Chapter 8

Modeling Quality of Service Techniques for Packet-Switched Networks

Wlodek M. Zuberek and Dariusz Strzeciwilk

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.71499

Abstract

Quality of service is the ability to provide different priorities to different applications, users or dataflows, or to guarantee a certain level of performance to a dataflow. The chapter uses timed Petri nets to model techniques that provide the quality of service in packet-switched networks and illustrates the behavior of developed models by performance characteristics of simple examples. These performance characteristics are obtained by discrete-event simulation of analyzed models.

Keywords: quality of service, packet-switched networks, timed Petri nets, priority queuing, fair queuing, weighted fair queuing, performance analysis, discrete-event simulation

1. Introduction

Quality of service (or simply QoS) is the ability to provide different priorities to different applications, users or dataflows, or to guarantee a certain level of performance to a dataflow [1]. For example, a computer network can guarantee certain levels of error rate or jitter, or maximal delay of transmitted information. Quality of service guarantees are important if the network capacity is insufficient, especially for real-time streaming multimedia applications such as voice over internet (voice over IP), TV over internet (TV over IP), or multimedia applications, since these are often delay sensitive and require fixed bit rate [1, 2]. Similarly, quality of service is essential in networks where the capacity can be a limiting factor, for example, in cellular data communication [3].

There are two principal approaches to QoS in modern packet-switched IP networks, an integrated services approach based on application requirements that are exchanged with the network, and a differentiated approach where each packet identifies a desired service level to the network [4]. Early networks used the integrated services approach. It was realized, however, that in broadband networks, the core routers are required to deal with tens of thousands of service requirements—the integrated services approach does not scale well with the growth of the internet. Routers supporting differentiated services configure their network schedulers to use multiple queues for packets awaiting transmission. In practice, packets requiring low jitter (e.g., voice over IP or videoconferencing) are given priority over packets of other types. Typically, some bandwidth is also allocated to network control packets, which must be sent over the network without any unnecessary delay.

Modern packet-switched networks are complex structures [5], which, for modeling, require a flexible formalism that can easily handle concurrent activities as well as synchronization of different events and processes that occur in such networks. Petri nets [6, 7] are such formal models.

As formal models, Petri nets are bipartite directed graphs, in which two types of vertices represent, in a very general sense, conditions and events. An event can occur only when all conditions associated with it (represented by arcs directed to the event) are satisfied. An occurrence of an event usually satisfies some other conditions, indicated by arcs directed from the event. So, an occurrence of one event causes some other event (or events) to occur, and so on.

In order to study performance aspects of systems modeled by Petri nets, the durations of modeled activities must also be taken into account. This can be done in different ways, resulting in different types of temporal nets. In timed Petri nets [8], occurrence times are associated with events, and the events occur in real time (as opposed to instantaneous occurrences in other models). For timed nets with constant or exponentially distributed occurrence times, the state graph of a net is a Markov chain (or an embedded Markov chain), in which the stationary probabilities of states can be determined by standard methods [9]. These stationary probabilities are used for the derivation of many performance characteristics of the model [10].

In this chapter, timed Petri nets are used to model priority queuing systems, which provide the quality of service in packet-switched networks. Section 2 recalls basic concepts of Petri nets and timed Petri nets. Section 3 discusses (strict) priority queuing and its performance characteristics. Fair scheduling is described and illustrated in Section 4, while Section 5 deals with weighted fair scheduling. A combined approach, using several types of scheduling methods, is outlined in Section 6. Section 7 concludes the chapter.

2. Timed Petri nets

In Petri nets, concurrent activities are represented by *tokens*, which can move within a (static) graphlike structure of the net. More formally, a marked inhibitor place/transition Petri net $\mathcal{M}$ is defined as a pair $\mathcal{M} = (\mathcal{N}, m_0)$, where the structure $\mathcal{N}$ is a bipartite-directed graph, $\mathcal{N} = (P, T, A, H)$, with two types of vertices, a set of places P and a set of transitions T, a set of directed arcs A connecting places with transitions and transitions with places, $A \subseteq T \times P \cup P \times T$, and a set of inhibitor arcs H connecting places with transitions, $H \subset P \times T$; usually $A \cap H = \emptyset$. The initial marking function m_0 assigns non-negative numbers of tokens to places of the net, $m_0 : P \to \{0, 1, \ldots\}$. Marked nets can be equivalently defined as $\mathcal{M} = (P, T, A, H, m_0)$.

A place is shared if it is connected to more than one transition. A shared place p is free choice if the sets of places connected by directed arcs and inhibitor arcs to all transitions sharing p are identical. A shared place p is (dynamically) conflict free if for each marking reachable from the initial marking at most one transition sharing p is enabled. A net is a free choice if all its shared places are either free choice or (dynamically) conflict free. Only free-choice nets are used in this chapter.

In timed nets [8], occurrence times are associated with transitions, and transition occurrences are real-time events; i.e., tokens are removed from input places at the beginning of the occurrence period, and they are deposited to the output places at the end of this period. All occurrences of enabled transitions are initiated in the same instants of time in which the transitions become enabled (although some enabled transitions may not initiate their occurrences). If, during the occurrence period of a transition, the transition becomes enabled again, a new, independent occurrence can be initiated, which will overlap with the other occurrence(s). There is no limit on the number of simultaneous occurrences of the same transition (sometimes, this is called infinite occurrence semantics). Similarly, if a transition is enabled "several times" (i.e., it remains enabled after initiating an occurrence), it may start several independent occurrences in the same time instant.

More formally, a free-choice timed Petri net is a triple, $\mathcal{T} = (\mathcal{M}, c, f)$, where $\mathcal{M}$ is a marked net, c is a choice function that assigns probabilities to transitions in free-choice classes, $c \to [0, 1]$, and f is a timing function that assigns an (average) occurrence time to each transition of the net, $f: T \to \mathbf{R}^+$, where $\mathbf{R}^+$ is the set of non-negative real numbers.

The occurrence times of transitions can be either deterministic or stochastic (i.e., described by some probability distribution function); in the first case, the corresponding timed nets are referred to as D-timed nets [11]; in the second, for the (negative) exponential distribution of occurrence times, the nets are called M-timed nets (Markovian nets) [12]. In both cases, the concepts of state and state transitions have been formally defined and used in the derivation of different performance characteristics of the model. In simulation applications, other distributions can also be used; for example, the uniform distribution (U-timed nets) is sometimes a convenient option. In timed Petri nets, different distributions can be associated with different transitions in the same model providing flexibility that is used in simulation examples that follow.

In timed nets, the occurrence times of some transitions may be equal to zero, which means that the occurrences are instantaneous; all such transitions are called immediate (while the others are called timed). Since the immediate transitions have no tangible effects on the (timed) behavior of the model, it is convenient to "split" the set of transitions into two parts, the set of immediate and the set of timed transitions, and to first perform all occurrences of the (enabled) immediate transitions, and then (still in the same time instant), when no more immediate transitions are enabled, to start the occurrences of (enabled) timed transitions. It should be noted that such a convention effectively introduces the priority of immediate transitions over the timed ones, and therefore conflicts of timed and immediate transitions are not allowed in timed nets. Detailed characterization of the behavior or timed nets with immediate and timed transitions is given in [8].

3. Priority queuing

The basic idea of priority scheduling [13] is that separate queues are used by servers for packets of different priority classes, as shown in **Figure 1** (there are three classes of priorities with queues Q1, Q2, and Q3, for example, for voice, video and data packets, respectively). It is assumed that priorities decrease with the queue numbers; i.e., Q1 is the highest priority queue. In **Figure 1**, a_1 is the arrival rate of packets in priority class 1, a_2 is the arrival rate in class 2, and s is the service rate; $1/s$ is the average transmission time of a packet over the communication channel (represented as the server in **Figure 1**). For simplicity of the description, it is assumed that the transmission times are (approximately) the same for packets of different classes, but this can easily be replaced by rates s_1, s_2, and s_3.

To select a packet for transmission, the scheduler first checks the highest priority queue (Q1) and if this queue is nonempty, the scheduler uses the first packet from this queue. If Q1 is empty, the scheduler checks the next queue in the order of priorities (i.e., Q2 and then Q3). Consequently, a lowest priority packet is transmitted only when all other (higher priority) queues are empty.

Figure 2 shows a Petri net model of priority queuing outlined in **Figure 1**. Transitions t_{01}, t_{02}, and t_{03} with places p_{01}, p_{02}, and p_{03} are (independent) sources of packets for classes 1, 2, and 3,

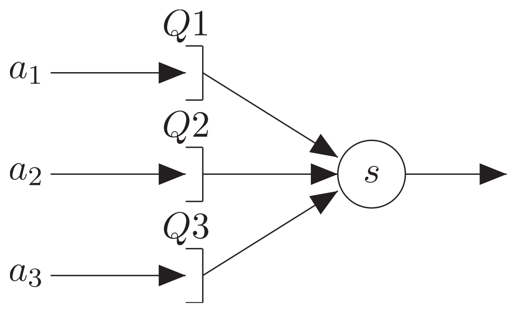

Figure 1. Priority queuing.

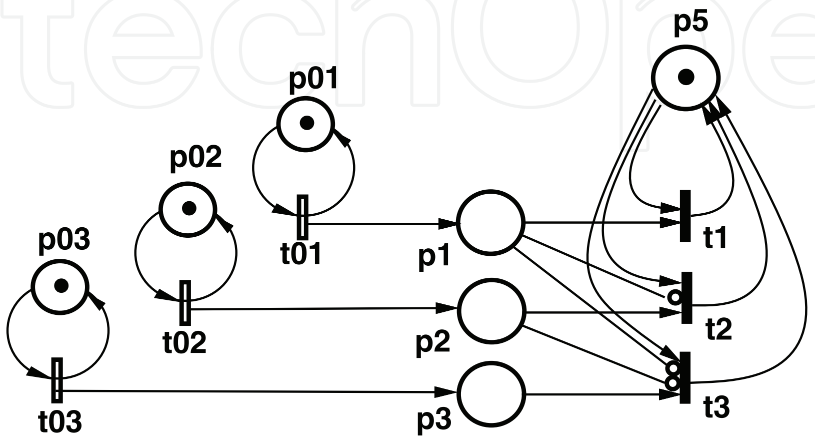

Figure 2. Timed Petri net model of priority queuing.

respectively. The rates of generated packets are determined by the occurrence times associated with t_{01}, t_{02}, and t_{03}; $a_1 = 1/f(t_{01})$, etc. Also, the mode of the timed transition (M, D, or U) determines the distribution of the interarrival times for the respective source.

Transitions t_1, t_2, and t_3 model the transmission times for packets of classes 1, 2, and 3, respectively. Place p_5, shared by these three transitions, guarantees that no two packets can be transmitted at the same time. Finally, the inhibitor arcs (p_1, t_2), (p_1, t_3), and (p_2, t_3) implement the priorities: if there is a packet of class 1 waiting for transmission (in p_1), no packet of class 2 or 3 can be transmitted; if there is a packet of class 1 or 2, no class 3 packet can be transmitted.

The priority scheme works well when the traffic intensity is small; when, however, the traffic becomes intensive, it is quite possible that bursts of packets of higher priorities can (temporarily) block the transmission of lower priority packets for extended periods of time, significantly degrading their performance. **Figure 3** shows the utilization of a transmission channel shared by three streams of packets (as in **Figure 1**) as a function of traffic intensity of packets of priority 1, ρ_1, with fixed traffic intensities for packets of priorities 2 and 3 (at the values of $\rho_2 = 0.5$ and $\rho_3 = 0.25$).

It can be observed that for traffic intensity $\rho_1 > 0.25$, i.e., $\rho_1 > 1.0 - \rho_2 - \rho_3$, channel utilization for packets of priority 3 decreases to zero; for $\rho_1 \geq 0.5$, packets of priority 3 are blocked. Similarly, for $\rho_1 > 0.5$, the utilization of the channel for packets of priority 2 decreases, which means that only some of such packets can be send through the channel.

Waiting times of packets with priorities 1, 2, and 3, for the case shown in **Figure 3**, are presented in **Figure 4**. As the traffic intensity ρ_1 approaches 0.25, waiting times for packets of priority 3 increase indefinitely, and for ρ_1 approaching 0.5, so do waiting times of packets of priority 2.

It should be noted that for $\rho_1 > 0.25$, queue Q3 (with infinite capacity) is nonstationary as the rate of packets entering the queue is greater than the rate of packets removed from it (for transmission through the channel). Similarly, queue Q2 (with infinite capacity) is also nonstationary for

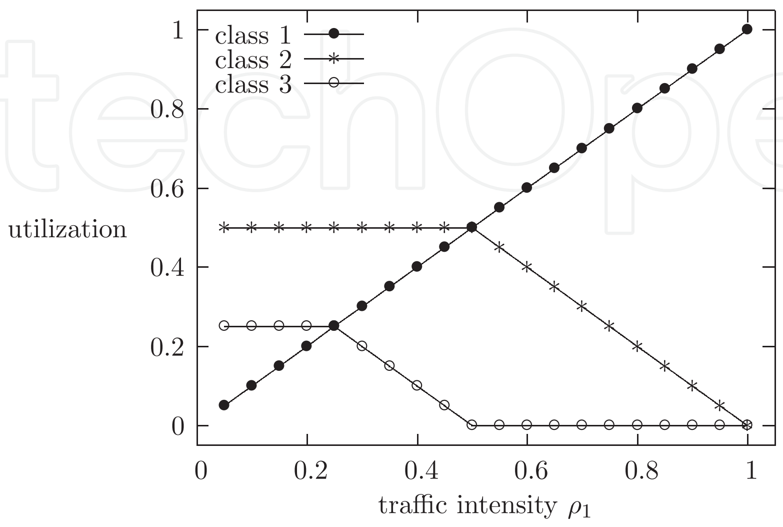

Figure 3. Channel utilization as a function of ρ_1 with $\rho_2 = 0.5$ and $\rho_3 = 0.25$.

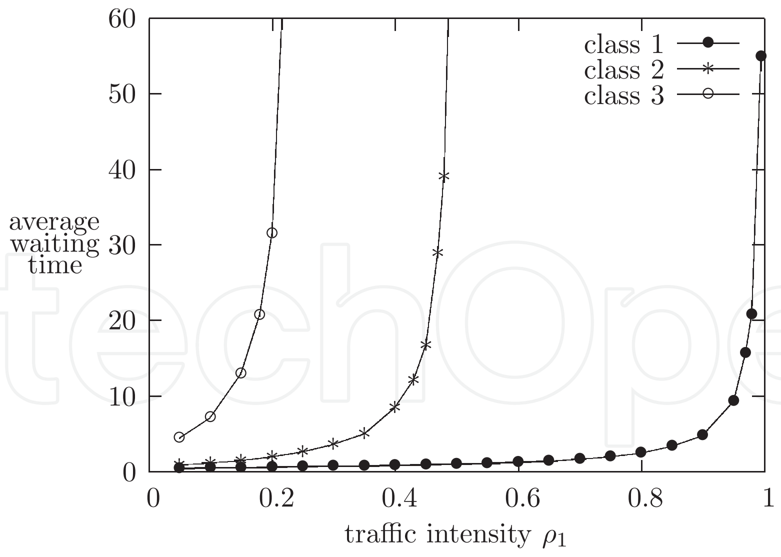

Figure 4. Average waiting times as functions of traffic intensity ρ_1 with $\rho_2 = 0.5$ and $\rho_3 = 0.25$.

$\rho_1 > 0.5$. This is the reason that, in **Figure 4**, waiting times are shown only for stationary regions of behavior.

4. Fair queuing

In fair queuing [14], a transmission medium is shared (in a fair way, i.e., in "equal parts") by all classes of traffic (and the classes of traffic correspond to different packet flows, e.g., voice over IP, video, etc.). Implementations of fair queuing use separate queues for different classes of traffic, and packets are forwarded from these queue in a cyclic way, providing the same service for all classes.

Timed Petri nets model of fair queuing with three classes of traffic (as in **Figure 1**) is shown in **Figure 5**.

Places p_1, p_2, and p_3 are queues for classes 1, 2, and 3, respectively. If the queues for all classes are nonempty, the selection is performed cyclically in a loop:

$$p_{31}, t_{11}, p_{10}, t_{10}, p_{12}, t_{22}, p_{20}, t_{20}, p_{23}, t_{33}, p_{30}, t_{30}, p_{31}.$$

This loop is modified if (at the selection time) some queues are empty. For example, if after selecting a packet of class 1 packet, the queue for class 2 is empty, while the queue for class 3 is nonempty, the sequence is:

$$p_{31}, t_{11}, p_{10}, t_{10}, p_{12}, t_{32}, p_{30}, t_{30}, p_{21}.$$

If the only nonempty queue is the queue for class 1, the sequence is

$$p_{31}, t_{12}, p_{10}, t_{10}, p_{31}.$$

Generally, there are three cases when packets are selected from (nonempty) queue 1:

- queue 1 is checked in the order of fair queuing (marked place p_{31}) and the queue is nonempty (i.e., transition t_{11} can occur),
- queue 3 is checked in the order of fair queuing (marked place p_{23}), but the queue is empty so queue 1 is checked as the next one and it is nonempty (i.e., transition t_{13} can occur),
- queue 2 is checked in the order of fair queuing, but queue 2 as well as the next queue, queue 3, are empty, while queue 1 is nonempty (i.e., transition t_{12} can occur).

There are similar cases for queues 2 and 3.

It should be observed, that the set of places:

$$\{p_{31}, p_{12}, p_{23}, p_{10}, p_{20}, p_{30}\}$$

always contains a single token (initially shown in p_{31} in **Figure 5**). The cyclic checking of queues of fair queuing is controlled by this token.

Figure 6 shows the average waiting times for a system with fair queuing and three classes of traffic when the traffic intensities are the same for all three classes. Since the service rates are

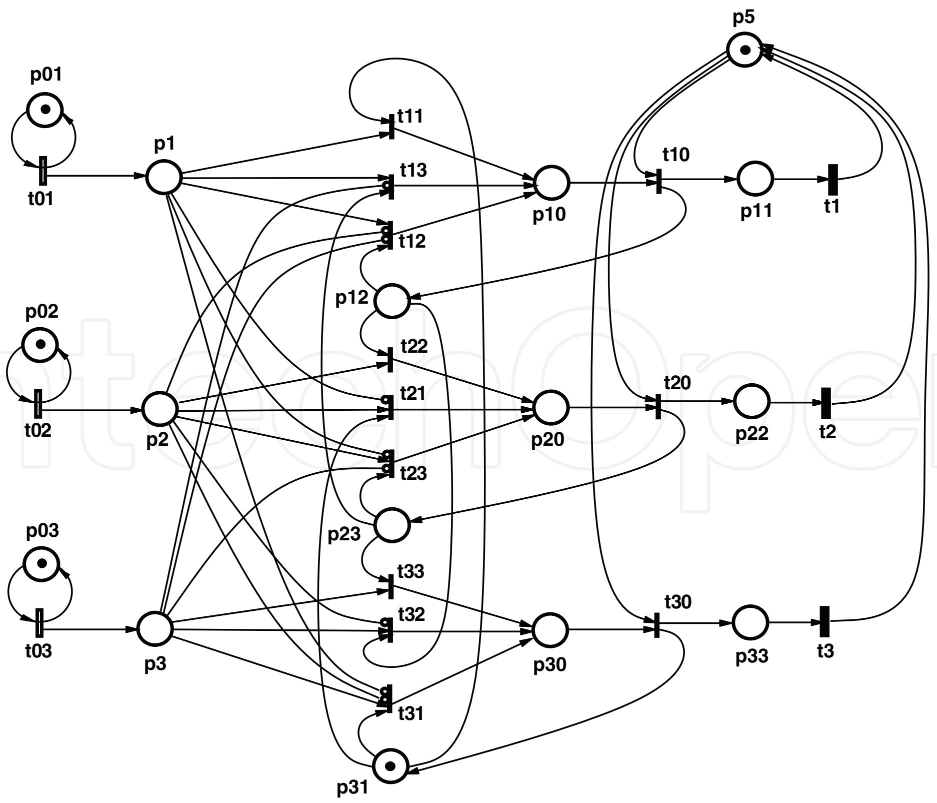

Figure 5. Timed Petri net model of fair queuing.

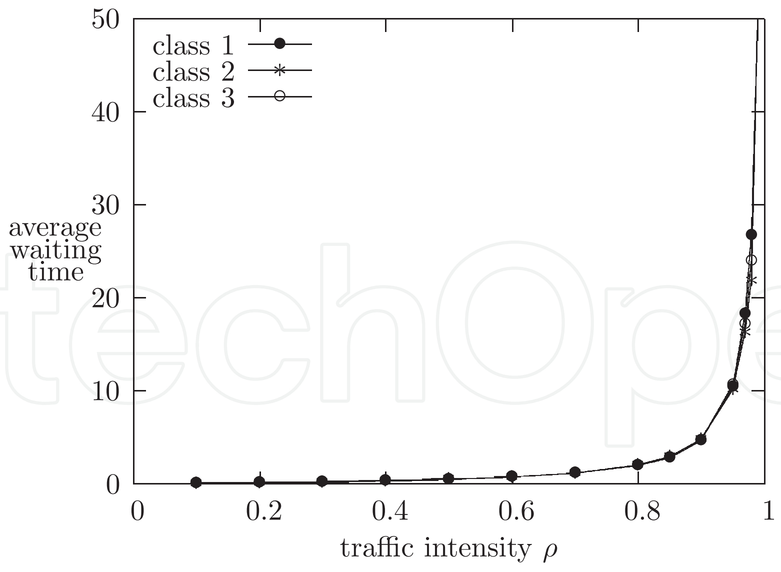

Figure 6. Average waiting times as functions of traffic intensity ρ with $\rho_1 = \rho_2 = \rho_3$.

also the same for all classes, channel utilizations as well the average waiting times for are the same in this case for all classes of traffic.

If, however, traffic intensities are different for different classes, average waiting times as well as utilizations of the shared transmission channel are also different. **Figure 7** shows the average waiting times for the case when $\rho_1 = 0.5\rho$, $\rho_2 = \rho_3 = 0.25\rho$.

In this case, the average waiting times for traffic classes with greater traffic intensity (class 1) increase much faster with increasing traffic intensity than for other classes of traffic (classes 2 and 3).

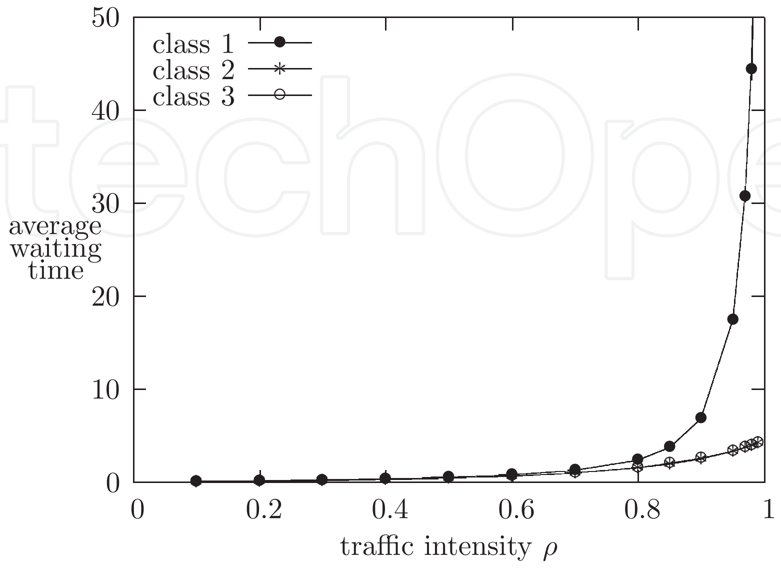

Figure 7. Average waiting times as functions of traffic intensity ρ with $\rho_1 = 0.5\rho$, $\rho_2 = \rho_3 = 0.25\rho$.

5. Weighted fair queuing

Weighted fair scheduling [13] restricts the priority scheduling by introducing limits on the number of consecutive packets of the same class that can be transmitted over the channel; when the scheduler reaches such a limit, it switches to the next nonempty priority queue and follows the same rule. So, if there are sufficient supplies of packets in all priority classes, the scheduler selects w_1 packets of class 1, then w_2 packets of class 2, then w_3 packets of class 3, and again w_1 packets of class 1, and so on, where w_1, w_2, and w_3 are the weights for classes 1, 2 and 3, respectively. Consequently, in such a situation (i.e., for sufficient supply of packets in all classes), the channel is shared by the packets of all priority classes, and the proportions are

$$u_i = \frac{w_i/s_i}{\sum_{j=1,\ldots,k} w_j/s_j}, i = 1, 2, \ldots k, \tag{1}$$

where k is the number of priority classes and s_i, $i = 1, \ldots, k$, is the transmission rate for packets of class i. If the transmission rates are the same for packets of all priority classes (as is assumed for simplicity in the illustrating examples), the properties are

$$u_i = \frac{w_i}{\sum_{j=1,\ldots,k} w_j}, i = 1, \ldots, k. \tag{2}$$

For an example with three priority classes and the weights equal to 4, 2, and 1 for classes 1, 2, and 3, respectively, these "utilizations bounds" are equal to 4/7, 2/7, and 1/7, for classes 1, 2, and 3, respectively.

A Petri net model of weighted fair scheduling for three priority classes with weights 4, 2, and 1 is shown in **Figure 8**. The model is composed of three identical interconnected sections corresponding to the three priority classes. The main elements of the model are the three queues represented by places p_1, p_2, and p_3 for classes 1, 2 and 3, respectively, and timed transitions t_1, t_2, and t_3 modeling the transmission of selected packets through the communication channel. The three classes of packets are generated (independently) by transitions t_{01}, t_{02}, and t_{03} with places p_{01}, p_{02}, and p_{03}.

As in fair queuing, the scheduling is based on cyclic selection of queues for the transmission of waiting packets. This cyclic operation is represented by a (rather complex) loop with places r_1, r_2, and r_3; q_1, q_2, and q_3; and also s_1, s_2, and s_3. There is a single "control token" in this loop (shown in place s_3 in **Figure 8**). This token always indicates the queue that is used for transmission of packets.

The section for class 2 is shown separately in **Figure 9** to make its description easier to follow.

Place r_2 becomes marked only when nonempty queue 2 is used for the selection of packets. Place w_2 contains the weight of class 2 (in this case 2). Transition a_2 selects a packet (from p_2) and forwards it to p_{20}, moving a single token from w_2 to u_2. When the channel becomes available (i.e., p_5 becomes marked), the selected packet is forwarded from p_{20} to p_{22} and then is transmitted (transition t_2). At the same time, a token is returned by t_{20} to r_2 to allow selecting another packet from p_2. This is repeated until:

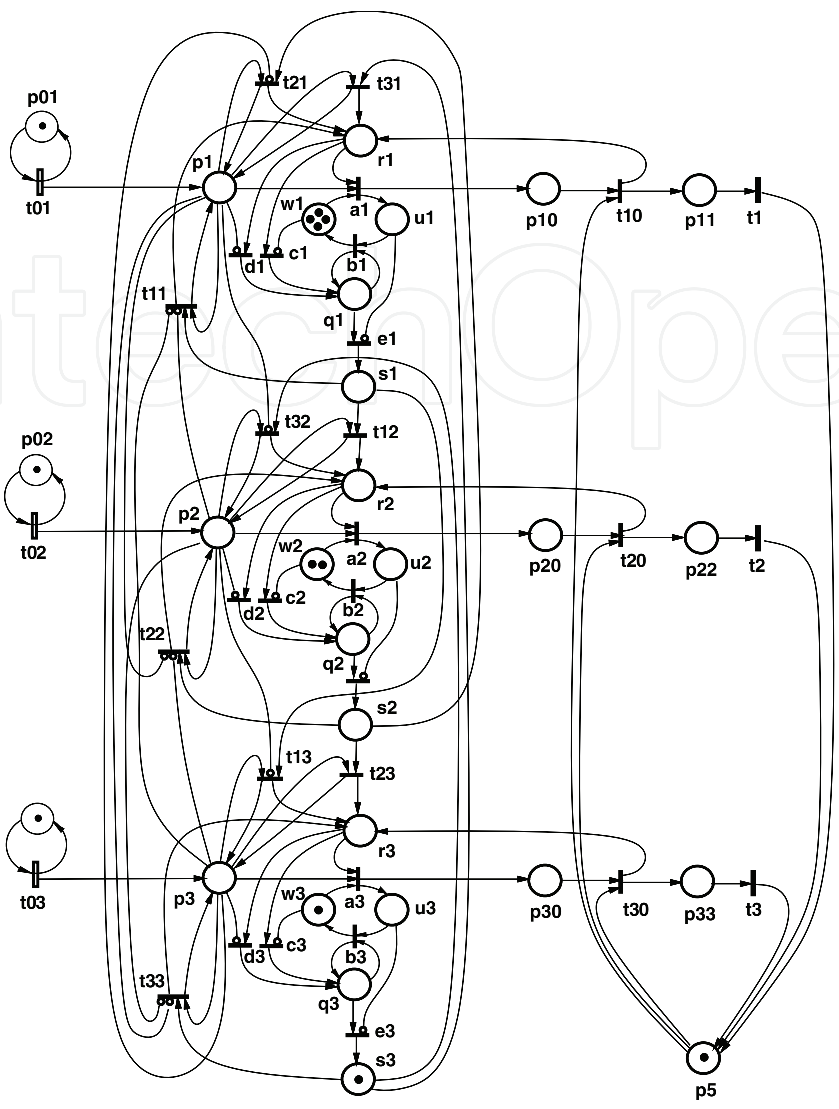

Figure 8. Petri net model of weighted fair queuing with weights 4-2-1.

- there are no more token in w_2 (transition c_2 occurs), or
- there are no more token in p_2 (transition d_2 occurs).

In both cases, the token from r_2 is moved to q_2 and then a number of occurrences of b_2 moves all tokens form u_2 back to w_2. When u_2 becomes unmarked, transition e_2 moves the token from q_2 to s_2 in order to select the next traffic class. If p_3 is nonempty, transition t_{23} moves the token from s_2 to r_3. If p_3 is unmarked and p_1 is marked, transition t_{21} moves the token from s_2 to r_1. If both p_1 and p_3 are unmarked but p_2 is marked, transition t_{22} moves the token from s_2 to r_2 and transmission of class 2 packets continues. Finally, if none of t_{21}, t_{22}, and t_{23} is enabled, the token remains in s_2 waiting for a packet arriving to p_1, p_2, or p_3.

It should be observed that when the traffic is intense, i.e., when the queues p_1, p_2, and p_3 are nonempty most of the time, the queuing mechanism repeatedly selects w_1 packets from p_1, then w_2 packets from p_2, then w_3 packets from p_3, and so on. On the other hand, when the traffic is light, all packets are transmitted with very little delay.

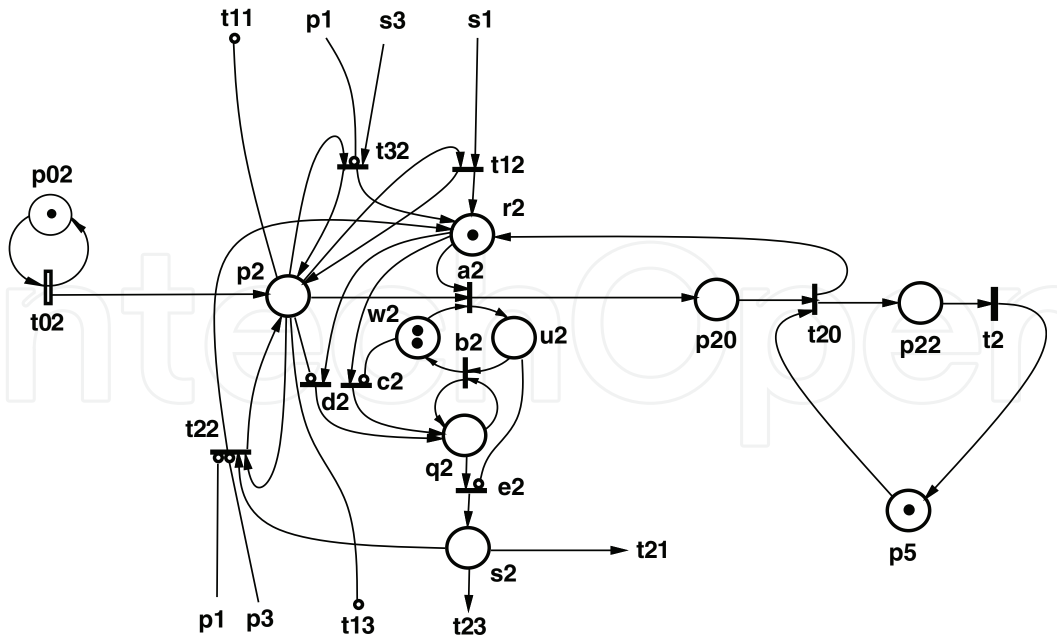

Figure 9. Petri net model of class 2 of weighted fair queuing.

Figure 10 shows the utilization of the transmission channel shared by three classes of packets (as in **Figure 1**) as a function of traffic intensity of traffic class 1, ρ_1, with fixed traffic intensities for traffic classes 2 and 3 (at the values of $\rho_2 = 0.5$ and $\rho_3 = 0.25$).

For $\rho_1 > 0.25$, channel utilization for packets of priority 3 decreases from the initial value of 0.25 to its weighted value of 0.14 (i.e., 1/7). Also, channel utilization for packets of priority 2 decreases from its initial value of 0.5 to its weighted value of 0.28 (i.e., 2/7). At $\rho_1 = 0.57$ (i.e., 4/7), the total utilization of the channel becomes 100% and no further increase of utilization is possible.

Similarly as before (**Figures 3** and **4**), queues Q2 and Q3 are nonstationary for $\rho_1 > 0.25$ and queue Q1 is nonstationary for $\rho_1 > 0.57$ (i.e., 4/7).

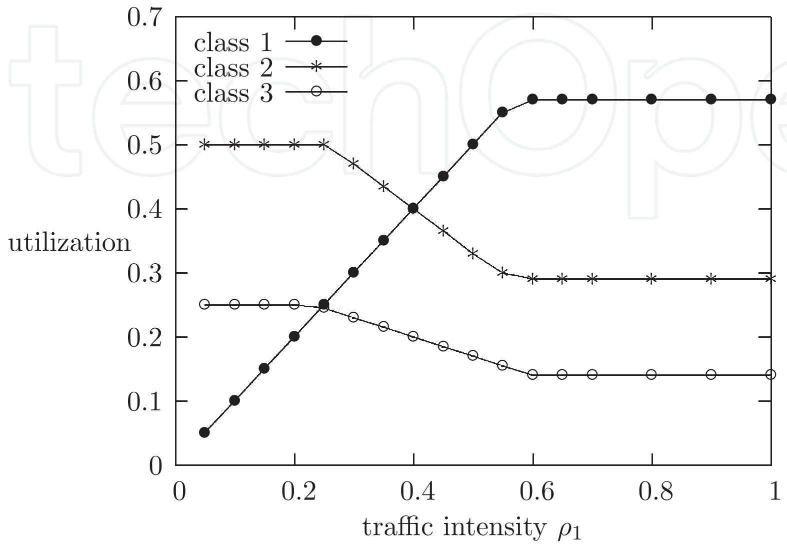

Figure 10. Channel utilization as a function of ρ_1 with $\rho_2 = 0.5$ and $\rho_3 = 0.25$.

For the same weights but for different (fixed) arrival rates for classes 2 and 3, i.e., for $\rho_2 = 0.25$ and $\rho_3 = 0.1$, the utilization of the transmission channel as a function of traffic intensity of traffic class 1 is shown in **Figure 11**.

For this case, queues Q2 and Q3 are stationary for $0 \le \rho_1 < 1$ and Q3 is nonstationary for $\rho_1 > 0.65$ (i.e., 1.0–0.25–0.1). The average waiting times for classes 2 and 3 depend in a limited way on the traffic intensity ρ_1, as shown in **Figure 12**.

In weighted fair queuing, if weights are equal to the arrival rates, the traffic of lower priority classes is (almost) independent of the traffic intensity of higher-priority classes; the effect is

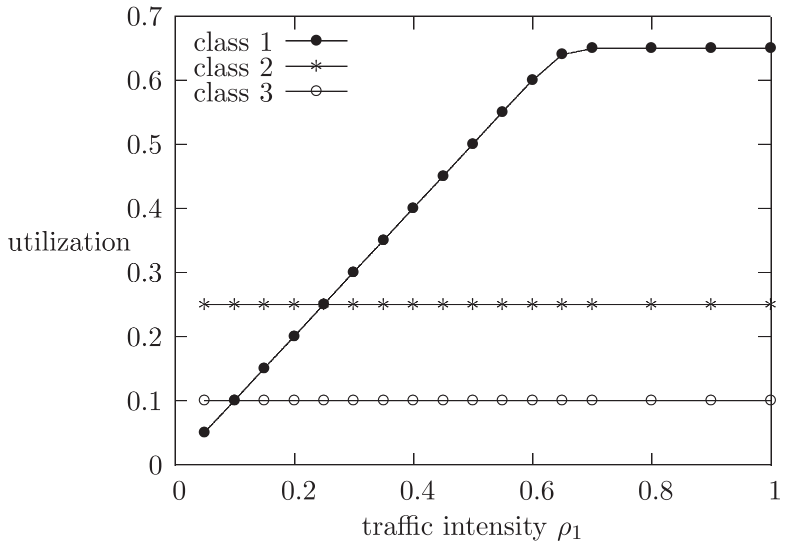

Figure 11. Channel utilization as a function of ρ_1 with $\rho_2 = 0.25$ and $\rho_3 = 0.1$.

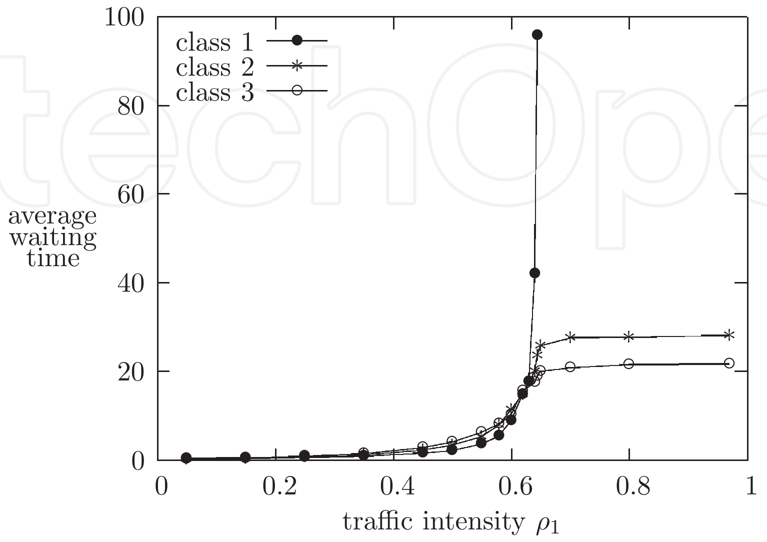

Figure 12. Average waiting times as functions of ρ_1 with $\rho_2 = 0.25$ and $\rho_3 = 0.1$.

similar to assigning some (shared) transmission capacity to lower priority traffic classes. Moreover, if this "reserved" capacity is not used, it is available to other classes of traffic.

6. Combined queuing

The basic queuing methods discussed earlier can be combined into more complex systems. For example, the combination of priority queuing and weighted fair queuing is known as low-latency queuing (LLQ) [15]. A simpler case of combining priority queuing and fair queuing, as shown in **Figure 13**, is used as an illustration of the combined approach.

It is assumed in this example that 40% of bandwidth is allocated to the priority queue (Q1) and that the remaining 60% of bandwidth is equally divided among queues Q2, Q3 and Q4.

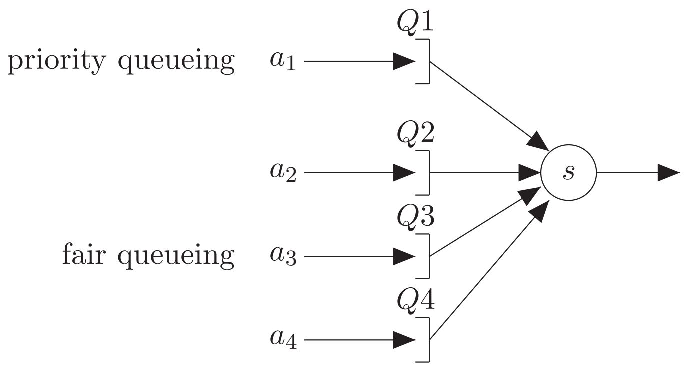

Figure 13. Priority queuing combined with fair queuing.

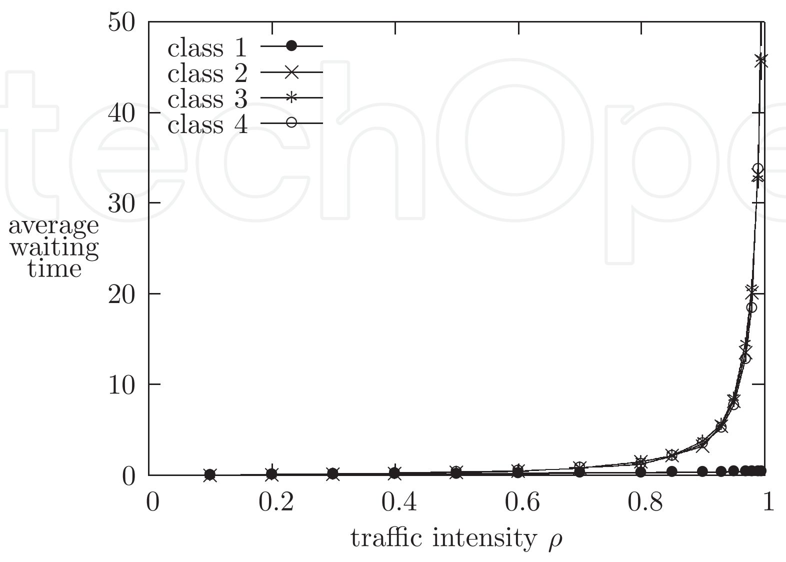

Figure 14. Average waiting times as functions of ρ with $\rho_1 = 0.4\rho$ and $\rho_2 = \rho_3 = \rho_4 = 0.2\rho$.

Average waiting times for all four classes of traffic (**Figure 13**) as functions of traffic intensity are shown in **Figure 14**.

Figure 14 shows the effects of priority scheduling (class 1) on lower priority classes (classes 2, 3, and 4) when traffic intensity approaches 1—the average waiting times increase rather significantly for classes 2, 3, and 4 while class 1 remains practically unaffected by the increased traffic. Also, because of fair queuing, the average waiting times for classes 2, 3, and 4 are practically identical.

7. Concluding remarks

The Internet 2 project, launched in 2001, was probably too early for implementation of QoS protocols with the equipment that was then available [16]. It should not be surprising that this resulted in a conclusion that adding more bandwidth (i.e., over-provisioning) is more effective than any of the various schemes for accomplishing QoS [17]. But cost and other factors prevent service providers to built and maintain permanently over-provisioned networks; other approaches must be used to guarantee the performance of services available in modern packet-switched networks [16].

Several models of techniques used for providing quality of service in packet-switched networks are discussed in this chapter. These models are used to derive performance characteristics of scheduling methods and to provide some insights into the behavior of packet-switched networks. In particular, the blocking of lower priority classes of traffic, typical for priority scheduling, can easily be observed. Also, the guaranteed levels of service of fair scheduling and weighted fair scheduling can easily be illustrated.

It should be noted, however, that the discussed techniques are just basic elements of complex computer networks and that the behavior of real systems is very dynamic and usually difficult to predict [18]. Therefore, more work is needed in this area to use the networks in an efficient and predictable way.

Also, an attractive aspect of the models would be some kind of compositionality that would allow models to be easily combined into more complex ones, as outlined in Section 6. The models presented in this chapter need to be revised to make such compositions straightforward.

Author details

Wlodek M. Zuberek[1]* and Dariusz Strzeciwilk[2]

*Address all correspondence to: wlodek@mun.ca

1 Department of Computer Science, Memorial University, St. John’s, NL, Canada

2 Department of Applied Informatics, University of Life Sciences, Warszawa, Poland

References

[1] Guerin R, Peris V. Quality of service in packet networks: Basic mechanisms and directions. Computer Networks. 1999;**31**:169-189

[2] Wang Z. Internet QoS: Architectures and Mechanisms for Quality of Service. San Francisco, CA: Morgan Kaufmann; 2001

[3] Marchese M. QoS over Heterogeneous Networks. Chichester: Wiley and Sons; 2007

[4] Fergusson P, Huston G. Quality of Service: Delivering QoS on the Internet and in Corporate Networks. New York, NY: Wiley and Sons; 1998

[5] Robertazzi TG. Computer Networks and Systems: Queueing Theory and Performance Evaluation. Berlin, Heidelberg: Springer-Verlag; 1990

[6] Murata T. Petri nets: Properties, analysis and applications. Proceedings of IEEE. 1989;**77**(4): 541-580

[7] Reisig W. Petri Nets — An Introduction (EATCS Monographs on Theoretical Computer Science 4). Berlin, Heidelberg: Springer-Verlag; 1985

[8] Zuberek WM. Timed Petri nets — Definitions, properties and applications. Microelectronics and Reliability (Special Issue on Petri Nets and Related graph models). 1991;**31**(4): 627-644

[9] Allen AA. Probability, Statistics and Queueing Theory with Computer Science Applications. 2nd ed. San Diego, CA: Academic Press; 1991

[10] Jain R. The Art of Computer Systems Performance Analysis. Berlin, Heidelberg: Springer-Verlag; 1991

[11] Zuberek WM. D-timed Petri nets and modelling of timeouts and protocols. Transactions of the Society for Computer Simulation. 1987;**4**(4):331-357

[12] Zuberek WM. M-timed Petri nets, priorities, preemptions, and performance evaluation of systems. In: Advances in Petri Nets 1985 (LNCS 222). Berlin, Heidelberg: Springer-Verlag; 1986. p. 478-498

[13] Georges P, Divoux T, Rondeau E. Strict priority versus weighted fair queueing in switched Ethernet networks for time-critical applications. In: Proc. 19-th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05); 2005. pp. 141-145

[14] Park KI. QoS in Packet Networks. Boston, MA: Springer Science; 2005

[15] Dekeris B, Adomkus T, Budnikas A. Analysis of QoS assurance using weighted fair queueing (WFQ) scheduling with low latency queue (LLQ). In: Proceedings of 26-th Int. Conference on Information Technology Interfaces (ITI'06); 2006. pp. 507-512

[16] Lindgren A, Almquist A, Schelen O. Quality of service for IEEE 802.11—A simulation study. In: Quality of Service – IWQoS 2001 (LNCS 2092). Berlin, Heidelberg: Springer-Verlag; 2001. p. 281-287

[17] Brachman A, Miszczanin J. Scheduling algorithms for different approaches to quality of service provisioning. In: Computer Networks 2011 (CCIS 160). Berlin, Heidelberg: Springer-Verlag; 2011. p. 135-143

[18] Tannenbaum AS. Computer Networks. 4th ed. Englewood Cliffs, NJ: Prentice-Hall; 2003