energies

MDPI

Article

Timed Petri Nets for Modeling and Performance Evaluation of a Priority Queueing System

Dariusz Strzęciwilk

Institute of Information Technology, Warsaw University of Life Sciences—SGGW, Nowoursynowska Str. 159, 02-776 Warsaw, Poland; dariusz_strzeciwilk@sggw.pl

Abstract: The application of queueing theory is very broad. Examples include electronic communication systems and devices. New technologies, electronic communication systems, and devices are used by many modern organizations. However, this implies certain requirements and risks. The requirements are, first and foremost, reliability, which accounts for the complexity and interdependence of the system. On the other hand, the stochastic characteristics and complexity of these systems introduce risks related to the demands of reliability control, transmission quality, availability, and security. The research conducted so far is concerned with relatively simple queueing models that require certain assumptions to be made about the stochastic nature of the event stream. This is because complex queueing systems are very difficult to analyze using analytical methods. Hence, this paper attempts to use timed Petri nets in the modeling and performance evaluation of queueing systems belonging to the PQS (Priority Queueing System) group. IntServ and DiffServ architectures are discussed, as well as queueing systems used in quality-of-service assurance. A weighted PQS that eliminates the possibility of blocking lower-priority traffic is investigated. Based on a Petri model, the performance characteristics of the studied system are obtained. The impact of data generation on the system performance was analyzed, showing that temporal Petri nets can be effectively used in the modeling and performance evaluation of PQS systems.

Keywords: Petri nets; quality of service; IntServ; DiffServ; QoS; priority queueing system; performance modeling and analysis

Citation: Strzęciwilk, D. Timed Petri Nets for Modeling and Performance Evaluation of a Priority Queueing System. *Energies* **2023**, *16*, 7690. https://doi.org/10.3390/en16237690

Academic Editor: Alberto Reatti

Received: 22 November 2022
Revised: 3 September 2023
Accepted: 9 November 2023
Published: 21 November 2023

1. Introduction

Queueing theory applications cover a wide range of businesses. Examples include electronic communication systems and devices. New technologies, electronic communication systems, and devices are used by many organizations today. This is because they enable sustainable and uninterrupted growth and are a prerequisite for competitive advantage. However, this implies certain requirements and risks. The requirements are, first and foremost, reliability that takes into account the complexity and interdependence of the system. On the other hand, however, the stochastic characteristics and complexity of systems carry risks related to the requirements of reliability control [1], transmission quality [2], availability, and security [3]. In research work on electronic communication systems and devices, queueing models in which events flow into queues in a Markovian or non-Markovian manner are often used. Here, queueing theory makes use of the mathematical apparatus associated with the theory of stochastic processes, in particular Markov processes [4,5]. Queueing theory is also a useful tool for estimating capacity requirements or managing demand for any system where the demand time for the system's services is random. Recent research on electronic communication systems supported by queueing theory indicates a significant development potential in many areas, such as industry, business, transport, energy, and medicine. Many organizations, such as banks, airlines, healthcare systems, telecommunications companies, and security departments, routinely use queueing theory models to help determine capacity levels needed to meet experienced demands in a more

efficient way. An analysis of the literature on the subject shows that studies and applications relating to service control for queueing systems are rich and varied. In a queueing system, the elements arriving to receive a service can refer to a variety of phenomena. These could be customers arriving at a system, such as a bank, patients arriving at a hospital, items in a partially completed production component, or transport vehicles, planes, or cars waiting to be served by the system. However, queueing theory is a tool that is heavily used in modeling and testing the quality of transmission in electronic communication systems and devices [6–8]. QoS (Quality of Service) is one of the most important challenges that arise in the design and maintenance of NGNs (Next-Generation Networks) [9–12]. The interconnection between embedded devices, sensors, and machines over the Internet has revolutionized the way people live and work. Sustaining organizational growth and increasing capacity requires the introduction of new intelligent technology solutions; hence, different techniques are proposed in the literature to improve the current network architectures [13]. These requirements are particularly important with the development of new smart technologies, such as IoT (Internet of Things) [14] and IoE (Internet of Everything) [15]. IoT and IoE technologies embody a fantastic vision of the future where everything is connected to the internet. This technological revolution that is ushering us into a new era of ubiquitous connectivity, computing, and communication is primarily driven by the continuous, relentless demand for bandwidth, mobile communications of higher data rates, ultra-low latency, high reliability, and QoS guarantees for systems connecting billions of devices, machines, people, and intelligent processes. The queueing systems that inspired this article are queueing systems that support QoS, particularly found in electronic communication devices and advanced telecommunications systems, which manage intelligent data flow in networks.

This publication is divided into 10 main sections. Section 1 provides an introduction to the topic and research area. Section 2 discusses QoS functions and areas for which QoS provision is critical. Section 3 discusses the performance of modern networks based on QoS architectures. Two architectures are discussed here. The first architecture is IntServ and the second is DiffServ. Section 4 describes the means and methods of data markings used in the architectures discussed. Section 5 describes the queueing systems used in the QoS architectures. Section 6 discusses the parameters of the queueing system. Section 7 concludes with the theoretical foundations used to build models of PQS systems; it discusses the topic of Petri nets and defines timed Petri nets that illustrate an example model. Section 7 summarizes the research context used to build the actual PQS models described in the following sections. Section 8 describes the designed and tested models of the priority queueing system. Section 9 presents the performance characteristics of the tested models obtained through simulation using software tools. Section 10 provides a summary.

2. QoS Functions and Application Areas

Quality of service is the ability to provide different priorities to different applications, users, or dataflow, or to guarantee a certain level of performance to a dataflow [16]. Intensive research work on QoS has been carried out for many years [17,18]. Several researchers have already reviewed the state-of-the-art and challenges of QoS [19]. As the results of QoS research show, QoS is one of the most important challenges that arise in the design and maintenance of NGNs [20] and modern TSN (Time- Sensitive Network) [21]. Although QoS research has received sustained attention long before the advent of cloud computing, QoS is fundamental in this area as well. Adequate implementation of cloud services and cloud computing in enterprises can reduce the operational costs of organizations, but one of the main challenges posed by cloud applications is the management of QoS. Ardagna, D. et al. [22] argue that QoS is fundamental for cloud users, who expect providers to deliver the advertised quality characteristics, and for cloud providers, who need to find the right trade-off between QoS levels and operational costs. The findings of Abdelmaboud et al. [23] also confirm that QoS in cloud computing has become an important research topic in which

open challenges and gaps remain, which require exploration and future research. With the increasing use of services in a cloud environment, QoS improvements are, therefore, needed for cloud computing solutions. Cloud computing technologies are as important as the IoT, which is a new paradigm that is gaining immense popularity in today's world. However, the success of the IoT world requires the provision of services that are credited with ubiquity, reliability, high performance, efficiency, and scalability. The IoT is one of the most disruptive technologies for wireless systems due to its ability to provide connectivity to anything, anytime, anywhere. However, existing heterogeneous IoT solutions cannot handle the highly diverse QoS requirements with the increasing number of IoT applications. Conclusions and a detailed assessment of the current state of the art on proposed approaches to QoS issues in IoT were presented by White et al. [24]. Another area of research carried out on QoS focuses on WSN (Wireless Sensor Network) architectures [25]. The applications of WSNs are numerous and can be used to monitor areas such as the environment, health/healthcare, logistics applications, and smart grids. The increasing use of wireless sensors in a variety of areas makes QoS a critical parameter for determining their application. However, due to their dynamic architectural nature, it is difficult to ensure an adequate level of QoS in WSN architectures. An overview of methods and mechanisms for implementing various QoS metrics and enhanced communication capabilities in WSNs was presented by Letswamotse et al. [26]. Mechanisms for efficient resource management for QoS improvement were investigated and critical design and implementation aspects for QoS requirements and guarantees in WSNs were identified. Based on the analysis of the available research work, it can be concluded that QoS requirements are critical for many services and applications and are still a broad area of intensive research work.

3. QoS Architectures

The structure of the internet today is based on a multitude of heterogeneous networks working with different techniques and technologies. The internet today consists of a vast collection of networks, intricately layered and bridged, including instances where they are composed with themselves [27]. For communication between applications running on such a network, an IP (Internet Protocol) is used, which is designed to achieve maximum simplicity and scalability even at the expense of compromising network performance and QoS. Analysis of the IP protocol indicates that there are no mechanisms responsible for QoS that can be effectively applied to the operation of applications [28]. The lack of any guarantees or QoS levels on the internet is considered one of the main limitations of internet usage. Therefore, the aspiration of many designers is to develop a technical infrastructure that could enable the efficient transmission of information related to the various classes of services. The fundamental difficulty in developing such a technical infrastructure lies in guaranteeing the proper quality of information transmission for each of the classes of services offered (in other words, ensuring adequate QoS for the applications). Hence, QoS architectures for NGNs are being studied by the most important standardization organizations. As part of the research conducted by the IETF (Internet Engineering Task Force) on QoS architectures for IP networks, two architectures have been defined that allow traffic to be divided and handled into classes with differentiated QoS. These architectures are as follows:

- Integrated Services architecture—IntServ (Integrated Services) [29];
- Differentiated Services architecture—DiffServ (Differentiated Services) [30];

The introduction of these architectures has made it possible to extend the default BE (Best Effort) model [31] with new classes of service, enabling the delivery of services with the desired QoS. The BE model is the dominant one in all IP networks connecting to the internet. As the IP protocol provides a non-guaranteed (BE), connectionless data transfer service, the BE model treats all packets the same, regardless of their content. This means that the BE model does not actively differentiate the treatment of services passing through the network. In a BE-based network, all IP packets are treated identically. Consequently, parameters like bandwidth utilization, delay, or packet loss are unpredictable

in this model. Since the modern internet structure operates primarily on the BE model, this architecture must be enhanced with mechanisms that guarantee the QoS necessary for the correct operation of applications, especially critical ones. The IntServ and DiffServ architectures allow for extending the default BE model used on the internet with models that support QoS.

3.1. Integrated Services Architecture

The integrated services architecture was the first network model with a QoS guarantee developed by the IETF organization. It assumes that resources are reserved for individual or aggregated data streams. The architecture ensures that specific data streams are provided with E2E QoS (end-to-end QoS), which is required by applications, taking into account the time-critical or real-time services for which the guaranteed service is intended. The IntServ architecture uses 'resource reservation' and 'access control' mechanisms as key elements to establish and maintain QoS. The IntServ model is also referred to as the *hard QoS* model and is designed for demanding applications that require constant, dedicated bandwidth to ensure an adequate level of service to the recipients. The service level is defined by the different service classes assigned to this model. The following service classes are defined in the IntServ model:

- Guaranteed Service (GS) [32];
- Controlled Load Service (CLS) [33];
- Unclassified Service Best Effort (BE) [31];

The GS and CLS classes assigned to the model define how data streams are to be managed in a single transmission node of the IntServ architecture. The GS class provides guaranteed transmission bandwidth, no packet loss due to queueing at the device nodes, and a maximum end-to-end delay for transmitted packets. The GS class only controls the maximum delay; it does not manage the minimum delay or minimize jitter. The delay present in the IntServ model is a component of fixed delay and queueing delay. The fixed delay is a property of the path and is determined by the RSVP mechanism during the path setup, while the queueing delay is determined by the GS class. As reported by the authors of [32], the value of the end-to-end delay bound can be described by Equation (1):

$$D = \begin{cases} \frac{(b-M)(p-R)}{R(p-R)} + \frac{(M+C_{tot})}{R} + D_{tot}, & p > R \geq r \\ \frac{(M+C_{tot})}{R} + D_{tot}, & r \leq p \leq R \end{cases} \quad (1)$$

where r—token bucket rate (in bytes/second); b—token bucket size (in bytes); p—peak rate (in bytes/second); M—maximum datagram size (in bytes); R—flow service rate or bandwidth (in bytes/second); C_{tot}—end-to-end calculation of C; D_{tot}—end-to-end calculation of D; The GS class is based on a model following the token bucket algorithm to ensure that the queue delay of each packet in the flow is less than the total delay computed along the flow path. The total delay computed along the flow path, D_R, can be described by Equation (2):

$$D_R = \frac{b}{R} + \frac{C}{R} + D \quad (2)$$

where b—maximum number of tokens; R—packet service rate; C—additional delay depending on the transmission rate (in bytes); D—additional delay that does not depend on the transmission rate and is the result of the transmission waiting time of the communication node (in microseconds). The token bucket algorithm is based on an analogy of a fixed capacity bucket, where tokens are added at a fixed rate until the bucket is full. A description of how the token bucket algorithm works is presented in [34]. The CLS class does not provide strict quantitative guarantees of QoS parameters and is intended for real-time applications that can tolerate a certain level of delay and packet loss. The unclassified BE service is intended for classical data transmission without any QoS guarantees. A signaling protocol is still required to ensure and control the QoS connection quality of a

given data stream or streams in the model. The IntServ architecture requires the reservation of certain hardware resources, so a signaling protocol is necessary for the establishment of the required connection. The RSVP (Reservation Protocol) signaling protocol [35,36] is introduced for this purpose. It was developed specifically for QoS signaling in the IntServ architecture. The RSVP protocol creates a flow-specific reservation state at communication device nodes and hosts and; it also serves as a component of the QoS extensions for internet architecture, known as integrated services. As bandwidth is reserved by the entire network in the IntServ model, no other type of traffic can occupy the reserved bandwidth, regardless of whether it is in use or not. Two types of reservations are defined in this context: shared and individual. Additionally, there are two methods for selecting the senders entitled to use the reservation: reservation with strict selection and reservation without selection. Sessions with shared reservations can accept data from multiple senders. In contrast, distinct reservations are designed for exactly one sender. In addition, three booking styles are defined using different booking methods and sender selection methods:

- WF (Wildcard-Filter) style;
- SE (Shared-Explicit) style;
- FF (Fixed-Filter) style;

A summary of the reservation attributes and styles defined for the RSVP protocol is shown (see Table 1).

Table 1. RSVP reservation attributes and styles.

RSVP Reservation Type			
		Shared	Distinct
Sender session	Explicit	SE *(Shared-Explicit) style*; predefined list of senders for whom data are subject to reservation	FF *(Fixed-Filter) style*; the data of only one sender are subject to reservation
Sender session	Wildcard	WF *(Wildcard-Filter) style*; no specific senders. All data sent to the session address are subject to reservation	(None defined)

The IntServ architecture model is schematically shown in Figure 1. In each node (1 to N), mechanisms are implemented to distinguish between data belonging to individual sessions, which can be addressed in point-to-point and point-to-multi-point modes. We can divide the node architecture into two layers. The signaling and management layer is responsible for configuring the node according to the requirements defined by the RSVP signaling protocol. The packet processing layer, on the other hand, is responsible for handling the data stream according to the specifications defined during the configuration. Upon entry to the node, streams are classified into appropriate buffers. For data streams belonging to one signaling session (QoS aware flows), a dedicated buffer is allocated so that the handling of individual streams does not affect the handling of neighboring streams. The number of nodes is denoted by N and the number of QoS aware flows is denoted by I. Appropriate resources must be allocated for each session and handled according to the preset QoS parameters assigned to the session. Resources should be understood as those elements of the node that have been allocated to handle a given data stream transported by the node. These may include elements such as memory in the queue buffer, priority queues, or CPU time. Resources should be allocated in such a way that data streams are handled within the required time frame, avoiding excessive waiting in queues, and are handled at an appropriate speed. This is because excessive waiting times affect the likelihood of queued data being rejected.

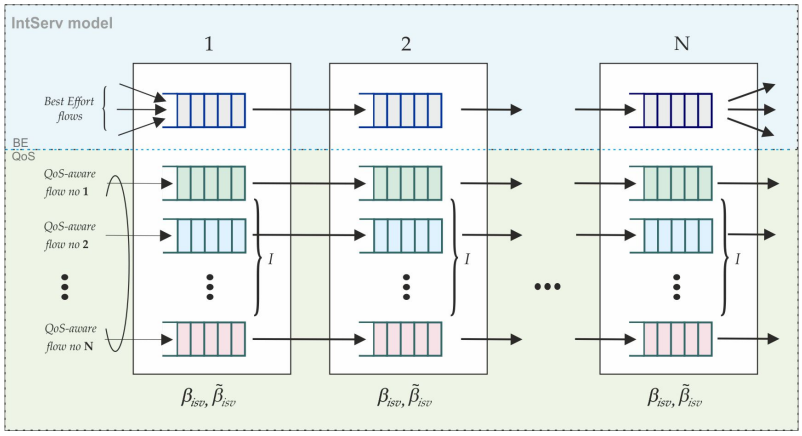

Figure 1. IntServ architecture model.

3.2. Differentiated Services Architecture

The integrated services architecture is an alternative model for supporting QoS data handling developed by the IETF organization. It defines *a priori* an appropriate set of network services based solely on the prioritization mechanisms of the data streams. The development of the DiffServ architecture was motivated by the scalability issues of the IntServ model; hence, the main goal of the new model was to achieve a scalable architecture that guarantees the independence of the resources required for data transmission with the required QoS support. The DiffServ model is also referred to as a *soft QoS* model, but unlike the *hard QoS* model, it does not require a dedicated signaling protocol. The parameters of the data stream transmitted using the soft QoS model, such as bandwidth, delay, and packet loss, are installed and checked on each element of the communication node individually. An example of the DiffServ architecture model is shown in Figure 2. Such a model can be visualized as a series of N communication nodes (1 to N) in which QoS aware flows and BE flows are multiplexed together. Each node has separate buffers for QoS-aware flows, according to the established QoS policy and a separate one for BE flows. It is possible to assign data to a specific traffic class based on a set of characteristics and specific factors specific to the application profile. A given traffic class is provided for the service of selected applications and provides strict guarantees for the transmission of packets. Guaranteeing the appropriate quality of service is particularly important for real-time applications, such as VoIP (Voice over IP) [37], multimedia streaming like IPTV (Internet Protocol Television) [38], and VoD (Video-on-Demand) [39].

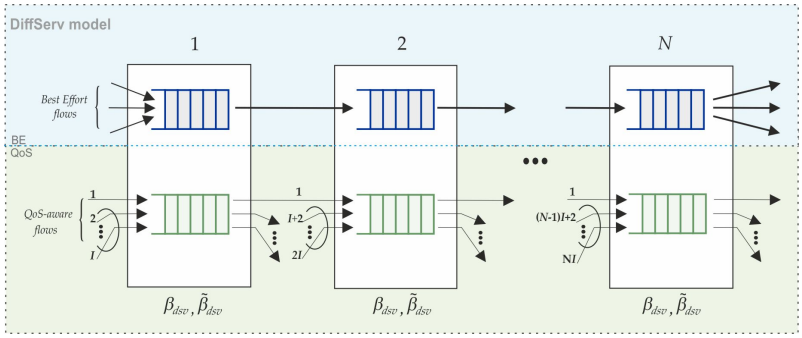

Figure 2. Differentiated Services architecture model.

There can be multiple DiffServ domains in a network, which are a well-defined set of nodes (see Figure 3). Although DiffServ domains are independent, they can still cooperate. In the DiffServ architecture, two types of communication nodes can be distinguished. The ER (Edge Router) node, which is responsible for functioning at the edge of the network, and the CR (Core Router) node, which is responsible for operating the core of the network, with the functionality of the core nodes being simplified as much as possible. In addition,

the nodes have some sort of data serialization mechanism to allow handling with a set QoS. Dedicated handling of individual streams is only possible in ER nodes located at the input or output of the domain. The classification and handling of data is a time-consuming and demanding process, so the processing of incoming data to the node also requires considerable computing power. For this reason, data reprocessing is not performed on all the nodes through which the data pass, but only on the ER edge nodes located at the entrance to the domain. Instead, the handling of aggregate traffic, i.e., the DiffServ aggregate (DiffServ behavior aggregate), takes place in the CR nodes. With each subsequent hop through the CR nodes, the transmitted data are treated according to the assigned marking for the DiffServ aggregate. This is due to the fact that, within the network, all streams can be assigned to a limited number of DiffServ aggregates. The term DiffServ aggregate refers to a stream of data transmitted on a unidirectional link, designated with the same DSCP code. The handling of aggregate traffic, i.e., traffic classes defined by the same DSCP code value, allows these classes to be treated according to accepted transmission models, referred to as PHBs (Per-Hop Behavior). PHBs, therefore, define how traffic within a class is to be transmitted between CR nodes. A CR node can identify the type of PHB group to which data will be forwarded based on the settings of the DSCP field found in the IP header. Currently, the IETF defines the following main PHB groups within the DiffServ architecture:

- EF PHB (Expedited Forwarding Per-Hop Behavior) [40,41];
- AF PHB (Assured Forwarding Per-Hop Behavior) [42];
- CS PHB (Class Selector Per-Hop Behavior) [43];

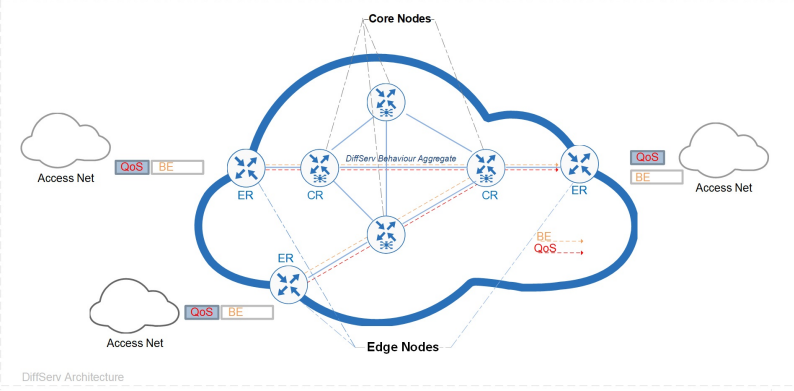

Figure 3. DiffServ domain model.

The EF PHB packet transmission rule defines the packet transmission mechanisms for the transmission of data requiring real-time service, ensuring the highest quality parameters for packet transmission. Typical applications for the EF class include VoIP, interactive games, and online trading programs. The second AF PHB rule is defined for data forwarding within multiple classes of flexible traffic. The AF class has a lower priority than EF and consists of several subclasses with different grades of service priority. The handling of data belonging to the same PHB group is made possible by the setting nodes recognizing the value of the DSCP field found in the IP header. Roughly speaking, EF PHBs are intended for real-time services, while AF PHBs are intended for services better than the best-effort class. The CS PHB rule, on the other hand, indicates the same class as the IP precedence value. It provides DSCP code points that can be used for backward compatibility with IP precedence models. The corresponding definitions for packet header field values for both IPv4 and IPv6 are provided in RFC 2474 [43]. In nodes that support more than one forwarding rule (i.e., which support other traffic classes than just BE traffic), the implementation of a suitably intelligent classification and serialization algorithm is required to handle each class. Furthermore, in order to minimize maximum latency, the concept of intelligent buffers and queueing systems and serialization algorithms for requests are introduced in DiffServ

architectures [44]. The problem of a buffer policy and its parameters in principle can be solved by checking router documentation [45].

4. Marking and Classification

The operation of QoS requires labeling and classifying the imposition of appropriate policies related to data transmission. Different labeling and classification schemes have been designed and standardized. For IPv4, this is the 8-bit ToS (Type of Service) field, while for IPv6, the 8-bit SC (Service Class) field is used [46]. In practice, however, a 6-bit DSCP (Differentiated Services Code Point) bit field located in the IP packet header is used to properly identify aggregates (see Figure 4).

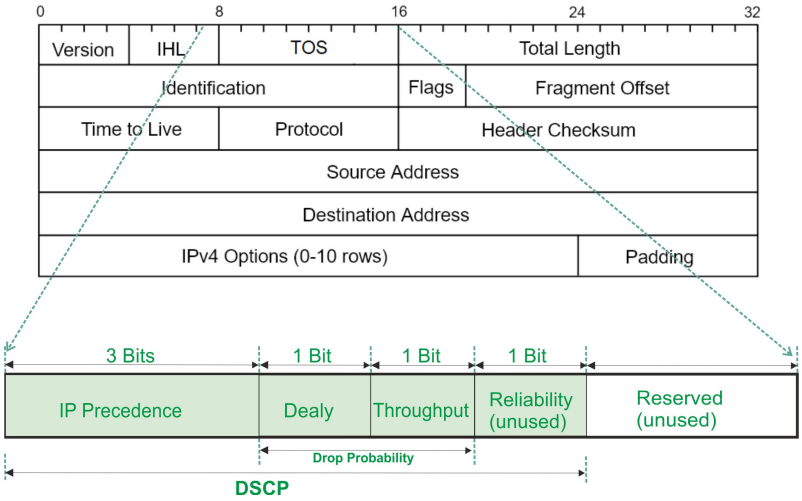

Figure 4. DSCP field in the IP header [43].

The number found in the DSCP field is sometimes referred to as the DSCP code point. Each code point is a combination of bits with an agreed *a priori* meaning. Typically, the assignment of the corresponding DSCP code values is done at the entrance to the network infrastructure. Recommendation RFC 2475 [30] defines the division of the DSCP field into two parts: the class selector field, which ensures compatibility with previous solutions (Type of Service) by analogy with the 3-bit IP precedence field in the ToS field, and the drop probability field, which defines the level of probability of packet loss. The first three bits, therefore, define the IP precedence. The higher the value of this field, the more important the IP packet is. This means that in the event of congestion, the communication node will discard packets with a lower value of this field first. A binary value of 111 set in this field indicates the highest and a value of 000 indicates ordinary or no priority. The 2-bit ECN (Explicit Congestion Notification) field indicates congestion in the network and is a so-called congestion indicator. Setting both ECN bits to 1 informs the communication nodes that the buffer is overloaded. The intention behind the introduction of this mechanism was to inform a suitably intelligent protocol, e.g., TCP, of the occurrence of congestion at the network layer and to signal the sender to slow down the rate of sending new data. The class selector field allows network traffic to be identified by appropriately dividing the traffic into network service classes and prioritizing the packets belonging to this traffic. Packets with the same DSCP class value should be subject to similar handling rules at the communication nodes. Associated with each DSCP value is a specific forwarding strategy of the communication node, known as a PHB. In defining the PHB for each DSCP code point, an attempt was made to maintain compatibility with the original, i.e., to preserve the bit mapping of the DSCP pattern to the former ToS field structure (respectively, the *P*—precedence, *D*—delay, *T*—throughput, *R*—reliability fields). The *D*, *T*, and *R* bits express the need for special treatment of the packet, in terms of low delay (*D*), high throughput (*T*), and high reliability (*R*). An unclassified service is assigned a DSCP value of 000 000. For example, the mapping of a DSCP code with a decimal value of 22 to a

P-D-T-R structure is as follows. The value of 22 in binary notation is 010110; we can map it to a precedence equal to 010 and the *D* and *T* bit set, so the PHB resulting from setting the DSCP to such a value should be characterized by an immediate priority (Table 2), with the packet having the lowest possible delay and the highest possible throughput. The relevant IETF documents define the assignment of specific code point values to different classes of service. Table 2 shows the precedence values in the binary code.

Table 2. Precedence values.

Value	Name of the Priority
000	Routine
001	Priority
010	Immediate
011	Flash
100	Flash override
101	Critical
110	Internetwork control
111	Network control

In the mapping of the DSCP to the *P-D-T-R* structure, the three most significant bits of the DSCP identify the so-called PHB class, while the other two bits reflect the packet's resistance to rejection: the higher the value, the less likely the packet is to be rejected. Thus, the PHB class subfield is the equivalent of the precedence subfield in the ToS/service class field. Although it is possible to use any combination of bits to create distinct traffic classes, it is nevertheless recommended to follow the standards defined by the IETF RFC, which define two basic traffic classes. RFC 2597 [42] defines the AF class and RFC 3246 [41] defines the EF class. These two RFC specifications outline the fundamental traffic marking schemes, including the precise binary values that should populate the type of service byte in the IP header. Each AF class is independently forwarded with its guaranteed bandwidth. The EF PHB packet forwarding rule is designed to handle real-time application data streams. The EF PHB rule, therefore, provides handling mechanisms with the highest quality parameters. Both rules define the packet handling mechanisms implemented in the communication node of the DiffServ domain. Packets belonging to the same class are placed in the same queue. In order to distinguish between different classes of service and enforce priorities between them, a certain queueing and scheduling scheme is used in the communication nodes. A diagram of an example router architecture supporting DiffServ aggregation is presented in [47]. Packets arriving at a communication node are subjected to classification and, based on the value of the DSCP code, are assigned to the appropriate DiffServ behavior aggregate. This means that packets with the same DSCP code value will be handled with the same quality level and are placed in the same queue. Each aggregate represented by a unique DSCP code value is handled by one queue (see Figure 5). On the outgoing link, packets are classified and forwarded to the appropriate buffers implementing the corresponding PHB. This is followed by a serialized process of forwarding packets to the next communication node. A communication node can have multiple input links and multiple output links. For each packet that arrives on an input link, the router determines the next hop in its path and forwards and transmits the packet over the corresponding output link. The service discipline on each output port of the router allocates the resources of the corresponding output link between the different classes of services in link sharing [48]. In the DiffServ domain, packets of flows are classified into a small fixed number of classes, such as CS, EF, or AF. Complex classification for individual flows is only implemented in the edge nodes. In the core, the nodes provide service guarantees only on a class basis and not on a flow basis. The number and variety of defined DSCP values are very large, as 64 different classes can be defined using the 6-bit DSCP field. However, it seems that such detailed data classification does not affect the efficiency of QoS because, in practice, only a few classes are used in most QoS prioritization schemes. Usually, there are between four

and eight traffic classes. Incoming packets are first classified according to the value of the DSCP field. DSCP values other than CS, AF, and EF can be treated as BE traffic.

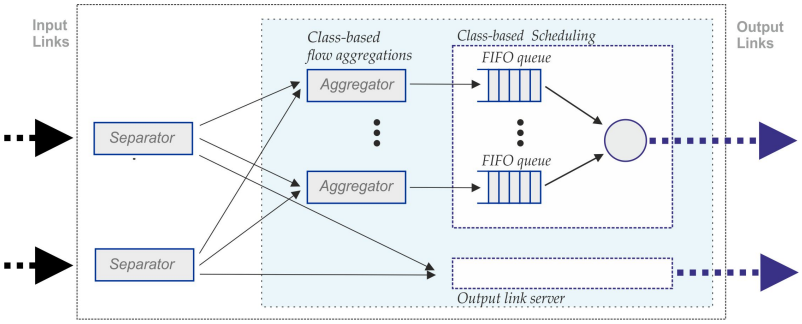

Figure 5. Router architecture.

5. Queueing Systems in QoS Assurance

Queueing theory research has been carried out for many years and is also currently the subject of intensive study. The literature in this area is extensive. In T. Saaty's work [49], he provides a rather extensive list of references and discusses numerous queueing theory applications. Studies devoted to the mathematical aspects of queueing theory were conducted by many other authors [50–53]. Queueing theory is also one of the intensively used tools in the modeling and research of QoS transmission quality in networks [54,55]. The theory makes use of the mathematical apparatus associated with the theory of stochastic processes, in particular Markov processes [56]. Queueing systems theory deals with the construction and analysis of analytical models that can be used in rational management and optimization of critical parameters of queueing systems. A queueing system is understood to be a system in which, on the one hand, there is an influx of requests that require services and, on the other hand, there are service apparatuses that serve the needs of these requests. If the inflow of requests exceeds the capacity to handle them immediately, a queue is formed. A block diagram of a simple queueing system is shown in Figure 6.

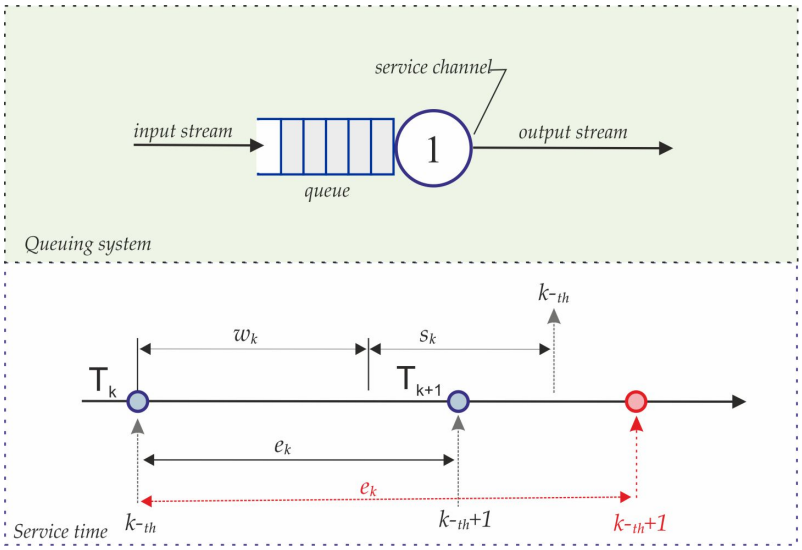

Figure 6. Queueing system and service times.

A service channel, an input and output stream, and a system queue are marked in the system. The service channel (marked 1) can be any device or team handling incoming requests to the system. If the service channel is busy, the request goes to the queue and waits in the queue. The input stream is, therefore, made up of requests arriving at the

system and the output stream of requests leaving the system. A characteristic feature of such a model is the sequence of waiting times for requests described by Equation (3):

$$w = (w_1, w_2, w_3, \ldots) \tag{3}$$

waiting to enter the system. We can derive a recursive equation of waiting times for the system. Consider the k_{th} request marked with an arrow in Figure 6 arriving at the system at time T_k. The request k_{th} waits in the queue for a time period w_k, and is then serviced in the system for the time period, marked in the diagram as s_k; only after time Equation (4)

$$T_k + w_k + s_k \tag{4}$$

does the request leave the system. The location of the k_{th} request leaving the system is marked with an arrow in the diagram. The next request arriving in the system is request k_{th+1}, which arrives in the system at time Equation (5):

$$T_{k+1} = T_k + e_k \tag{5}$$

The arrival time of a request k_{th+1} is indicated by an arrow in the diagram. If the subsequent request k_{th+1} arrives in the system during the handling time of the earlier request k_{th} , and if

$$e_k \leq w_k + s_k \tag{6}$$

then the waiting time for the request k_{th+1}, as shown in Figure 6, will be

$$w_{k+1} = w_k + s_k - e_k \tag{7}$$

However, the next request k_{th+1} may arrive in the system after the handling of the earlier request has been completed; hence, if

$$e_k > w_k + s_k \tag{8}$$

then the request arrives in an empty system. This case is indicated by the red arrow in Figure 6. The request arrives in an empty system, so the waiting time for such a request k_{th+1} is $w_{k+1} = 0$. If we assume that the waiting time w_0 is the 'warming-up' time of the server, this is the time during which the server cannot serve the requested request starting at $T_0 = 0$; hence, the service time takes the value service time $s_0 = 0$. The above arguments, therefore, lead to the recursive relation Equation (9):

$$w_{k+1} = (w_k + s_k - e_k)_+, k \geq 0; \tag{9}$$

where

$$(.)_+ = max(0, .)$$

From this relation, it follows by induction that the waiting time w_k for all $k \geq 1$ is described by Equation (10):

$$\begin{aligned} w_k = max\{0, (s_{k-1} - e_{k-1}), (s_{k-1} - e_{k-1}) + (s_{k-2} - e_{k-2}), \ldots, (s_{k-1} - e_{k-1}) \\ + \ldots + (s_1 - e_1) + \ldots + (s_1 - e_1) + (s_0 - e_0) + w_0\} \end{aligned} \tag{10}$$

Similar to non-negative waiting times w_k, we can also consider the magnitude of v_k, which takes both non-negative and negative values. If $v_k > 0$, then v_k is the waiting time for a request k_{th} arriving at the system. In other words, the value of $v_k = w_k$. On the other hand, if $v_k < 0$, then $|v_k|$ is the idle time of the server waiting for a k_{th} request to arrive in the system. The waiting time $v_{(k+i)}$ can, therefore, be described by Equation (11):

$$v_{k+i} = (v_k)_+ + s_k - e_k, k \geq 0; \tag{11}$$

Since $(v_k)_+ = w_k$, using Equation (10), we obtain the relation Equation (12):

$$v_k = max\{(s_{k-1} - e_{k-1}), (s_{k-1} - e_{k-1}) + (s_{k-2} - e_{k-2}), \ldots, (s_{k-1} - e_{k-1}) + \ldots + (s_1 - e_1), (s_{k-1} - e_{k-1}) + \ldots + (s_1 - e_1) + (s_0 - e_0) + w_0\}, k \geq 1; \quad (12)$$

In order to characterize the queueing system, it is necessary to define the statistical distributions that describe the distribution of times between consecutive requests arriving at the system and the distribution of service times that describe the distribution of times required to service consecutive requests. The model shown in Figure 6 consists of a single server to which a stream of requests arrives, handled in order of arrival. Such a model is traditionally referred to as a FIFO (First-In-First-Out) queueing discipline. It is also the primary mechanism supporting data transfer in computer networks. FIFO queueing is easy to implement and it treats all data equally. However, FIFO queueing does not support priority data so all incoming data are treated equally. The disadvantage of FIFO queueing is that it is of limited use in providing QoS. If data come into the system from different traffic streams, then one stream can easily disrupt the flow of the other streams. This means that FIFO processing can lead to a situation where an aggressive stream can monopolize more capacity or the entire queue buffer. The result can be a rapid deterioration of QoS, causing, for example, a sudden increase in the delay of transmitted data. However, several scheduling algorithms have been developed, demonstrating better memory isolation between flows [57]. Classification of queueing systems can be performed based on various criteria. The most commonly used classification criteria are:

- Number of operating channels:
  - single-channel;
  - multi-channel.
- Occurrence of queues:
  - systems with losses;
  - systems with waiting.
- Queue length:
  - limited;
  - unlimited.
- Service discipline:
  - system with priority;
  - system without priority.

A queueing system can be characterized by its queueing rules, which determine the order in which requests in the system are handled [58]. Several different queueing algorithms, depending on the needs and characteristics of the traffic, are used to manage congestion at communication nodes. Apart from the aforementioned FIFO queueing, the most common queueing disciplines are:

- PQ (Priority Queueing);
- RR (Round Robin);
- WRR (Weight Round Robin);
- DRR (Deficit Round Robin);
- CBQ (Class-Based Queueing);
- SFQ (Stochastic Fairness Queueing).

Queueing algorithms are the primary mechanisms used to shape traffic and ensure QoS transmission quality. They prioritize requests and send them, maintaining a certain queueing discipline. In priority queues, separate queues are used for the data of different traffic classes, i.e., different priorities [59]. Requests intended for transmission over a common communication channel are always selected, starting from the non-empty queues with the highest priority. Consequently, submissions from lower-priority queues are only selected if all higher priority queues are empty. Priority queueing systems form a large

class of queueing systems, where the incoming requests are to be distinguished by their importance [60]. A schematic of priority queueing systems is shown in Figure 7. This system is a fairly simple data serialization tool that consists of two or more FIFO queues $(Q_1, Q_2, \ldots, Q_N)$ with different priorities. Incoming requests are initially sorted and placed in the queues according to their priority. Individual queues are handled, starting with the queue with the highest priority (Q_1). The lower-priority queues $(Q_2, \ldots, Q_N)$ are only served when the higher priority queues are empty. The main problem with the PQ algorithm is that the continuous flow of requests to the highest priority queue can lead to a complete blockage of the lower-priority queues and starve the data streams. In the worst case, requests from such queues may never be served.

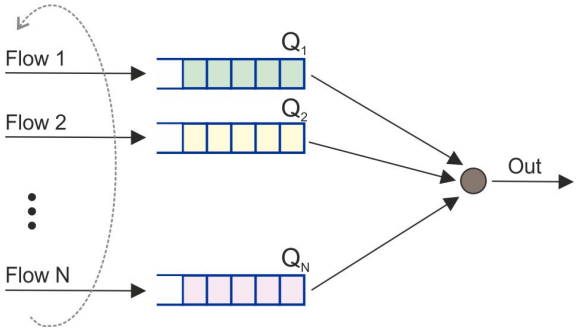

Figure 7. Priority queueing systems.

There are various modifications of the PQ algorithm that eliminate the problem of stream starvation. These belong to the group of so-called FQ (Fair Queueing) algorithms. WRR (Weighted Round Robin) limits the number of consecutive requests belonging to the same class that can be transmitted over a channel. When the limit of requests belonging to the same class is reached, the scheduler switches to the next non-empty priority queue and follows the same rule. These limits are called weights and are denoted as w_i. With k classes of traffic handled by the algorithm, if there are enough packets in all classes, the algorithm will select w_1 requests from class 1, followed by w_2 requests from class 2, ..., up to w_k requests from class k. At the end of the cycle, it selects w_1 requests from class 1, and so on. As a result of such an operation, the communication channel is shared by notifications belonging to all classes according to Equation (13):

$$u_i = \frac{w_i / s_i}{\sum_{j=1,\ldots,k} w_j / s_j}, i = 1, 2, \ldots, k; \tag{13}$$

where $s_i, i = 1, \ldots, k$ is the transmission rate for the submissions belonging to class i. If the transmission rates are identical for submissions belonging to all classes, then the ratio is described by Equation (14):

$$u_i = \frac{w_i}{\sum_{j=1,\ldots,k} w_j}, i = 1, 2, \ldots, k; \tag{14}$$

Thus, in the example system with 3 priority classes and with weights of 3, 2, and 1, respectively, these "utilisation bounds" are equal to 1/2, 1/3, and 1/6 for classes 1, 2, and 3, respectively. Another variation of the weighted priority algorithm is the DRR algorithm. It allows flows with variable packet lengths to share bandwidth fairly. A detailed analysis of the performance of the DDR algorithm is presented in [61]. Like DDR, the SFQ algorithm also belongs to the group of fair queueing algorithms. Here, flows can be classified based on parameters such as source and destination addresses. The allocation of data to the appropriate queue is based on the value of the hash function from these parameters. Each queue is handled fairly and cyclically in a round-robin fashion. It is observed that SFQ outperforms DRR by maintaining a comparatively low per-packet delay. On the other hand, DRR has the highest packet ratio compared to SFQ [62]. Another fair

queueing algorithm is the CBQ algorithm, which is a more advanced form of the WRR algorithm. Here, data are classified into traffic classes, so that specific groups of data can be given a specific priority—or a weight. The weight can be specified as a percentage of the transmission channel throughput. A typical example of the use of the queueing algorithms discussed here involves communication nodes responsible for the transmission of various types of data in networks. They allow the implementation of various QoS mechanisms. The operation of communication nodes without any QoS mechanisms is based on the assumption that the device processes data in the order in which the data were received, i.e., using the FIFO method. Devices supporting QoS can classify incoming data into predefined traffic classes and then handle the aggregated data streams. The handling process follows an implemented queueing mechanism and a defined handling and traffic-shaping policy to provide services with established QoS [63].

6. Queueing System Parameters

The notation proposed by Daviad G. Kenadall [64] is widely used to describe queueing systems. He proposed a simple symbolism, with the following syntax:

$$A/B/m/N - S \tag{15}$$

where A—distribution of the inter-arrival time; B—distribution of the service; m—number of servers; N—queue capacity; if $N = \infty$, then this letter is omitted; S—optional parameter, indicating the service discipline used (FIFO, LIFO, etc.). It is assumed that if the S parameter is omitted, then the service discipline is always FIFO. Similarly, the N parameter represents the capacity of the queue. If the queue is infinite $N = \infty$, then the N parameter is not entered in the Kenadall notation. Appropriate letter designations are placed in the A and B positions to indicate the type of schedules that are used in the system. The following designations for distributions are commonly used: M—an exponential distribution with

$$A(t) = 1 - e^{-\lambda t} \tag{16}$$

and

$$a(t) = \lambda e^{-\lambda t} \tag{17}$$

where parameter $\lambda > 0$, is a continuous distribution having the Markov property, D—deterministic distribution, and E_k—Erlang distribution of order k, $(k \geq 1)$. For the E_k distribution, we have

$$A(t) = 1 - e^{-k\mu t} \sum_{j=0}^{k-1} \frac{(k\mu t)^j}{j!} \tag{18}$$

H_k—hyperexponential distribution with k phases. Here, we have

$$A(t) = \sum_{j=1}^{k} q_j (1 - e^{-\mu j t}) \tag{19}$$

G—General distribution.

Of the distributions mentioned, the exponential distribution is considered extremely useful. On the one hand, it describes the behavior of the system well and, on the other hand, its simplicity facilitates mathematical notation. The simplest queueing system with FIFO support can, therefore, be written in Kenadall's notation as $M/M/1$. Such a model designation indicates a queueing system with a single service channel, an infinite queue, and an exponential distribution of the service time and influx of subsequent requests to the system. For any queueing system, it is very important to determine its most important characteristics and parameters, which include the average number of requests residing in the system or queue, the average time a request stays in the system or queue, the average intensity of the stream of requests in the system. The parameters describing the characteristics of the queueing system can be determined based on the argument presented

in [65]. The average residence time of a request in the system is the sum of the average residence time of the request in the queue and the average handling time. Similarly, the average number of requests in the system is the average number of requests in the queue and the average number of requests handled. Let us assume the following designations for the parameters in question:

- L—average number of requests in the system;
- W—average length of time a request stays in the system;
- Q—average number of submissions in the queue;
- T—average waiting time of a request in a queue.

Let us consider the inflow and handling of requests in the queueing system shown in Figure 8. The process is denoted as follows: $A(t)$—the number of requests flowing into the system at time $(0, t)$; $D(t)$—number of submissions leaving the system at time $(0, t)$; t_i—the residence time of the submissions in the system i; τ—observation period.

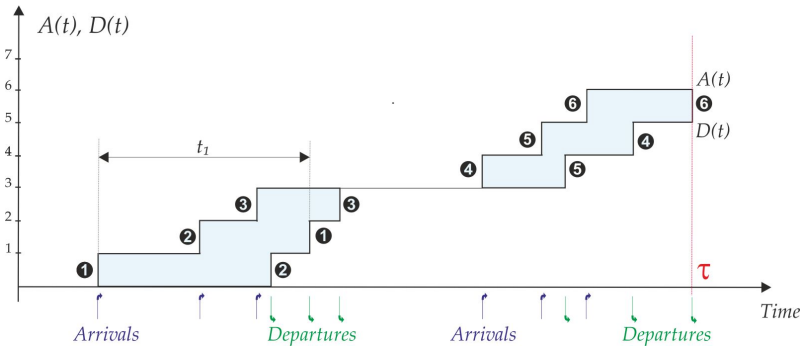

Figure 8. Inflow and handling process in the queueing system.

If the incoming system is in a steady state, then the average number of incoming requests and the average number of requests leaving the system are equal to each other. Both streams have an intensity equal to λ. The number of requests entering and leaving the system is described by the discrete functions $A(t)$ and $D(t)$. At a given time t, their difference is $Z(t) = A(t) - D(t)$ and this determines the number of requests present in the system. If $A(t) = D(t)$, this means that there are no submissions in the system. In Figure 8, this is the area where $A(t)$ and $D(t)$ form a single line. The average number of requests located in the system L during the time period analyzed τ can be determined by the integral of the function $Z(t)$ divided by the length of the time segment τ; hence, we have Equation (20):

$$L = \frac{1}{\tau} \int_0^\tau Z(d)\, dt \tag{20}$$

The calculated integral is the area of the figure shown in Figure 8. After sufficient time τ, we can write the value of L as Equation (21):

$$L = \frac{1}{\tau} \int_0^\tau Z(d)\, dt = \frac{1}{\tau} \sum_i t_i \tag{21}$$

Multiplying and dividing the right-hand side of Equation (21) by the intensity yields Equation (22):

$$L = \frac{1}{\tau\lambda} (\sum_i t_i)\lambda \tag{22}$$

The value $\tau\lambda$ is, therefore, the average number of requests that arrived in the system during τ. We can write the average time a submission stays in the system as Equation (23):

$$W = \frac{1}{\tau\lambda} (\sum_i t_i) \tag{23}$$

By substituting the value of Equation (23) into Equation (22), we finally obtain Little's law Equation (24):

$$L = \lambda W \tag{24}$$

Equation (24) implies a very useful law in queueing theory called Little's law, which asserts that the time-averaged number of requests in a queueing system, denoted as L, is equal to the rate at which requests arrive and enter the system, represented by $\lambda \times$ the average sojourn time of a request, W. Little's law applies to any queueing system, regardless of the type of request stream, service time distribution, or queue discipline.

7. Petri Nets and Timed Petri Nets

Petri nets, sometimes called net structures, are abstract formal models that have been developed in the search for natural, simple, and efficient methods, in order to describe and analyze the flow of information in systems. Carl A. Petri, in his work [66], proposed a formal model of systems composed of independently operating concurrent elements that can communicate with each other to exchange information or synchronize their processes. This model is commonly known as PNs (Petri nets) [67–69]. PNs represent a significant advancement in modeling parallel and concurrent processes. In PNs, the system's state is derived from a combination of local state variables, enabling a direct representation of concurrency, causality, and independence. A central feature of the model is that any form of synchronization in the system must be achieved through the exchange of information between the system components. This principle is reflected in the formal notation of PNs. A Petri net is an abstract concept, formally given in the form of an ordered triple [68,70]:

$$N = (P, T, A) \tag{25}$$

for which $P = \{p_1, p_2, \ldots p_n\}$ is a non-empty finite set of places representing conditions, $T = \{t_1, t_2, \ldots t_n\}$ is a non-empty finite set of transitions representing events, such that: $(P \cap T) = \oslash$ and $(P \cup T) \neq \oslash$; A is a binary relation that binds the elements of the sets P and T; otherwise, A is a set of networks, such that

$$A \subseteq (P \times T) \cup (T \times P) \tag{26}$$

i.e., A does not directly connect places or transitions and is known as the flow relation, also referred to as the causality relation. It is an incidence of network vertices or alternatively, directed arcs. Relation A between places and transitions can be represented by two separate directional relations:

$$a \subseteq (P \times T) \tag{27}$$

$$b \subseteq (T \times P) \tag{28}$$

P-elements and T-elements contain input and output sets. P-elements connected by arcs directed toward t-elements are called input elements, while p-elements connected by arcs directed away from t-elements are called output elements. The sets of input and output transitions of a p-element space are defined as follows:

$$Inp(p) = \{t \in T : (t, p) \in A\} \tag{29}$$

$$Out(p) = \{t \in T : (p, t) \in A\} \tag{30}$$

Similarly, sets of input and output sites are defined for a particular transit t

$$Inp(t) = \{p \in P : (p, t) \in A\} \tag{31}$$

$$Out(t) = \{p \in P : (t, p) \in A\} \tag{32}$$

A characteristic feature of networks is the graphical representation of information flow. The following interpretation is adopted for networks. The network is represented as

a bipartite graph $G = (V, A)$, whose nodes are elements of the set P and T, i.e., $V = P \cup T$, while the arcs are pairs of relations A. Figure 9 shows an example of a net structure modeling the producer–consumer system, for which

$$P = [p_1, p_2, p_3, p_4, p_5] \tag{33}$$

$$T = [t_1, t_2, t_3, t_4] \tag{34}$$

$$A = [(p_1, t_2), (t_2, p_2), (p_2, t_1), (t_1, p_1), (t_2, p_5), (p_5, t_3), (t_3, p_4), (p_4, t_4), (t_4, p_3), (p_3, t_3)] \tag{35}$$

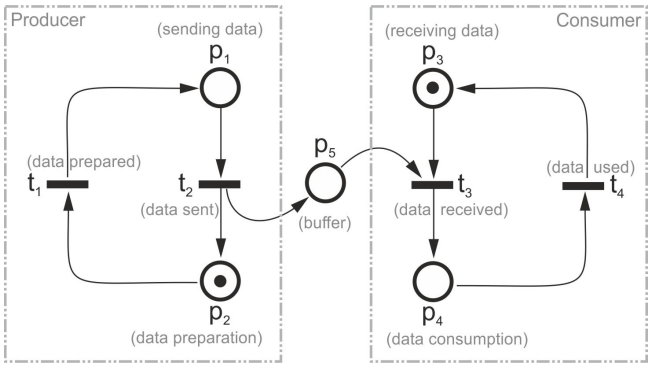

Figure 9. Graphical representation of a Petri net.

The graphical representation of the network shown in Figure 9 is a simple model of two processes communicating over an unlimited buffer, i.e., a producer–consumer system with an unbounded buffer. To fully illustrate the communication process, the model shown is supplemented with *tokens* inside the p-elements. The behavior of events in the network is represented precisely by the *tokens* assigned to the p-elements of the network. In PNs, the distribution of *tokens* over places changes by occurrences or transition firings. Enabling and firing rules are associated with transitions. Informally, we can say that the enabling rule defines the conditions that allow a transition to fire, and the firing rule specifies the change of state produced by the transition. Both rules are specified in terms of arc characteristics. An active transition removes one token from each input location and adds one *token* to each output location. Typically, a p-element with at least one *token* associated with it means that the condition represented by that element is satisfied. The distribution of *tokens* in p-elements can be described by the marking function $m : P \rightarrow \{0, 1, 2 \ldots\}$ or can be represented as a vector describing the number of *tokens* assigned to consecutive p-elements of the network: $m = [m(p_1), m(p_2), \ldots]$. The network N, including the initial marking function m_0, is called a marked net M and can be equivalently defined as in Equation (36):

$$M = (N, m_0) = (P, T, A, m_0) \tag{36}$$

An example of a net structure extended with a m_0 marking function, which determines the initial state or marking of the net, is shown in Figure 9. In many practical applications of PNs, the activation of transitions is synchronized by external events, and certain actions are assigned to the locations of the net that are performed when the described system is in a certain state. The modeled example shows the process of asynchronous communication between the producer and consumer. The network, including the initial marking function with *tokens* placed inside p-elements, i.e., p_2 and p_3, represents a marked net, M. The two cyclic subnets (t_1, p_1, t_2, p_2) and (p_3, t_3, p_4, t_4) represent the producer and the consumer, respectively, while place p_5 is the buffer. As mentioned earlier, the marking function, m, is an assignment of *tokens* to p-places and can be represented by a vector with as many components as there are places in the network; the i_{th} component then represents the number of *tokens* assigned to a p-place in the network. A change in the state of the sender occurs as a result of events assigned to transitions. Transition t_1 is assigned the *data prepared*

event, and transition t_2 is assigned the *data sent* event. Similarly, the consumer may be in one of two states, p_3 or p_4, and its change occurs as a result of the occurrence of events, assigned to transitions t_3 and t_4, respectively. In a marked net, M, a transition (t) is enabled by a marking m *iff*:

$$\forall_p \in Inp(t) : m(p) > 0 \tag{37}$$

An enabled transition (t) can fire, transforming a marking (m) into a directly reachable marking, m':

$$\forall_p \in P : m'(p) = \begin{cases} m(p) - 1, & if p \in Inp(t) - Out(t), \\ m(p) + 1, & if p \in Out(t) - Inp(t), \\ m(p), & otherwise. \end{cases} \tag{38}$$

A pair comprising a place (p) and a transition (t) is called a self-loop if p is both the input and output of t $p \in Inp(t) \wedge p \in Out(t)$. A net is said to be pure if it has no self-loops. Pure nets are completely characterized by a single matrix, C. The connectivity matrix, C, or—in other words—the network invariance matrix, is an $n_p \times n_t$ matrix, where n_p is the number of p-elements and n_t is the number of t-elements. For the network shown in Figure 9, the incidence matrix takes the form:

$$C = \begin{bmatrix} +1 & -1 & 0 & 0 \\ -1 & +1 & 0 & 0 \\ 0 & 0 & -1 & +1 \\ 0 & 0 & +1 & -1 \\ 0 & 1 & -1 & 0 \end{bmatrix}$$

The analysis of the model shown in Figure 9 shows that the depicted communication process is a unidirectional process and is determined by a sequence of arcs and nodes, corresponding to the unidirectional flow of *tokens* in the model. Many practical process synchronization schemes are derived from this simple model. The model can be considered in different variants, where data *tokens* flow into a queue buffer of varying capacities. Bernardini et al. [71] researched models that synchronized through a buffer with a capacity of one element, as well as a parallel version with direct access to two separate buffers, each having equal capacity. The Petri nets theory has been extensively studied for many years, and this research has resulted in the definition of many types of networks and a substantial body of literature describing their properties. There are many modifications to Petri nets created to solve specific problems of a complex nature. Although there are many different variations of Petri nets [72–75], their common feature is a structure based on a bipartite directed graph, i.e., a graph with two types of vertices, alternately connected by directed edges i.e., arcs. These two types of vertices represent, in a general sense, the conditions and events occurring in the modeled system, with each event being able to occur only if all the conditions associated with it are satisfied. A system interpretation that introduces events that depend on their associated conditions (condition–event systems) is very convenient and helpful for modeling complex discrete systems [76]. The Petri net class choice is, thus, determined by the purpose of the analysis. The behavior of PNs is time-independent and characterized by the non-deterministic firing of transitions that are simultaneously included in a given label. In order to investigate the performance aspects of systems modeled with PNs, however, the duration of the modeled activities must be taken into account. This can be done in different ways, resulting in different types of temporal networks. Network models with attached temporal descriptions are called TPNs (Temporal Petri Nets). We are dealing with TPNs when the realization of transitions is not immediate but takes a finite amount of time. This implies that in defining such a net, the characteristics described in the transitions are considered. A TPN is defined as a pair, $T = (M, f)$, where M is the marked net, $M = (N, m_0)$; f is the firing time function that assigns the average firing time or occurrence time to the transition of the net, and $f : T \rightarrow R^+$, where R^+ is the set of nonnegative real numbers. To analyze the performance of temporal networks, an additional component is needed to describe random decisions in nondeterministic

networks. This is c (conflict resolution function), $c : T \rightarrow [0, 1]$, which assigns probabilities of firings to transitions in free-choice classes of transitions and relative frequencies of firings to transitions in conflict classes [75]. Temporal characteristics in transitions can determine the time associated with the realization of transitions in various ways. In particular, it can be a deterministic value or described by a random variable with a given probability distribution. In the first case, the corresponding timed nets are referred to as D-timed nets [77], in the second, for the exponential distribution of occurrence times, the nets are called M-timed nets, Markovian nets [78]. In simulation applications, other distributions can also be used, such as the uniform distribution U-timed. In TPNs, different distributions can be associated with different transitions of the same model, providing great flexibility in working with simulation models. A detailed characterization of TPNs, as well as a detailed characterization of the behavior of timed nets with immediate and timed transitions, are presented in the literature [75,79,80].

8. The Model

In order to evaluate the performance of the modeled systems, a Petri Net model, shown schematically in Figure 10, was used. It is advisable to drop data that exceed the priority queue size rather than incur additional queueing latency, which can lead to network congestion, resulting in delays, packet loss, and degradation of quality of service metrics. Therefore, the research presumes that the model of the system under study would consist of three priority queues with finite capacity. These will serve as system queues, i.e., storage areas for priority data after classification. Each queue will store incoming data of the same priority class. Such a mechanism prevents a priority queue from monopolizing the entire interface. This approach allows for effective traffic management for high, medium, and low priority data. In the models studied, the data are designated as class-1, class-2, and class-3, respectively. Transitions t_1, t_2, and t_3, together with places p_1, p_2, and p_3 are independent sources of generated data for class-1, class-2, and class-3, respectively. The rates of generated data $a_1 = 1/f(t_1)$, $a_2 = 1/f(t_2)$, and $a_3 = 1/f(t_3)$ are determined by the occurrence times associated with t_1, t_2, and t_3. The Markovian mode of the timed transition determines the distribution of the inter-arrival times for the source. In the models tested, assumptions were made that the PQ model would consist of three priority queues. The three places, p_4, p_5, and p_6, are queues for class-1, class-2, and class-3 data, respectively. Timed transitions t_4, t_5, and t_6 represent the transmission channels for class-1, class-2, and class-3 data. For simplicity of the description, it is assumed in this paper that the transmission times are approximately the same for data of different classes. In the model, the data transmission time is assumed to be deterministic at one time unit (for all three classes). In addition, in the model, place p_7 is shared by transitions t_4, t_5, and t_6, in order to ensure that only one data type is transmitted at a time. Hence, place p_7 is denoted by a single token (i.e., $m_0(p_7) = 1$). In addition, arc inhibitors (p_4, t_5) do not allow data to be transmitted over transition t_5 when there is class-1 data waiting to be transmitted. Similarly, arc inhibitors (p_4, t_6 and p_5, t_6) only allow transmissions when there is no class-1 or class-2 data waiting to be transmitted. One characteristic of priority queues is that, as long as any of the data are in a higher priority queue, data waiting in lower-priority queues cannot be transmitted. Transmitting data only from higher-priority queues can lead to a transmission starvation problem for data classified in lower-priority queues. The model under study uses a modification of the PQ algorithm that eliminates the problem of starvation of lower-priority data streams. It was assumed that the system would sequentially handle five requests from the class-1 queue, three requests from the class-2 queue, and finally two requests from the class-3 queue. In addition, the model used finite input queues, where requests waiting to be handled in the system flow. Based on the models prepared, traffic shaping mechanisms were investigated in systems based on priority queueing with 5-3-2 priorities and a finite input queue of 5, 25, and 75 requests. M-timed Petri nets (Markovian nets) were used as the sources of data generation in all three queues.

The rates of generated data are determined by the occurrence times associated with transitions t_1, t_2, and t_3: $\rho_1 = 1/f(t_1)$, $\rho_2 = 1/f(t_2)$, $\rho_3 = 1/f(t_3)$, so traffic intensity $\rho = \rho_1 + \rho_2 + \rho_3$. The average waiting time is determined by $(t(p_x)/n(p_x))$ and the average queue length is determined by $(t(p_x)/SimTime)$; finally, utilization and throughput are determined by $(t(t_x)/SimTime)$, where $x = 1, 2, 3$. The $t(\dots)$ arguments refer to the total (cumulative) firing times or total token waiting times. The $n(\dots)$ argument refers to the number of firings or the number of tokens entering the places. *SimTime* refers to the simulation times.

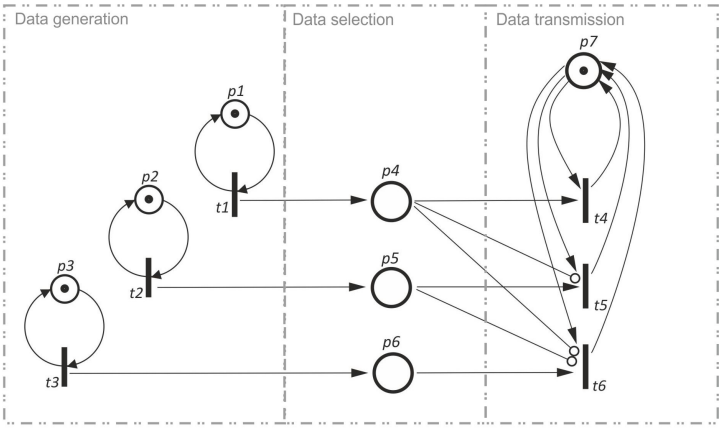

Figure 10. Petri net model.

9. Performance Characteristics

Performance results are obtained by the event-driven simulation of timed Petri Net models. A collection of software tools for the analysis of temporal Petri nets, TPN-tools, developed and under development at the Department of Computer Science at Memorial University, St. John's, Canada, was used to perform simulations and build models of PQS systems. In timed Petri nets, the occurrence times of some transitions may be equal to zero, which means that such occurrences are instantaneous; all of such transitions are referred to as immediate, while the others are referred to as timed [75]. The model developed was evaluated for different combinations of modeling parameters. Performance results were compared for three different data classes with weights equal to 5, 3, and 2 for three different queue lengths of 5, 25, and 75 requests, respectively. With three priority classes and weights 5, 3, and 2, utilization bounds were equal to 0.5, 0.3, and 0.2 for classes 1, 2, and 3, respectively. In order to compare the system performance, an analysis was made of the parameters, such as AWT (Average Waiting Time), AQL (Average Queue Length) and utilization. The research carried out has made it possible to estimate the impact of queue lengths on the performance characteristics of the system and different classes of data. Experimental results reveal that, for the same number of data generated and channel usage, AWT is different. In addition, the study found significant differences in systems with queues of 5, 25, and 75. It was found that the AWT characteristics in the range up to $\rho < 0.30$ are similar in all models and have linear behavior. However, for $\rho > 0.30$, there are dynamic changes in the performance characteristics in the models studied. The largest increase in AWT values was found for class-3 data. Furthermore, the maximum AWT values for class-3 data were found to vary between different queue lengths. In the model with a queue length = 5, the $AWT_{max} = 24$ (Figure 11), in the model with a queue length = 25, the value increases to $AWT_{max} = 124$ (Figure 12), and for the longest queue length = 75, the $AWT_{max} = 372$ (Figure 13). As can be seen, increasing the queue length to a value of 75 resulted in a 15.5 times the increase in the AWT_{max} value for class-3 data. Class-2 data are similar, but here, increasing the queue length to a value of 75 results in a 16.2 times increase in the AWT_{max} value. For the highest priority class-1 data, the increase is the greatest at 17.65 times. As can be seen when the traffic volume approaches $\rho > 0.55$, AWT becomes significant and approaches the maximum value for all three data classes.

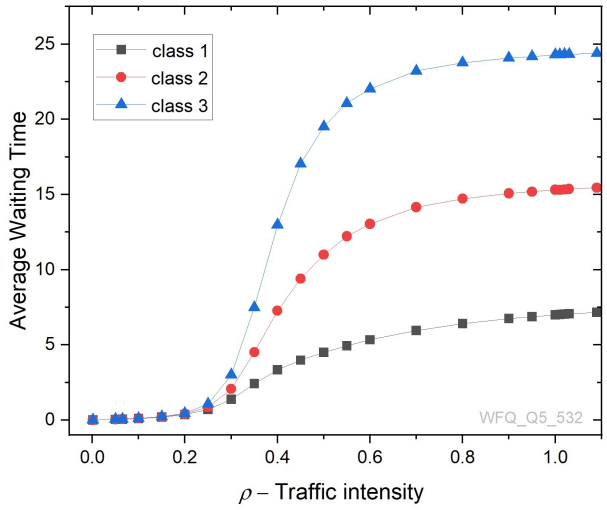

Figure 11. AWT as a function of ρ with a queue length = 5.

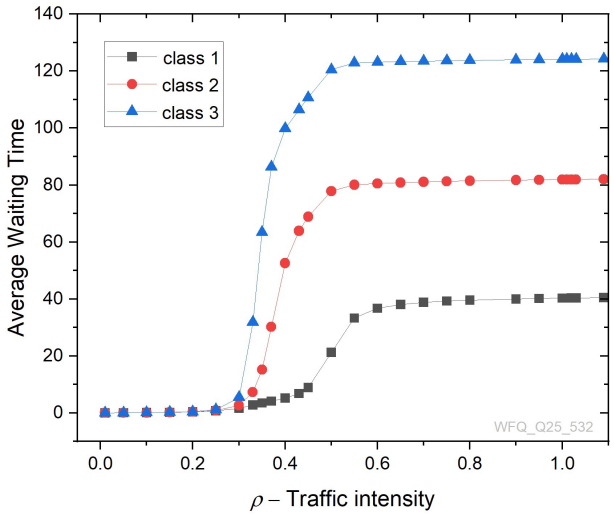

Figure 12. AWT as a function of ρ with a queue length = 25.

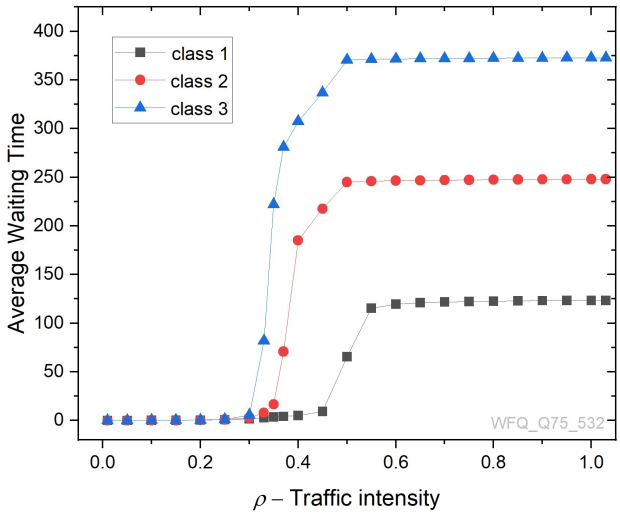

Figure 13. AWT as a function of ρ with a queue length = 75.

The dynamics of the changes occurring in the model become the most apparent for the AWT parameter, which is directly related to the AQL parameter. Both parameters studied show a similar character, as can be seen in (Figures 14–16). In all the studied models, the AQL tests showed that, for $\rho > 0.3$, there is a sharp increase in queue length. As expected, in the first stage, the length of the queue of the lowest priority data, i.e., data of type class-3, increases sharply. In the studied model with a queue length = 75, it was found that for $\rho = 0.3$, the $AQL = 2.08$. On the other hand, increasing the value of ρ by 0.1, i.e., increasing to $\rho = 0.4$, results in a dynamic increase in data queueing, and the value is $AQL = 73.5$. Similar dynamic changes were observed for the data of class-2 and class-1. The study showed that for $\rho > 0.6$, the AQL values in all models are close to the maximum values.

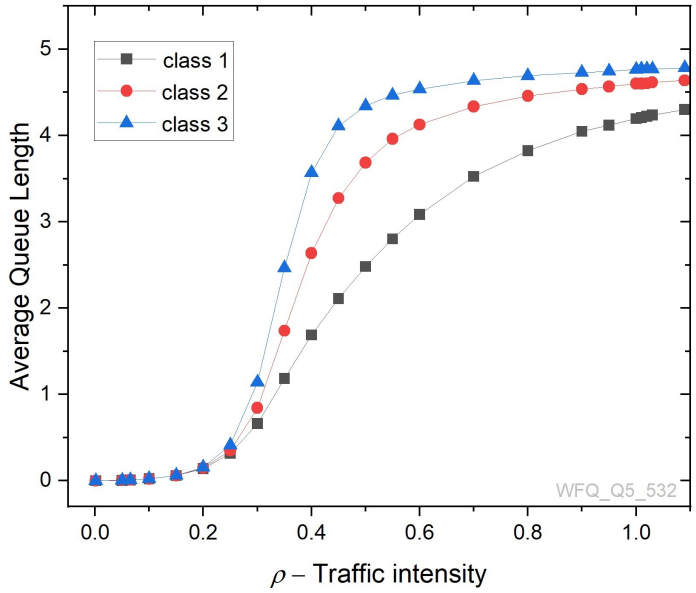

Figure 14. AQL as a function of ρ with a queue length = 5.

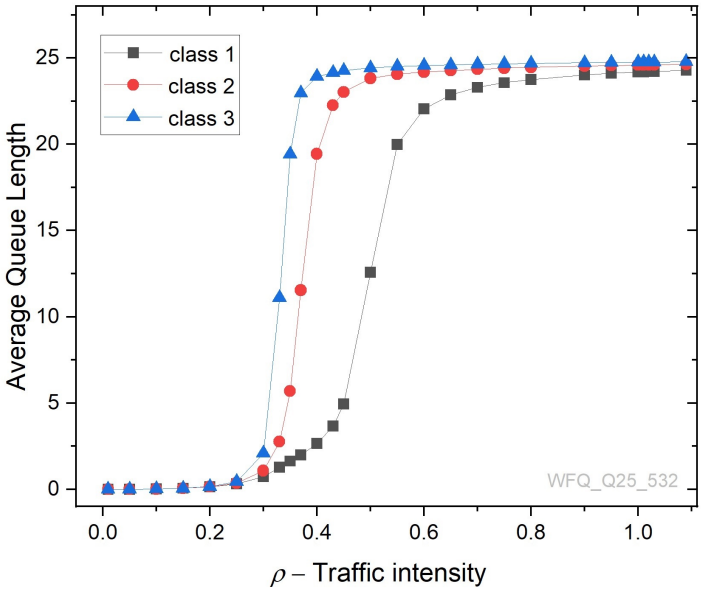

Figure 15. AQL as a function of ρ with a queue length = 25.

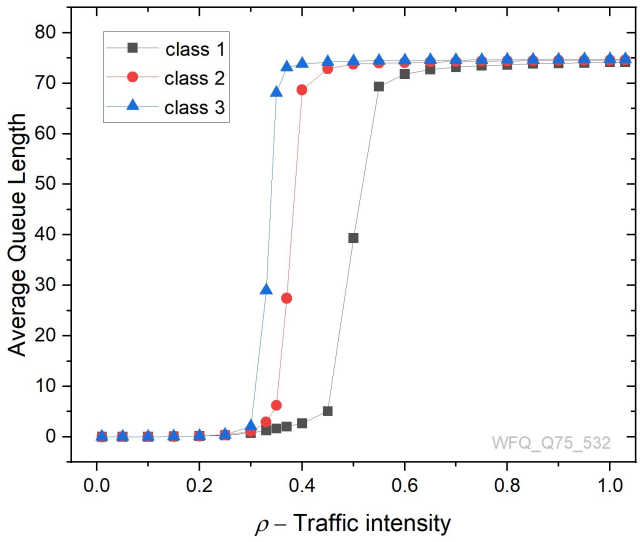

Figure 16. AQL as a function of ρ with a queue length = 75.

In the next stage, the utilization parameter was examined. As before, a system with queues with lengths of 5, 25 by 75 was analyzed. Experimental results reveal that the parameter has similar characteristics in all the models studied, as can be seen in Figures 17–19. As the value of ρ increases, there is a proportional increase in the value of the utilization parameter. The linear characteristics of the utilization parameter for the investigated classes occur in the range of ρ from 0 to 0.33. In this range, all classes proportionally share the entire available bandwidth of the transmission channel. For $\rho = 0.33$, class-1, class-2, and class-3 data use the entire available bandwidth, and the transmission channel is shared at a ratio of $1/3$ per class. As the amount of incoming data increases, the available bandwidth of the system decreases. Once all the bandwidth is used, i.e., for $\rho > 0.33$, we observe a process of data competition for access to the link. In the first instance, we observe a collapse in the linear characteristics of class-3 data, with the maximum utilization value for class-3 data occurring for $\rho = 0.33$ (see Figure 18). As the incoming data continue to increase, the utilization value decreases, but stabilizes at 0.2 for $\rho \geq 0.5$. A similar phenomenon is observed for class-2 data, but here, the maximum utilization value is higher, as class-2 data only compete for available bandwidth with class-1 data. As the ρ value increases, the utilization value stabilizes at 0.3 for $\rho \geq 0.5$. Class-1 data are the highest priority data and do not have to compete for available bandwidth with other data. The linear behavior of the utilization parameter for class-1 data occurs until it reaches a value of 0.5. For high throughputs, the value stabilizes and remains constant at the same level. Systems with queues of 5, 25, and 75 show no significant differences in the utilization parameter.

In the final stage, the behavior of the transmission channel was investigated with a large amount of incoming data into the system. The behavior of the transmission channel shared by the three data classes was analyzed as a function of class-1 data traffic intensity and at constant traffic intensities for class-2 and class-3 data (class-2 = 0.6 and class-3 = 0.3) (Figure 20). For class-3 data, channel utilization remains unchanged at the initial level for $\rho < 0.3$. Thereafter, we observe a decrease in channel utilization as the value of ρ increases. The value of the utilization parameter stabilizes at 0.2 for $\rho > 0.5$. For class-2 data, the channel utilization has a more dynamic course, and the channel utilization halves. From an initial value of utilization = 0.6, we observe a decrease to 0.3. The value of the utilization parameter stabilizes at 0.3 for $\rho > 0.5$. With class-2 data, channel utilization decreases for $\rho > 0.1$. This means that class-2 data start competing for link access when ρ reaches 0.1. Interestingly, class-3 data (lower-priority data) only start competing for link access when ρ reaches 0.3. This phenomenon is well illustrated by the AWT and AQL, as shown in Figures 21 and 22. The AWT and AQL studies carried out revealed that when there is a large volume of data traffic in the system, the performances of higher-priority

data can deteriorate. As can be seen in Figure 21, for $\rho < 0.2$, the average waiting time of class-2 data increases sharply, while lower-priority data (class-3 data) show much better characteristics in the same range. The AQL parameter shows a similar phenomenon. In the first stage, there is a sharp increase in the queue length for class-2 data, followed by a rise in the queue filling with class-3 data, which are lower-priority queues (Figure 22), i.e., with lower-priority queues. With class-1 data having the highest priority for both AWT and AQL parameters examined, it shows the best characteristics.

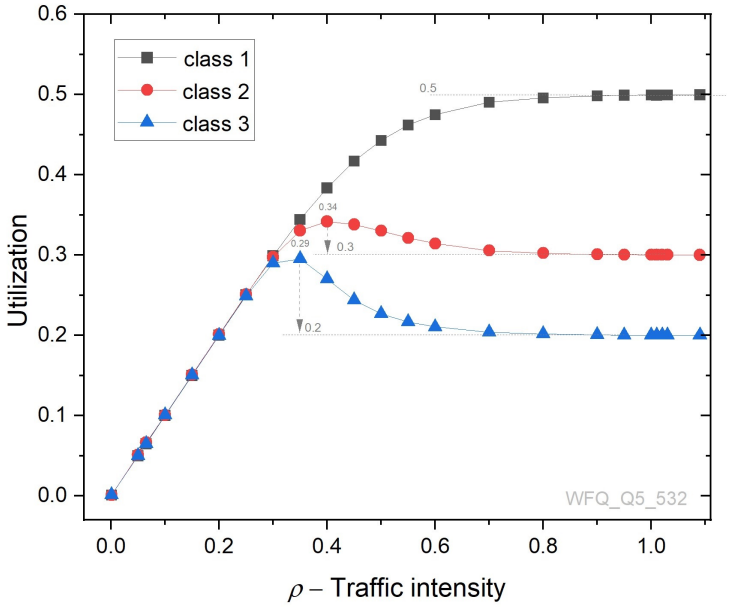

Figure 17. Utilization as a function of ρ with a queue length = 5.

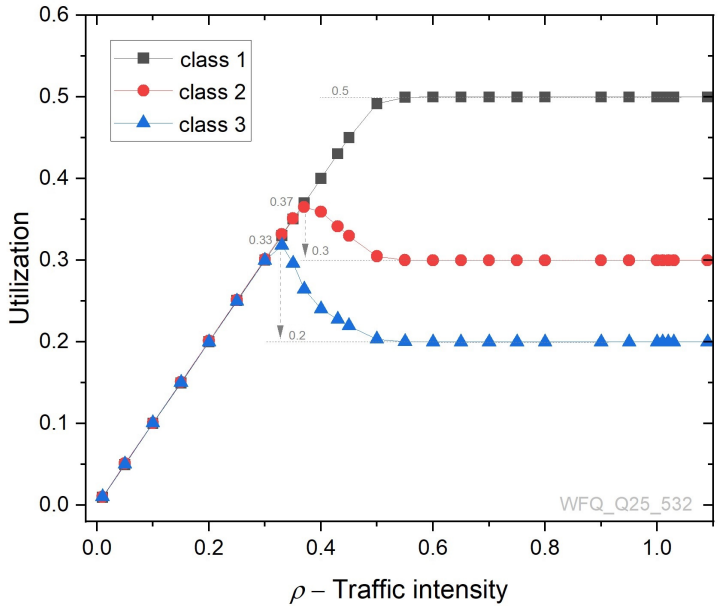

Figure 18. Utilization as a function of ρ with a queue length = 25.

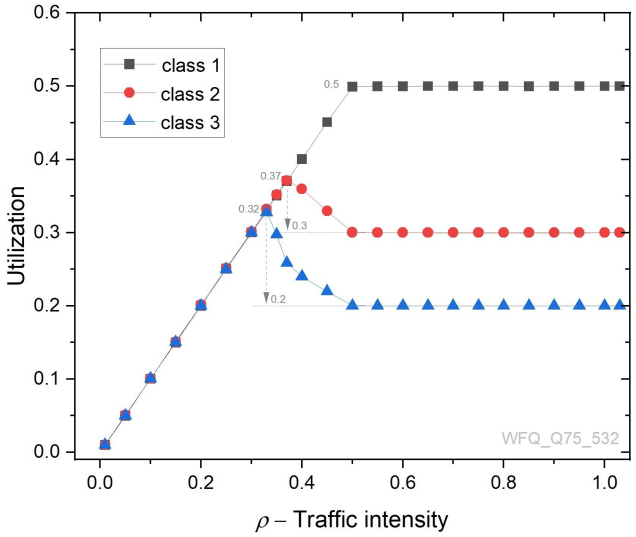

Figure 19. Utilization as a function of ρ with a queue length = 75.

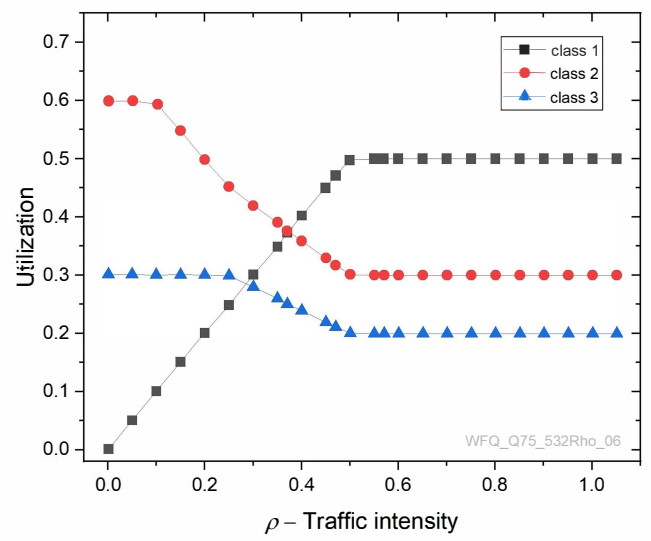

Figure 20. Utilization as a function of ρ_1 with $\rho_2 = 0.6$ and $\rho_3 = 0.3$; queue length = 75.

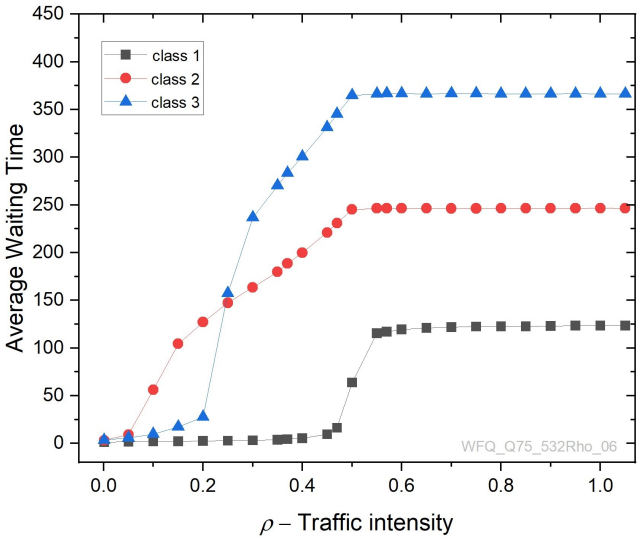

Figure 21. AWT as a function of ρ_1 with $\rho_2 = 0.6$ and $\rho_3 = 0.3$; queue length = 75.

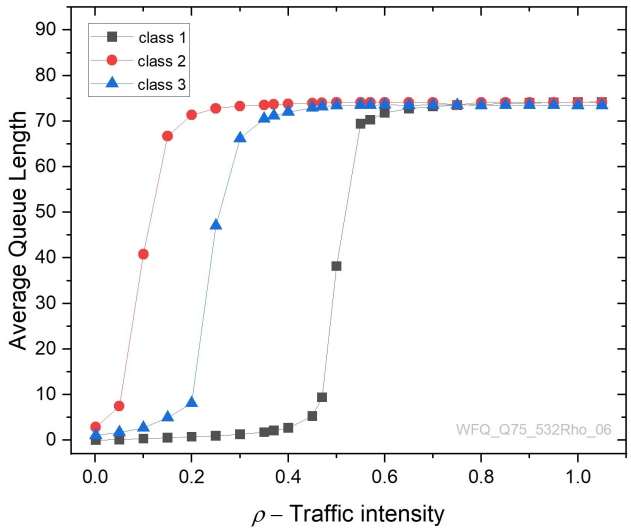

Figure 22. AQL as a function of ρ_1 with $\rho_2 = 0.6$ and $\rho_3 = 0.3$; queue length = 75.

10. Conclusions

The general behavioral characteristics of a system with priority queues are shown to eliminate the blocking of lower-priority traffic, which is typical of priority-based traffic management schemes. It was shown that characteristics in a range of up to $\rho < 0.30$ are similar in all models and have linear behavior. The tests carried out show that, in certain cases of a large influx of data routed to the system, the performance of the higher-priority queue may deteriorate, while the lower-priority data show much better characteristics at the same ρ values. It should be kept in mind that in real systems, the characteristics change frequently, and a method of dynamic weighting may be required to adjust the performance to the changing nature of the traffic. The huge amount of data transmitted by communication nodes require an adequate QoS, which is essential to ensure the complex quality of service parameters. Intensive use of resources can lead to dynamic changes in the system and a deterioration in the quality of service. It has been demonstrated that heavy system usage can result in longer queues at nodes, increased congestion, and extended waiting times. It has been shown that, in certain situations, the performance of higher-priority data can deteriorate in the system, while lower-priority data exhibit better parameters. The paper shows that the application of temporal Petri net models can be used to evaluate the performance and efficiency of priority queueing-based systems. Temporal Petri net models can be used to evaluate a range of system performance measures. The monitoring and control of priority systems is a very important challenge in QoS-enabled systems and, therefore, any insight that can help to understand and analyze this complex behavior is valuable.

Funding: This research received no external funding.

Data Availability Statement: Data supporting the reported results were obtained using the Petri Nets simulator.

Conflicts of Interest: The author declares no conflict of interest. Any funders had no role in the design of the study; in the collection, analysis, or interpretation of the data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Birolini, A. *Reliability Engineering: Theory and Practice*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2013.
2. Strzęciwilk, D. Examination of Transmission Quality in the IP Multi-Protocol Label Switching Corporate Networks. *Int. J. Electron. Telecommun.* **2012**, *58*, 267–272. [CrossRef]

3. Strzęciwilk, D.; Ptaszek, K.; Hoser, P.; Antoniku, I. A research on the impact of encryption algorithms on the quality of vpn tunnels' transmission. In *Proceedings of the ITM Web of Conferences*; EDP Sciences: Les Ulis, France, 2018; Volume 21, p. 00011. [CrossRef]
4. Ke, J.C.; Chang, C.J.; Chang, F.M. Controlling arrivals for a Markovian queueing system with a second optional service. *Int. J. Ind. Eng.* **2010**, *17*, 48–57.
5. Jain, M.; Kaur, S.; Singh, P. Supplementary variable technique (SVT) for non-Markovian single server queue with service interruption (QSI). *Oper. Res.* **2021**, *21*, 2203–2246. [CrossRef]
6. Czachórski, T.; Domański, A.; Domańska, J.; Rataj, A. A study of IP router queues with the use of Markov models. In *Proceedings of the International Conference on Computer Networks, Brunów, Poland, 14–17 June 2016*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 294–305. [CrossRef]
7. Strzęciwilk, D.; Nafkha, R.; Zawiślak, R. Performance analysis of a qos system with wfq queuing using temporal petri nets. In Proceedings of the International Conference on Computer Information Systems and Industrial Management, Ełk, Poland, 24–26 September 2021; Springer: Berlin/Heidelberg, Germany, 2021; pp. 462–476. [CrossRef]
8. Zuberek, W.M.; Strzeciwilk, D. Modeling Quality of Service Techniques for Packet-Switched Networks. *Dependabil. Eng.* **2018**, 125. [CrossRef]
9. Tarasiuk, H.; Hanczewski, S.; Kaliszan, A.; Szuman, R.; Ogrodowczyk, Ł.; Olszewski, I.; Giertych, M.; Wiśniewski, P. The IPv6 QoS system implementation in virtual infrastructure. *Telecommun. Syst.* **2016**, *61*, 221–233. [CrossRef]
10. Yu, Q.; Ren, J.; Fu, Y.; Li, Y.; Zhang, W. Cybertwin: An origin of next generation network architecture. *IEEE Wirel. Commun.* **2019**, *26*, 111–117. [CrossRef]
11. Shafique, K.; Khawaja, B.A.; Sabir, F.; Qazi, S.; Mustaqim, M. Internet of things (IoT) for next-generation smart systems: A review of current challenges, future trends and prospects for emerging 5G-IoT scenarios. *IEEE Access* **2020**, *8*, 23022–23040. [CrossRef]
12. Bensalah, F.; Bahnasse, A.; El Hamzaoui, M. Quality of service performance evaluation of next-generation network. In Proceedings of the 2019 2nd International Conference on Computer Applications & Information Security (ICCAIS), Riyadh, Saudi Arabia, 1–3 May 2019; pp. 1–5. [CrossRef]
13. Ak, E.; Canberk, B. Forecasting Quality of Service for Next-Generation Data-Driven WiFi6 Campus Networks. *IEEE Trans. Netw. Serv. Manag.* **2021**, *18*, 4744–4755. [CrossRef]
14. Ashton, K. That 'internet of things' thing. *RFID J.* **2009**, *22*, 97–114.
15. Liu, Y.; Dai, H.N.; Wang, Q.; Shukla, M.K.; Imran, M. Unmanned aerial vehicle for internet of everything: Opportunities and challenges. *Comput. Commun.* **2020**, *155*, 66–83. [CrossRef]
16. Guérin, R.; Peris, V. Quality-of-service in packet networks: Basic mechanisms and directions. *Comput. Netw.* **1999**, *31*, 169–189. [CrossRef]
17. Xiao, X.; Ni, L.M. Internet QoS: A big picture. *IEEE Netw.* **1999**, *13*, 8–18. [CrossRef]
18. Tomovic, S.; Radusinovic, I. A new traffic engineering approach for QoS provisioning and failure recovery in SDN-based ISP networks. In Proceedings of the 2018 23rd International Scientific-Professional Conference on Information Technology (IT), Zabljak, Montenegro, 19–24 February 2018; pp. 1–4. [CrossRef]
19. Modarressi, A.R.; Mohan, S. Control and management in next-generation networks: Challenges and opportunities. *IEEE Commun. Mag.* **2000**, *38*, 94–102. [CrossRef]
20. Taleb, T.; Aoul, Y.H.; Benslimane, A. Integrating security with qos in next generation networks. In Proceedings of the 2010 IEEE Global Telecommunications Conference GLOBECOM 2010, Miami, FL, USA, 6–10 December 2010; pp. 1–5. [CrossRef]
21. Houtan, B.; Ashjaei, M.; Daneshtalab, M.; Sjödin, M.; Mubeen, S. Synthesising schedules to improve QoS of best-effort traffic in TSN networks. In Proceedings of the 29th International Conference on Real-Time Networks and Systems, Nantes, France, 7–9 April 2021; pp. 68–77. [CrossRef]
22. Ardagna, D.; Casale, G.; Ciavotta, M.; Pérez, J.F.; Wang, W. Quality-of-service in cloud computing: Modeling techniques and their applications. *J. Internet Serv. Appl.* **2014**, *5*, 1–17. [CrossRef]
23. Abdelmaboud, A.; Jawawi, D.N.; Ghani, I.; Elsafi, A.; Kitchenham, B. Quality of service approaches in cloud computing: A systematic mapping study. *J. Syst. Softw.* **2015**, *101*, 159–179. [CrossRef]
24. White, G.; Nallur, V.; Clarke, S. Quality of service approaches in IoT: A systematic mapping. *J. Syst. Softw.* **2017**, *132*, 186–203. [CrossRef]
25. Bhuyan, B.; Sarma, H.K.D.; Sarma, N.; Kar, A.; Mall, R. Quality of service (QoS) provisions in wireless sensor networks and related challenges. *Wirel. Sens. Netw.* **2010**, *2*, 861. [CrossRef]
26. Letswamotse, B.B.; Malekian, R.; Chen, C.Y.; Modieginyane, K.M. Software defined wireless sensor networks (SDWSN): A review on efficient resources, applications and technologies. *J. Internet Technol.* **2018**, *19*, 1303–1313. [CrossRef]
27. Zave, P.; Rexford, J. The compositional architecture of the Internet. *Commun. ACM* **2019**, *62*, 78–87. [CrossRef]
28. Postel, J. *Rfc0791: Internet Protocol*; RFC Production Center: Marina del Rey, CA, USA, 1981.
29. Braden, R.; Clark, D.; Shenker, S. Rfc1633: Integrated Services in the Internet Architecture: An Overview. 1994. Available online: https://datatracker.ietf.org/doc/rfc1633/ (accessed on 8 November 2023).
30. Blake, S.; Black, D.; Carlson, M.; Davies, E.; Wang, Z.; Weiss, W. Rfc2475: An Architecture for Differentiated Services. Technical Report. December 1998. Updated by Rfc 3260. Available online: https://www.rfc-editor.org/info/rfc2475 (accessed on 8 November 2023).

31. Gevros, P.; Crowcroft, J.; Kirstein, P.; Bhatti, S. Congestion control mechanisms and the best effort service model. *IEEE Netw.* **2001**, *15*, 16–26. [CrossRef]
32. Shenker, S.; Partridge, C.; Guerin, R. *RFC2212: Specification of Guaranteed Quality of Service*; RFC Production Center: Marina del Rey, CA, USA, 1997.
33. Wroclawski, J. RFC 2211: Specification of the Controlled-Load Network Element Service. RFC Editor 1997. Available online: https://www.rfc-editor.org/info/rfc2211 (accessed on 8 November 2023).
34. Zuberek, W.M.; Strzeciwilk, D. Modeling traffic shaping and traffic policing in packet-switched networks. *J. Comput. Sci. Appl.* **2018**, *6*, 75–81. http://pubs.sciepub.com/jcsa/6/2/4 (accessed on 8 November 2023).
35. Braden, R.; Zhang, L.; Berson, S.; Herzog, S.; Jamin, S. *Resource Reservation Protocol (RSVP)—Version 1 Functional Specification*; Technical Report; RFC Production Center: Marina del Rey, CA, USA, 1997. [CrossRef]
36. Zhang, L.; Deering, S.; Estrin, D.; Shenker, S.; Zappala, D. RSVP: A new resource reservation protocol. *IEEE Netw.* **1993**, *7*, 8–18. [CrossRef]
37. Singh, H.P.; Singh, S.; Singh, J.; Khan, S.A. VoIP: State of art for global connectivity—A critical review. *J. Netw. Comput. Appl.* **2014**, *37*, 365–379. [CrossRef]
38. Kim, H.J.; Choi, S.G. A study on a QoS/QoE correlation model for QoE evaluation on IPTV service. In Proceedings of the 2010 The 12th International Conference on Advanced Communication Technology (ICACT),Gangwon, Republic of Korea, 7–10 February 2010; Volume 2, pp. 1377–1382.
39. Choi, J.; Reaz, A.S.; Mukherjee, B. A survey of user behavior in VoD service and bandwidth-saving multicast streaming schemes. *IEEE Commun. Surv. Tutorials* **2011**, *14*, 156–169. [CrossRef]
40. Jacobson, V.; Nichols, K.; Poduri, K. Rfc2598: An Expedited Forwarding PHB. RFC Editor, June 1999. Available online: https://www.rfc-editor.org/info/rfc2598 (accessed on 8 November 2023).
41. Davie, B.; Charny, A.; Bennet, J.; Benson, K.; Boudec, J.L.; Courtney, W.; Davari, S.; Firoiu, V.; Stiliadis, D. *Rfc3246: An Expedited Forwarding PHB (Per-Hop Behavior)*; RFC Production Center: Marina del Rey, CA, USA, 2002.
42. Heinanen, J.; Baker, F.; Weiss, W.; Wroclawski, J. *lRFC2597: Assured Forwarding PHB Group*; RFC Production Center: Marina del Rey, CA, USA, 1999.
43. Nichols, K.; Blake, S.; Baker, F.; Black, D. *RFC2474: Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers*; RFC Production Center: Marina del Rey, CA, USA, 1998.
44. Cruz, R.L. SCED+: Efficient management of quality of service guarantees. In Proceedings of the IEEE INFOCOM'98, the Conference on Computer Communications, Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies. Gateway to the 21st Century (Cat. No. 98), San Francisco, CA, USA, 29 March–2 April 1998; Volume 2, pp. 625–634. [CrossRef]
45. Rusek, K.; Janowski, L.; Papir, Z. Correct router interface modeling. In Proceedings of the 2nd ACM/SPEC International Conference on Performance Engineering, Karlsruhe, Germany, 14–16 March 2011; pp. 97–102. [CrossRef]
46. Babiarz, J.; Chan, K.; Baker, F. *Configuration Guidelines for DiffServ Service Classes*; Technical Report; RFC Production Center: Marina del Rey, CA, USA, 2006.
47. Jiang, Y. Link-based fair aggregation: A simple approach to scalable support of per-flow service guarantees. In Proceedings of the International Conference on Research in Networking, Athens, Greece, 9–14 May 2004; Springer: Berlin/Heidelberg, Germany, 2004; pp. 1084–1095. [CrossRef]
48. Goyal, P.; Lam, S.S.; Vin, H.M. Determining end-to-end delay bounds in heterogeneous networks. In Proceedings of the International Workshop on Network and Operating Systems Support for Digital Audio and Video, Durham, NH, USA, 19–21 April 1995; Springer: Berlin/Heidelberg, Germany, 1995; pp. 273–284. [CrossRef]
49. Saaty, T.L. *Elements of Queueing Theory: With Applications*; McGraw-Hill: New York, NY, USA, 1961; Volume 34203.
50. Rosanov, Y.A. List of Publications by Yu. V. Prokhorov. In Proceedings of the Mathematical Methods for Engineering Applications, Madrid, Spain, 12–14 July 2013. [CrossRef]
51. Kalashnikov, V.V. *Mathematical Methods in Queuing Theory*; Springer: Berlin/Heidelberg, Germany, 2013; Volume 271.
52. Sundarapandian, V. *Queueing Theory*; PHI Learning Pvt. Ltd.: New Delhi, India, 2009; Volume 59, p. 60.
53. Nelson, R. *Probability, Stochastic Processes, and Queueing Theory: The Mathematics of Computer Performance Modeling*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2013. [CrossRef]
54. Buchholz, P.; Kriege, J.; Felko, I. *Input Modeling with Phase-Type Distributions and Markov Models: Theory and Applications*; Springer: Berlin/Heidelberg, Germany, 2014. [CrossRef]
55. Donthi, D.R. Study of delay and loss behavior of Internet switch-Markovian modelling using circulant Markov modulated Poisson process (CMMPP). *Appl. Math.* **2014**, *5*, 512–519. [CrossRef]
56. Puterman, M.L. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*; John Wiley & Sons: Hoboken, NJ, USA, 2014. [CrossRef]
57. Bhatti, S.N.; Crowcroft, J. QoS-sensitive flows: Issues in IP packet handling. *IEEE Internet Comput.* **2000**, *4*, 48–57. [CrossRef]
58. Bose, S.K. *An Introduction to Queueing Systems*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2013. [CrossRef]
59. Georges, J.P.; Divoux, T.; Rondeau, E. Strict priority versus weighted fair queueing in switched ethernet networks for time critical applications. In Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium, Denver, CO, USA, 4–8 April 2005; p. 8. [CrossRef]

60. Mishkoy, G.; Giordano, S.; Andronati, N.; Bejan, A. Priority queueing systems with switchover times: Generalized models for QoS and CoS network technologies and analysis. In Proceedings of the XIV Conference on Applied and Industrial Mathematics, Satellite Conference of ICM, Chisinau, Chisinau, Moldova, 17–19 August 2006; pp. 244–247.
61. Tabatabaee, S.M.; Le Boudec, J.Y. Deficit round-robin: A second network calculus analysis. *IEEE/ACM Trans. Netw.* **2022**, *30*, 2216–2230. [CrossRef]
62. Khalil, F.K.; Khan, S.; Faisal, F.; Nawaz, M.; Javed, F.; Khan, F.A.; Noor, R.M.; Shoaib, M.; Masood, F.U. Quality of Service Impact on Deficit Round Robin and Stochastic Fair Queuing Mechanism in Wired-cum-Wireless Network. *Int. J. Adv. Comput. Sci. Appl.* **2018**, *9*, 287–293. [CrossRef]
63. Aurrecoechea, C.; Campbell, A.T.; Hauw, L. A survey of QoS architectures. *Multimed. Syst.* **1998**, *6*, 138–151. [CrossRef]
64. Kendall, D.G. Some problems in the theory of queues. *J. R. Stat. Soc. Ser. B Methodol.* **1951**, *13*, 151–173. [CrossRef]
65. Kleinrock, L. *Queueing Systems, Volume 1: Theory by Leonard Kleinrock*; John Wiley & Sons: Hoboken, NJ, USA, 1975; Volume 19, p. 417. [CrossRef]
66. Petri, C. Kommunikation mit Automaten. Ph.D. Thesis, University of Bonn, Bonn, Germany, 1962.
67. Peterson, J.L. Petri nets. *ACM Comput. Surv.* **1977**, *9*, 223–252. [CrossRef]
68. Murata, T. Petri nets: Properties, analysis and applications. *Proc. IEEE* **1989**, *77*, 541–580. [CrossRef]
69. Reisig, W. *Petri Nets: An Introduction*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2012; Volume 4. [CrossRef]
70. Peterson, J.L. *Petri Net Theory and the Modeling of Systems*; Prentice Hall PTR: Hoboken, NJ, USA, 1981.
71. Bernardini, F.; Gheorghe, M.; Margenstern, M.; Verlan, S. Producer/consumer in membrane systems and Petri nets. In Proceedings of the Conference on Computability in Europe, Siena, Italy, 18–23 June 2007; Springer: Berlin/Heidelberg, Germany, 2007; pp. 43–52. [CrossRef]
72. Bause, F.; Kritzinger, P.S. *Stochastic Petri Nets*; Citeseer: Pennsylvania, PE, USA, 2002; Volume 1.
73. Jensen, K.; Kristensen, L.M. *Coloured Petri Nets: Modelling and Validation of Concurrent Systems*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2009. [CrossRef]
74. Liu, H.C.; You, J.X.; Li, Z.; Tian, G. Fuzzy Petri nets for knowledge representation and reasoning: A literature review. *Eng. Appl. Artif. Intell.* **2017**, *60*, 45–56. [CrossRef]
75. Zuberek, W.M. Timed Petri nets definitions, properties, and applications. *Microelectron. Reliab.* **1991**, *31*, 627–644. [CrossRef]
76. Huang, Y.S.; Weng, Y.S.; Zhou, M. Modular design of urban traffic-light control systems based on synchronized timed Petri nets. *IEEE Trans. Intell. Transp. Syst.* **2013**, *15*, 530–539. [CrossRef]
77. Zuberek, W. D-timed Petri nets and modeling of timeouts and protocols. *Trans. Soc. Comp. Simul.* **1987**, *4*, 331–357.
78. Zuberek, W.M. M-timed Petri nets, priorities, preemptions, and performance evaluation of systems. In Proceedings of the European Workshop on Applications and Theory in Petri Nets, Xiamen, China, 13–15 October 1985; Springer: Berlin/Heidelberg, Germany, 1985; pp. 478–498. [CrossRef]
79. Popova-Zeugmann, L. *Time Petri Nets*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 31–137. [CrossRef]
80. Berthomieu, B.; Diaz, M. Modeling and verification of time dependent systems using time Petri nets. *IEEE Trans. Softw. Eng.* **1991**, *17*, 259. [CrossRef]