of the buffer capacity C is shown. As the buffer capacity increases, the latency increases. As a consequence of the starvation problem, priority scheduler is the worst for low priority application packets in terms of latency. The low priority packets need to wait for all the higher priority packets to be serviced under the priority scheduler.

The latency under the PRMT scheduler is better than that of the priority scheduler. The PRMT scheduler would let lower priority packets to be served without placing them at the end of the queue, when the buffer occupancy is lower than the

corresponding threshold values. Thus, the resulting waiting times are lower than the priority scheduler.

In addition, it can be observed that under the FCFS scheduler, both low and medium priority application packets result in similar latency figures as expected.

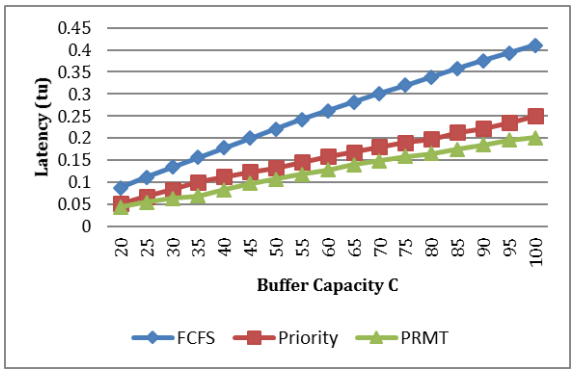

Figure 13: Average latency for priority level 2 packets.

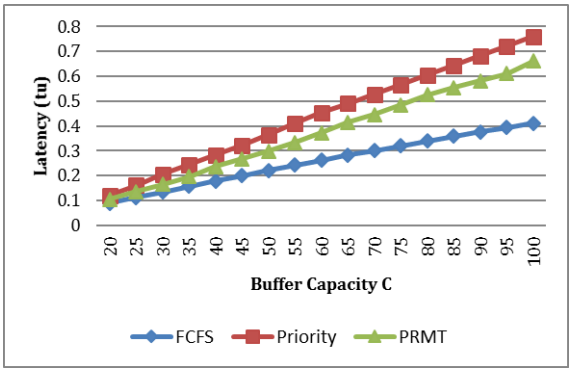

Figure 14: Average latency for priority 3 packets.

5 Conclusions

Packet scheduling algorithms allocate router resources such as bandwidth by deciding which of the buffered packets will be transmitted next or which packet will be dropped when the buffer is full. In this paper, a packet scheduler for routers named as non-preemptive priority scheduler with multiple thresholds (PRMT) is proposed. The classical FCFS scheduler does not differantiate among different types of application flows, although they may have different QoS requirements. As a result, a high or a low priority application flow receives similar service from a FCFS scheduler and the QoS provided to a high priority application flow may decrease. On the other hand, priority schedulers favor high priority application flows by allocating router resources to them, which in turn results in starvation problem for low priority application flows. The proposed PRMT scheduler allows lower priority application flows to be served even in the presence of high priority traffic, given that the buffer occupancy is lower than the predefined threshold values. By this way, the starvation problem seen in priority schedulers is reduced and a better packet drop ratio and latency is achieved for lower priority traffic. PRMT also tries to keep the QoS of higher priority traffic in the desired levels.

Another benefit of the PRMT scheduler is the simple implementation. It's time complexity is more than FCFS or priority scheduling, however with its implementation it is possible to offer better QoS levels especially to low priority applications. The PRMT scheduler requires only a single queue for all priority levels of arriving traffic. On the other hand, priority scheduler needs separate queues for each priority level. As a consequence, the PRMT scheduler can adapt to different numbers of priority levels by defining different threshold levels. But this task is more difficult for priority schedulers as more queues are required to be installed for more priority levels.

For this work, the threshold levels are assumed to be predefined. In order to further increase the performance of PRMT, the threshold levels could be dynamic based on the characteristics of the incoming flows. However, such an approach will increase the complexity of the algorithm as continuous monitoring of the amount of different priority traffic will be needed in order to adjust the threshold levels.

6 References

[1] Zorzi M, Gluhak A, Lange S, Bassi A. "From today's intranet of things to a future internet of things: a wireless-and mobility-related view". *IEEE Wireless Communications*, 17(6), 44-51, 2010

[2] Gubbi J, Buyya R, Marusic S, Palaniswami M. "Internet of Things (IoT): A vision, architectural elements, and future directions". *Future Generation Computer Systems*, 29(7), 1645-1660, 2013.

[3] Gartner Inc. "Gartner Says 6.4 Billion Connected Things Will be in Use in 2016, up 30 Percent From 2015". http://www.gartner.com/newsroom/id/3165317 (19.10.2016).

[4] Tanenbaum AS, Wetherall DJ. *Computer Networks.* 5th ed. Boston, USA, Pearson, 2011.

[5] Kurose JF, Ross KW. *Computer Networking a Top-Down Approach.* 6th ed. Essex, England, Pearson, 2013.

[6] Zhang H. "Service disciplines for guaranteed performance service in packet-switching networks". *Proceedings of the IEEE*, 83(10), 1374-1396, 1995.

[7] Necker MC. "A comparison of scheduling mechanisms for service class differentiation in HSDPA networks". *AEU-International Journal of Electronics and Communications*, 60(2), 136-141, 2006.

[8] Bhatti SN, Crowcroft J. "QoS-sensitive flows: Issues in IP packet handling". *IEEE Internet Computing*, 4(4), 48-57, 2000.

[9] Kleinrock L. *Queuing Systems, Volume 2: Computer Applications*. New York, USA, Wiley Interscience, 1975.

[10] Zhao W, Stankovic JA. "Performance analysis of FCFS and improved FCFS scheduling algorithms for dynamic real-time computer systems". *Real Time Systems Symposium*, Santa Monica, CA, USA, 5-7 December 1989.

[11] Saleh M, Dong L. "Comparing FCFS & EDF scheduling algorithms for real-time packet switching networks". *2010 International Conference on Networking, Sensing and Control (ICNSC)*, Chicago, IL, USA, 10-12 April 2010.

[12] Jain V, Agarwal S, Goswami K. "Dynamic multilevel priority packet scheduling design for WSN". *2014 International Conference on Signal Propagation and Computer Technology (ICSPCT)*, Rajasthan, India, 12-13 July 2014.

[13] Justus JJ, Sekar AC. "Energy efficient priority packet scheduling with delay and loss constraints for wireless sensor networks". *International Conference on Inventive Computation Technologies*, Tamilnadu, India, 26-27 August 2016.

[14] Dag T, Uzungenc S. "Dynamic multi threshold priority packet scheduling algorithms". *2016 International Conference on Measurement Instrumentation and Electronics (ICMIE), MATEC Web of Conferences*, Munich, Germany, 6-8 June 2016.

[15] Yantong W, Sheng Z. "An enhanced dynamic priority packet scheduling algorithm in wireless sensor networks". *18th International Conference on Computer Modelling and Simulation (UKSim)*, Cambridge, UK, 6-8 April 2016.

[16] Bansode S, Sambare S. "Performance evaluation of dynamic multilevel priority (DMP) packet scheduling for wireless sensor networks". *2015 International Conference on Pervasive Computing*, Pune, India, 8-10 January 2015.

[17] Karim L, Nasse N, Taleb T, Alqallaf A. "An efficient priority packet scheduling algorithm for wireless sensor network". *2012 IEEE International Conference on Communications (ICC)*, Ottawa, ON, Canada, 10-15 June 2012.

[18] Kargahi M, Movaghar A. "Non-preemptive earliest-deadline-first scheduling policy: a performance study". *13th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems,* Atlanta, GA, USA, 26-29 September 2005.

[19] Liu D, Lee YH. "An efficient scheduling discipline for packet switching networks using earliest deadline first round robin". *12th International Conference on Computer Communications and Networks (ICCCN),* Dallas, TX, USA, 20-22 October 2003.

[20] Hamed M, Shukry S, El-Mahallawy MS, El-Ramly S. "Modified earliest deadline first scheduling with channel quality indicator for downlink real-time traffic in LTE networks". *3rd International Conference on e-Techologies and Networks for Development (ICeND),* Beirut, Lebanon, 29 April-1 May 2014.

[21] Demers A, Keshav S, Shenker S. "Analysis and Simulation of a Fair Queueing Algorithm". *ACM SIGCOMM Symposium*, Austin, TX, USA, 19-22 September 1989.

[22] Parekh AK, Gallager RG. "A generalized processor sharing approach to flow control in ıntegrated services networks: the single-node Case". *IEEE/ACM Transactions on Networking*, 1(3), 344-357, 1993.

[23] Banks J, Carson II JS, Nelson B, Nicol DM. *Discrete-Event System Simulation.* 5th ed. New Jersey, USA, Pearson, 2010.