

*Transport and Telecommunication*, 2026, volume 27, no. 3, 249-259  
*Transport and Telecommunication Institute, Lauvas 2, Riga, LV-1019, Latvia*  
DOI 10.2478/ttj-2026-0018

## SIMULATION OF REAL-TIME SYSTEMS

***Sławomir Hanczewski<sup>1</sup>, Joanna Weissenberg<sup>2</sup>***

*Faculty of Computing and Telecommunications, Poznan University of Technology*  
*Poznań, Poland, Jacka Rychlewskiego 1, 61-131*

*<sup>1</sup>slawomir.hanczewski@put.poznan.pl, <sup>2</sup>joanna.weissenberg@put.poznan.pl*

This article presents the development of a real-time system simulator and discusses the results obtained through its application. The simulator is designed in accordance with the fundamental principles of real-time systems, incorporating traffic sources that generate both hard and soft data streams. Hard data streams demand stringent transmission conditions, such as minimal latency beyond what is required for essential operations to ensure accurate data transmission. Conversely, soft data streams allow for variations in transmission parameters. The simulator was developed using the event scheduling approach and implemented in the C++ programming language.

**Keywords:** Real-time systems, simulation, statistical analysis

### 1. Introduction

The rapid development of wireless transmission technology has led to the successful implementation, or consideration for implementation, of such networks in systems of critical importance, such as public safety or applications requiring high reliability (e.g., Industry 4.0, communication Vehicle-to-Everything (V2X)). In these systems, the response time to input data, such as data transmitted from a sensor, is of paramount importance. In the literature, these systems are referred to as real-time systems. They are described in ISO/IEC and IEEE documentation (ISO/IEC/IEEE, 2017) and in the Oxford Dictionary (Butterfield *et al.*, 2016). Although expressed in different forms, these definitions characterise real-time systems as those in which the system's response time to input data or specific events occurring within the system is crucial to its proper functioning. This response time must be finite and explicitly defined. A delayed response may lead to system instability or incorrect operation. Real-time systems have been a subject of research for many years. The literature includes numerous studies on task (or process) scheduling algorithms in computer systems, which aim to minimize system response time to real-time tasks (Szalay *et al.*, 2021; Li *et al.*, 2022; Baital and Chakrabarti, 2019). Another group of studies addresses the issue of data transmission for real-time services, requiring transmission without additional delays beyond those inherent in the data transmission process itself (Chen *et al.*, 2024; Zhang *et al.*, 2023; Bekkali *et al.*, 2017). The most typical real-time systems include medical support systems, air traffic control systems, train control systems, Industry 4.0 applications, and many others. These are systems that handle applications (and the traffic generated by them) requiring low latency and zero or strictly defined data loss.

When faced with the task of designing new telecommunication systems or the devices that constitute their components, engineers must decide how these designs will operate. Since these systems do not yet physically exist, it is impossible to evaluate their behaviour, even under laboratory conditions (Banks *et al.*, 1996). In such scenarios, digital simulation becomes invaluable. This technique allows for the modelling and analysis of various telecommunication systems (and beyond). Digital simulation enables the construction of a digital model of the system under study, allowing for the modification of system parameters and, consequently, the observation of its behaviour under variable, strictly controlled conditions. Simulation also facilitates the optimisation of system performance, for example, by selecting appropriate system parameter values. Developing a simulation model of the system under study is often less costly, time-consuming, and significantly safer than constructing even a prototype of the telecommunication system.

Constructing a simulation model of the system under study involves using computers to imitate the real processes of system operation at a certain accepted level of accuracy. This model is then used to conduct experiments in a virtual environment. Two simulation methods can be employed for telecommunication systems: The Monte Carlo method and discrete-event simulation (Banks *et al.*, 1996; Tyszer, 1999). While both methods can be applied to simulating ICT systems, when it is necessary to determine time-dependent parameters, discrete-event simulation should be used. This method is successfully applied not only in

computer science but also in modelling business processes, production systems, and logistics. In this approach, the system is represented by its resources and as a sequence of events that change its state. The simulation process tracks these events and analyses their impact on the entire system (Tyszer, 1999).

The article presents the assumptions behind the construction and the sample results obtained using a simulator designed to meet the conditions of a real-time system. The simulator does not consider process scheduling algorithms that would guarantee a predetermined response time, but rather addresses the issue of data transmission generated by devices so that it can be processed in accordance with real-time system requirements, such as without additional delays.

The rest of the article is organised as follows. Section 2 provides fundamental information on the construction and functionality of the simulator. Section 3 presents the results obtained, including the determination of blocking probability, occupancy distribution, delay for soft real-time flows, compression ratio, and simulation execution time on two different computers for the selected system. Section 4 provides a brief summary of the article.

## 2. Structure and functionality of the simulator

The simulation process is organised according to the event scheduling method, which is described in detail in works by Tyszer (1999) and Banks *et al.* (1999). Figure 1 shows a simplified diagram of the real-time system simulator presented in this article. The modular design of the simulator allows for easy control over the conducted simulation experiments, which was crucial during its implementation and testing phases. This structure also enables the expansion of the simulator to include new telecommunication systems. Five modules are distinguished: system structure, input data, flow generators, flow service, and output data. These modules are described in more detail in the following sections of this chapter.

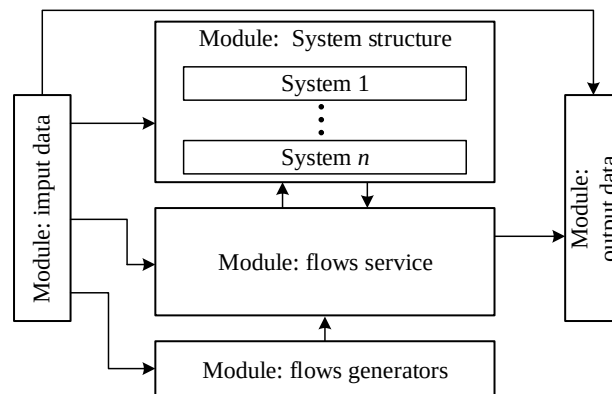


Figure 1. Block diagram of simulator

### 2.1. Module: system structure

This module represents the structure of the telecommunication system under study, allowing for the consideration of available resources and the determination of the system's capacity. The module is also responsible for storing and updating information about the resource utilisation of the studied systems during the simulation. Since the developed simulation model addresses data transmission issues, its capacity should be expressed in bps (bits per second). However, to streamline the simulation process, the system capacity is expressed in system capacity units (AU). One AU corresponds to a specific transmission rate in bps. The transmission rate associated with one AU is determined based on the rates required by the transmitted flows (data streams) using a process known as resource discretisation (Stasiak *et al.*, 2011). If one AU corresponds to 1 bps, then the capacity expressed in AU numerically matches the rate expressed in bps. The system capacity is denoted as  $C_r$ . In addition, to ensure simultaneous operation of flows of type hard and soft, for flows of type soft there is a limitation  $R$ . This parameter determines the maximum number of AU in the system, which can be occupied by flows of type hard. The  $R$  parameter takes values less than  $C_r$ . Due to the nature of the data streams (see subsection 2.4), the simulation model includes the possibility of compressing transmitted data streams. Generally, the compression mechanism involves reducing the available transmission rate for selected flows. This allows a greater number of flows to be transmitted simultaneously, but at a reduced rate. Figure 2 illustrates a system with a capacity of  $C_r = 4$  AUs. Under normal conditions (i.e., without compression), it can service two flows, each requiring 2 AUs (Figure 2a).

With compression, such a system can service more flows simultaneously, assuming their transmission rate is limited. This situation is depicted in Figure 2b. The system can service four flows, but each is allocated one AU. Using compression in the simulated system requires recalculating the resources available for each flow, which prolongs the simulation process. To avoid this, the simulator uses an alternative model of available resources, incorporating a virtual capacity  $C_v$  (see Figure 2c). This capacity defines the system's ability to compress flows. Thus, from the perspective of compressed flows, the system's capacity is greater, ultimately resulting in the ability to accommodate a larger number of flows (in the example considered, four flows).

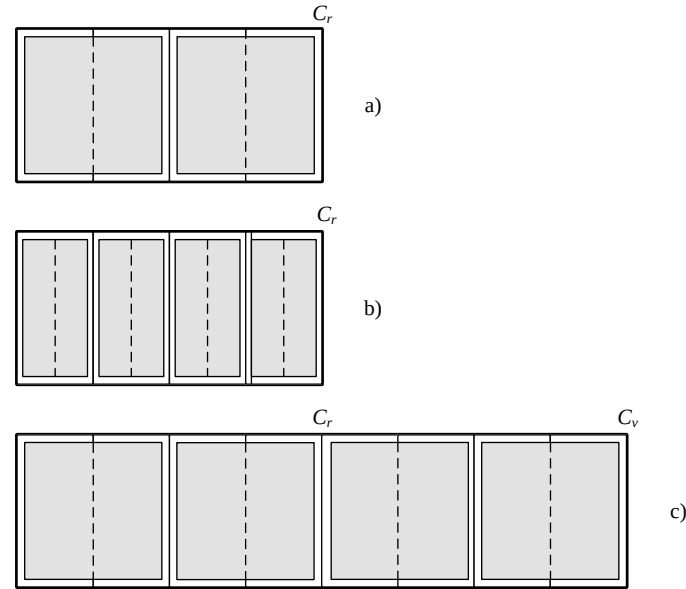


Figure 2. Resources of the simulated system, a) system without compression, b) system with compression, c) alternative model of system with compression

## 2.2. Module: input data

The input data module allows the program to receive the data necessary for the correct execution of the planned simulation experiment. This data pertains to the structure of the system under study, including its capacity and the relationships between individual components, as well as the parameters of the flow that the system is to service. In the simulator, flows are described using three parameters: the number of AUs required by a flow for processing -  $c_j$  (where  $j$  is identifier of class of flows, the arrival rate of new flows ( $\lambda_j$ ) and the parameter specifying the required service time ( $\mu_j$ ). Flows requiring the same number of AUs belong to the same class. The total number of flows classes is  $m$ . The ratio of the last two parameters, known as the offered load in traffic engineering (Stasiak *et al.*, 2011) is given by:

$$A_j = \frac{\lambda_j}{\mu_j}. \quad (1)$$

Input parameters also include data defining the simulation experiment itself, such as the length of the preliminary phase, the length of the main phase, and the number of main phases needed to determine a single result. The data is input into the simulator by reading a properly prepared text file. This solution is due to the fact that a graphical user interface was not implemented, as it was not necessary during the research phase.

## 2.3. Module: flows generators

The correct implementation of the algorithm governing the operation of the telecommunication system under study is one of the primary conditions for the effective construction of its simulator. An equally important element is the use of pseudorandom number generators (PRNGs). In the modelling of telecommunication systems, discrete event simulation is most commonly employed, where PRNGs are used to determine the timing of events. For example, in a computer system simulator, these would be the time moments when IP packets appear, as well as their lengths. In real systems, the appearance of new call/tasks/processes and their duration are characteristic of the system and are described using probability

distributions. For the simulator to function correctly, the generators used must match these real-world distributions.

In the literature, many types of generators can be found (Tian and Benkrid, 2009; Sujitha *et al.*, 2023; Zheng *et al.*, 2024). According to the definition, a PRNG is an algorithm that allows the determination of successive numbers that mimic the properties of random numbers. These numbers are called pseudorandom because, although they appear random, they are generated deterministically, based on an initial value known as the "seed." The basic properties of PRNGs include determinism (the generated sequence of numbers is determined by the adopted initial value—seed; with the same seed, the generator generates an identical sequence of numbers), periodicity (due to limitations, the generated sequence of numbers will eventually repeat), efficiency (the implementation of the generator should be simple, the time needed to generate numbers should be appropriately short, and the generator itself should not consume large amounts of resources), and good statistical properties (the generated sequence of numbers should meet statistical criteria, e.g., correlation, so that it is difficult to distinguish from true random numbers).

Among the most popular generators are linear congruential generators (LCGs) (Tyszer, 1999; Banks *et al.*, 1996). Although this is not a new solution, it can still be successfully used in the implementation of ICT system simulators. However, due to their properties, they should not be used in cryptography. The foundation of PRNGs with various distributions is uniform distribution generators in the range  $\{0,1\}$ . One variant of the aforementioned congruential generators is the multiplicative generator. These generators operate according to the following recursive relationship:

$$x_{i+1} = ax_i \pmod{m}, \quad (2)$$

where  $x_i$  represents the pseudorandom number generated in step  $i$ ,  $a$  is an appropriately chosen constant, and  $m$  represents the range of integers that can be represented in a word of length  $b$  bits:

$$m = 2^b - 1. \quad (3)$$

Due to the nature of traffic in the considered real-time system, the generator described by equation (2) forms the basis for constructing a PRNG with an exponential distribution. The inverse transform method is used to construct such a generator, which ultimately leads to the following relationship describing the generator:

$$y_i = -\frac{1}{\lambda} \ln(1 - x_i), \quad (4)$$

where  $y_i$  is the generated pseudorandom number with an exponential distribution,  $\lambda$  is the intensity parameter of the distribution, and  $x_i$  is the generated pseudorandom number with a uniform distribution in the range  $\{0,1\}$ .

#### 2.4. Module: flows service

The flows service module is responsible for servicing calls in accordance with the real-time algorithm adopted for the system under study. Within this module, decisions are made regarding which flow will be transmitted without delay, which will be delayed, and which will be rejected. This module also collects information on the flows that have occurred in the system under study, including, but not limited to, the number of lost flows, blocking time, and delay. The requirements defined for real-time systems allow the categorization of the traffic serviced by these systems into two types: hard and soft real-time services.

Hard real-time services generate flows for which QoS parameters must not deteriorate (Eckhardt and Steenkiste, 1996). These parameters remain unchanged regardless of the system's load. Such services are critical in nature, characterized by the absence of delays, low loss rates, and high reliability of data transmission. In contrast, soft real-time services generate flows for which delays in service delivery or reductions in transmission speed, leading to the degradation of QoS parameters, are permissible within a predetermined, acceptable range (Buttazzo *et al.*, 2005).

In the simulation environment, hard real-time services are defined as services whose quality parameters (such as the number of required resources and call processing time) do not change based on system load (Figure 3a). This means that a call for a hard real-time service will be dropped if the system cannot guarantee its delivery with the specified quality parameters (e.g., when insufficient resources are available). Soft real-time services are further divided into two service classes: elastic and adaptive services. Both elastic and adaptive real-time services generate flows that can be admitted for processing with a predefined maximum degradation of service quality parameters. In the case of elastic services, the number of requested resources (bitrate) may be reduced, and the call processing time may be extended (Figure 3b).

Processing time is extended to ensure that all data is sent. For adaptive real-time services, only the number of allocated resources (bitrate) changes (Figure 3c). However, this comes at the expense of the amount of data transferred.

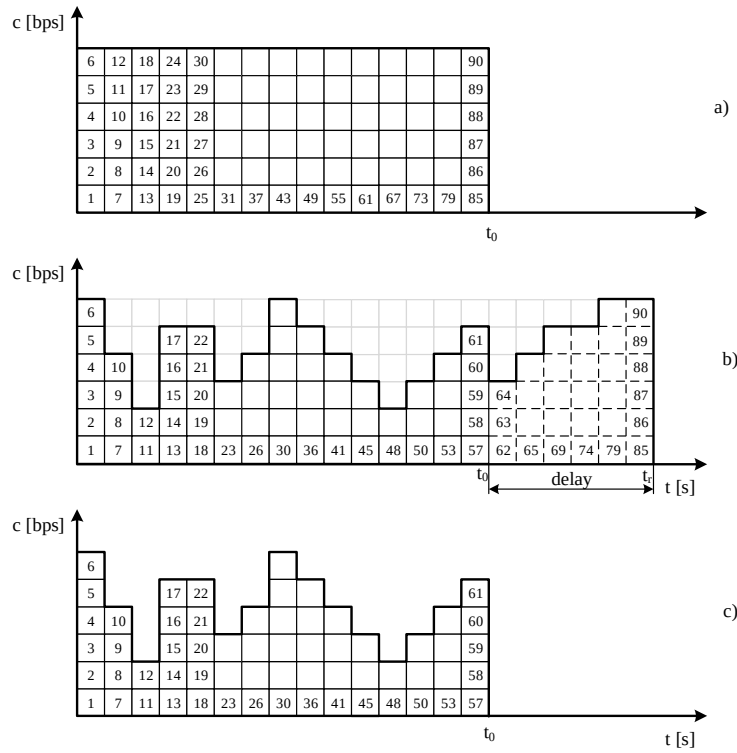


Figure 3. Traffic in the real-time system: a) traffic generated by hard real-time services, b) traffic generated by soft real-time services with the potential to extend data transmission time (elastic traffic), c) traffic generated by soft real-time services without the potential to extend data transmission time (adaptive traffic)

Compression, or the system's ability to accept flows with altered QoS parameters (such as processing time and the number of required Allocation Units (AUs)), is described in the simulator by the compression factor. This parameter is defined when the sum of AUs occupied by hard flows ( $n_r$ ) and soft flows ( $n_k$ ) exceeds  $C_r$ . All flows affected by this mechanism are compressed in the same manner (with an identical compression factor). The compression factor is defined as follows (Hanczewski *et al.*, 2025):

$$\xi = \frac{C_r - n_r}{n_k} \text{ for } n_r + n_k > C_r. \quad (5)$$

The significance of Equation (5) is intuitive, expressing how many AUs in the system, not occupied by hard real-time flows, are used to service soft real-time flows.

## 2.5. Module: output data

The output data module is responsible for processing the data obtained during the conducted simulation experiment. The simulation experiment consists of 10 series of simulations, during which selected characteristic system parameters are measured, such as the number of lost calls, blocking time, or delay for each flow offered to the system. Since the standard deviations of the studied characteristic parameters are unknown and the sample size is small, the Student's t-distribution at a 95% confidence level was used to determine confidence intervals for the expected value of each considered parameter. This implies that with a probability of 0.95, the estimated values of the parameters fall within the intervals

$$\left( \bar{x} - t_{\alpha, s} \frac{\hat{s}}{\sqrt{n}}; \bar{x} + t_{\alpha, s} \frac{\hat{s}}{\sqrt{n}} \right), \quad (6)$$

where

$t_{\alpha, s}$  represents the  $t$ -statistic value from the Student's  $t$ -distribution with 9 ( $n-1$ ) degrees of freedom and a 5% significance level,

the parameter  $\bar{x}$  represents the mean value of the studied feature, which is calculated using the formula:

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i, \quad (7)$$

the parameter  $\hat{s}$  represents the standard deviation of the feature and is defined by the formula

$$\hat{s} = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2}, \quad (8)$$

where  $n$  denotes the number of simulation series (number of measurements  $n=10$ ), and  $x_i$  is the measurement result in the  $i$ -th simulation series.

The processed results are saved in formatted text files. As previously mentioned, the simulator does not have a graphical interface for visualizing the obtained results as graphs. Therefore, the Gnuplot program (Gnuplot, 2024) was used for this purpose. This program is a free environment for creating plots, including three-dimensional graphs.

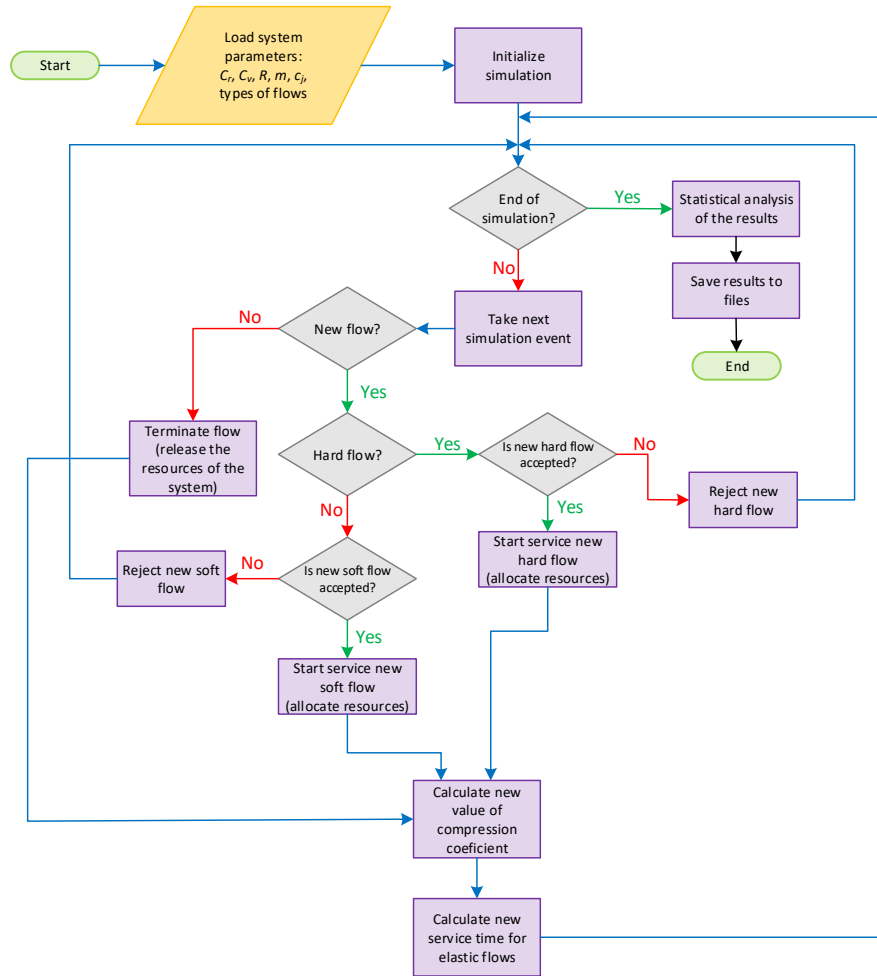


Figure 4. The general algorithm of the proposed simulator

## 2.6. Algorithm

Figure 4 presents the algorithm of the simulator described in the article. Upon launching the program, data regarding the system under analysis is retrieved from a previously prepared file. This includes the system's real capacity ( $C_r$ ), virtual capacity ( $C_v$ ), and constraint ( $R$ ), as well as information about the number of flow classes handled by the system ( $m$ ), their resource requirements ( $c_j, 1 \leq c_j \leq m$ ), and their types (hard real-time: stream flows; soft real-time: elastic and adaptive flows). Once this data has been read, the simulation process is initialised. This includes, among other things, the preparation of appropriate pseudorandom number generators responsible for the timing of new flow arrivals and their service time. The initial simulation events are also generated and placed on the event list. Next, after verifying the simulation end condition, the first event is retrieved from the event list and the system time is updated. If

the currently analysed event is the end of a flow's service, the system resources occupied by that flow are released (i.e. the resources used by hard ( $n_r$ ) and soft ( $n_k$ ) flows). Due to the change in system load following the completion of the flow's service, the compression ratio may change. The new ratio is calculated (see Formula (5)), and the service times for elastic flows are recalculated accordingly. Since releasing system resources leads to a reduction in compression, the service times of elastic flows are shortened. If the analysed event is the arrival of a new flow, its type is determined and a check is performed to see whether it can be admitted for service. In the case of hard flows, admission depends on the fulfilment of the following conditions:

$$R - n_r \geq c_j \text{ and } C_r + C_v - n_r - n_k \geq c_j. \quad (9)$$

This condition means that a new hard-type flow will be admitted if its inclusion does not cause the threshold  $R$  to be exceeded and, at the same time, the total number of occupied AU does not surpass the value of  $C_r + C_v$ . In the case of soft-type flows, the admission condition is simpler and involves only checking whether admitting the flow would exceed the system's capacity:

$$C_r + C_v - n_r - n_k \geq c_j. \quad (10)$$

Admitting a new flow for service requires updating the counters of occupied resources ( $n_r$  and  $n_k$ ), as well as recalculating the compression ratio, which in turn determines the new service times for elastic flows. Once the flow-handling procedure has been completed, the algorithm proceeds to the function that checks whether the simulation should end. The simulation end is determined based on the number of flows that have appeared in the system.

### 3. Results

This chapter presents the results obtained from the conducted studies. The research was performed for a selected system with specified parameters. The choice of the system was intended to best demonstrate the capabilities of the developed simulator.

The examined system was defined as follows:  $C_r = 30$  AUs,  $C_v = 50$  AUs. The system offers one class of hard real-time services, which requires 1 AU for processing, and two classes of soft services: one elastic service demanding 2 AUs for processing, and one adaptive service demanding 3 AUs for servicing.

One of the fundamental parameters that can be determined from the simulation studies is the blocking probability. In multi-service systems under consideration, this probability is defined as the ratio of the time the system spends in a blocked state (i.e., the time during which new flows of a given class cannot be accepted for servicing due to lack of available resources) to the total observation time of the system. The second basic parameter is loss probability, defined as the ratio of lost flows of given class to the total number of flows of this class that appeared in the system. Figures 5a and 5b show the obtained results for the basic system capabilities: blocking probability (Figure 5a) and loss probability (Figure 5b) as a function of the offered traffic per individual AU of the system:

$$a = \frac{\sum_{j=0}^m A_j}{C_r}, \quad (11)$$

where,  $A_j$  denotes the offered traffic by flows of class (see equation (1)).

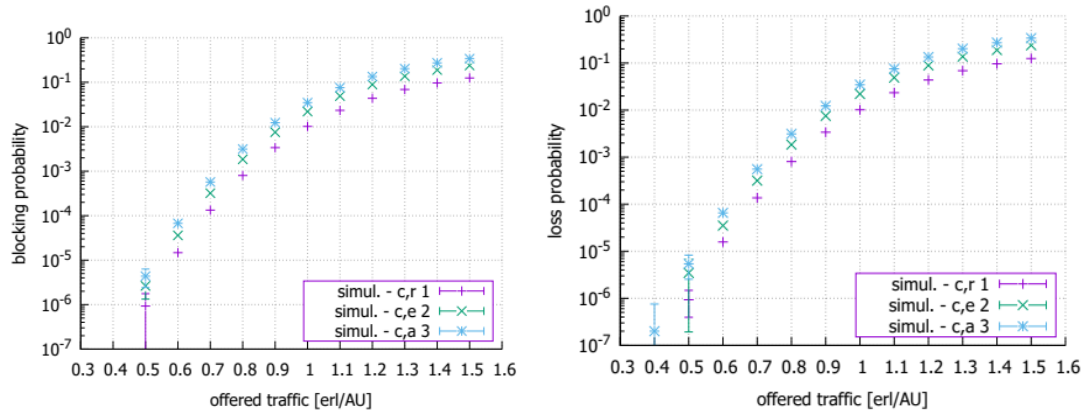


Figure 5. The fundamental parameters of the examined system a) blocking probability, b) loss probability

It should be noted that the values of these two parameters, although defined differently, are equal. This equality is due to the characteristics of the offered traffic. Noticeable intervals of confidence are due to the small number of requests for a given class. This issue can be mitigated by increasing the length of the simulation series for small values.

Another interesting aspect is the probability distribution within the system. It determines the probability of a given number of AUs being occupied in the system. Due to the alternative system structure assumed in the simulator, the distribution also takes into account virtual capacity. Subsequent graphs (Figure 6) illustrate distributions for the following values of  $a$ : 0.4, 0.7, and 1 Erl.

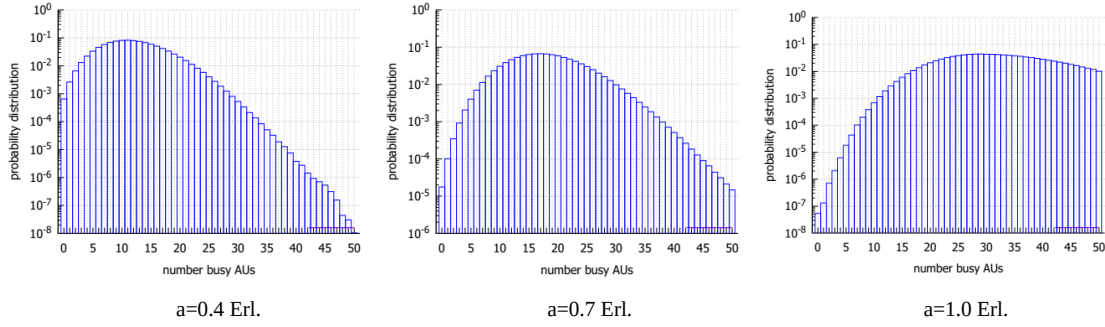


Figure 6. Occupancy distribution in the examined system

Since the examined system handles both hard and soft real-time flows, the occupancy distribution can be presented as a function of the number of AUs occupied by hard flows ( $n_r$ ) and the number of AUs occupied by soft flows ( $n_k$ ). This allows us to determine how many resources are occupied by requests of each type of real-time traffic (hard and soft). Heat-map graphs were chosen for visualisation (Figure 7).

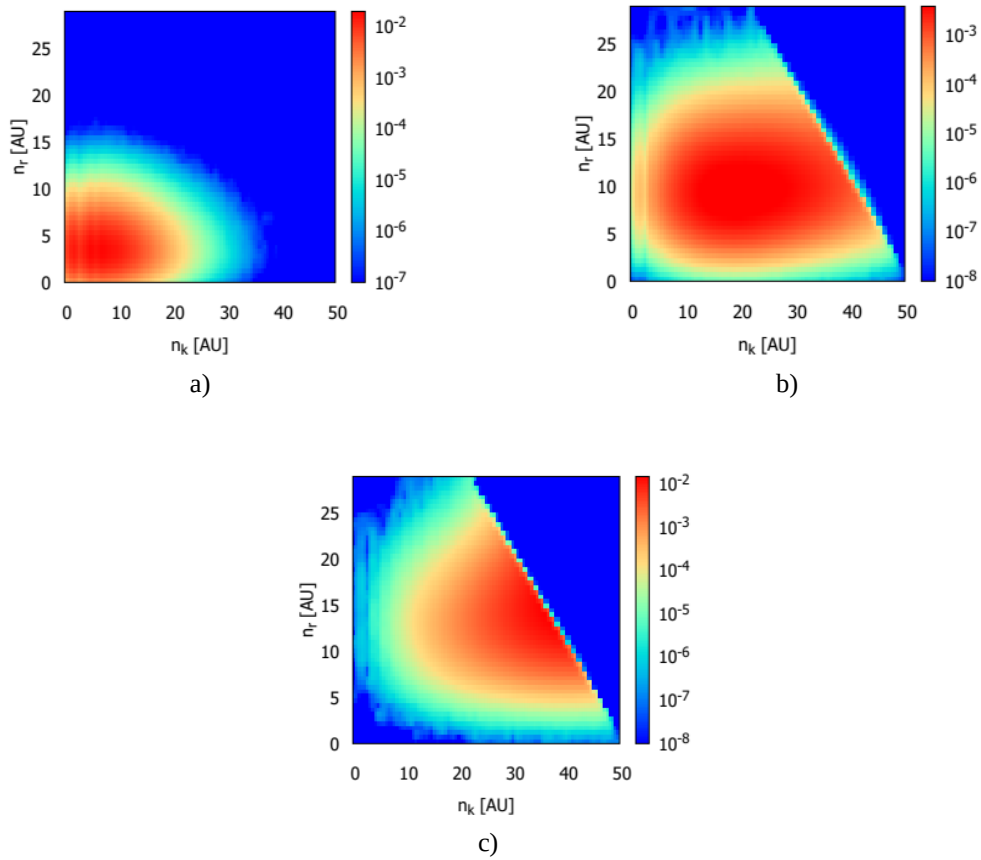


Figure 7. Occupancy distribution as a function of  $n_r$  and  $n_k$  for a)  $a = 0.4$ , b)  $a = 1.0$ , c)  $a = 1.5$  Erl

It is evident that, as the offered traffic  $a$  increases, the probabilities of states described by the number of occupied AUs by hard and soft flows change. It should be noted that when the number of occupied AUs in the entire system exceeds  $C_r$ , a compression mechanism for soft flows is triggered. Compression results in an extension of transmission time for elastic flows so that all scheduled data is transmitted. For adaptive flows, only the amount of transmitted data is limited. Figures 7b and 7c show the boundary between states with non-zero and zero probability. This sharp boundary arises from the fact that this triangular area includes states that cannot occur in the considered scenario.

The developed simulator also allows for the determination of the delay values for flows managed by the system. Figure 8 presents the calculated average delay values for different flows. As observed, non-zero delay values occur only for elastic flows.

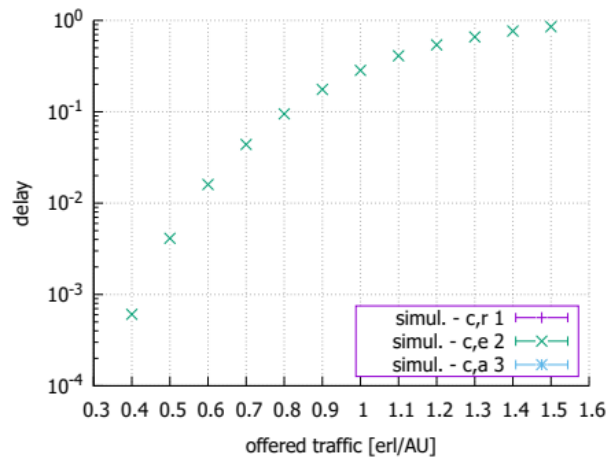


Figure 8. Average delay of flows in the real-time system

Figure 9 depicts changes in the compression coefficient. The red plane represents states (pairs  $n_r$  and  $n_k$ ) - AUs occupied by hard and soft flows but the compression factor is equal to 1. The blue plane represents states that cannot occur in the considered system because  $n_r + n_k$  cannot exceed  $C_r$ . States between these planes represent pairs  $n_r$  and  $n_k$  where elastic flows are subject to AU limitation and extension of service time, while adaptive flows are limited only in terms of available resources.

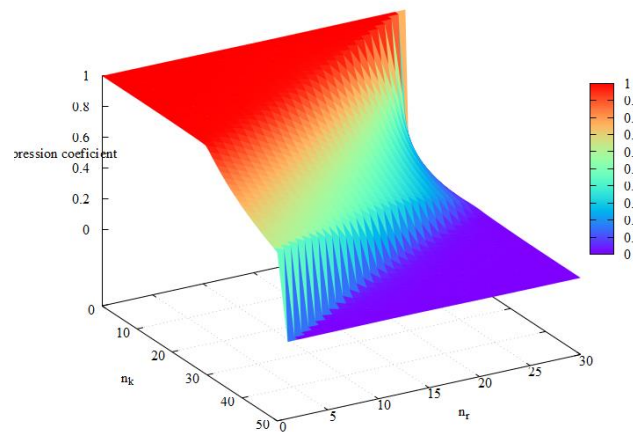


Figure 9. Compression coefficient in considered system

The experiments also included measurements of the time required to complete one simulation series. In the developed simulator, it was assumed that the length of a simulation series would be measured by the number of flow requests requiring the most AUs for processing that appear in the system. It was assumed that the length of a single series is 1,000,000 flows. This definition of simulation length implies that during one series, the system will service a significantly larger number of requests, including other classes of flows. The simulation program was executed on two computers with parameters listed in Table 1. In both cases,

as the offered traffic  $a$  (per individual AU of the system, see Equation (1)) increases, the time required to complete a single simulation series rises. This is due to the increase in the number of flows of different classes appearing in the simulated system. However, with traffic greater than 1.3 Erl, the time increases minimally. This is due to nearly 100 percent resource utilisation of the considered system. It is worth noting that the simulator was written as a single-threaded program, which results in the use of only one processor core, which is almost fully utilised during the simulator's operation. The execution times of a single simulation series were measured using tools available in the chrono library (Microsoft, 2023).

**Table 1.** Execution time of a single simulation series

| a [Erl] | Time to perform one series of simulations [s] |            |
|---------|-----------------------------------------------|------------|
|         | Computer 1                                    | Computer 2 |
| 0.4     | 7.350                                         | 17.134     |
| 0.5     | 8.085                                         | 18.971     |
| 0.6     | 9.345                                         | 21.164     |
| 0.7     | 10.420                                        | 23.656     |
| 0.8     | 11.697                                        | 26.639     |
| 0.9     | 13.141                                        | 29.792     |
| 1.0     | 14.773                                        | 33.072     |
| 1.1     | 16.097                                        | 35.747     |
| 1.2     | 16.956                                        | 37.814     |
| 1.3     | 17.402                                        | 39.153     |
| 1.4     | 17.683                                        | 39.658     |
| 1.5     | 17.772                                        | 39.669     |

Computer 1: processor Intel(R) Core(TM) i7-9700K CPU @ 3.60GHz, RAM 32 GB, 3600 MHz

Computer 2: processor Intel(R) Xeon(R) CPU E5-2699 v3 @ 2.30GHz, RAM 64 GB, 2400 MHz

#### 4. Conclusions

The paper presents the concept of developing a simulator designed to study the QoS parameters of flows transmitted within real-time systems. The simulator differentiates between two types of traffic characteristic of real-time system services, namely traffic associated with hard and soft real-time services. The paper also includes numerical results for a sample system. Since the simulator does not have a graphical interface for result visualization, the Gnuplot program was used. The simulator program, implemented in C++, is user-friendly and, due to its modular design, can be easily extended to accommodate other systems. The implemented set of core functionalities of the simulator, enabling among others the determination of blocking probabilities and delay values, allows for the analysis of real-time systems such as public safety networks (communication systems dedicated to emergency services, including ambulance, fire brigade, and police), medical systems (e.g., remote surgeries), and transport-specific solutions, such as the Future Railway Mobile Communication System (FRMCS). In all of the aforementioned systems, it is crucial to determine the probability of system unavailability (blocking probability), i.e. the ability to reliably transmit data, establish an emergency connection, or perform a remote operation (the high reliability of modern systems means that the issue of packet loss has become of secondary importance). According to traffic engineering principles, blocking in a system depends on the number of requests arriving per unit of time. Consequently, the simulator also makes it possible to determine the operational boundaries within which a given system meets the criteria of a real-time system. It should be emphasised that the presented simulator was used to evaluate the analytical model proposed for the Future Railway Mobile Communication System (FRMCS) (Hanczewski, 2025).

#### Acknowledgements

This research was funded in part by the Polish Ministry of Science and Higher Education under Grant 0313/SBAD/1316.

#### Declaration of Generative AI and AI-assisted technologies in the writing process:

During the preparation of this manuscript the authors did not use Generative AI and AI-assisted technologies and take full responsibility for this declaration.

## References

1. Baital, K. and Chakrabarti, A. (2019) Dynamic scheduling of real-time tasks in heterogeneous multicore systems. *IEEE Embedded Systems Letters*, 11(1), 29-32. DOI: 10.1109/LES.2018.2846666.
2. Banks, J., Carson, J. and Nelson, B. (1996) *Discrete-event system simulation*. Upper Saddle River: Prentice Hall.
3. Bakkali, A., Kobayashi, T., Nishimura, K., Shibagaki, N., Kashima, K. and Sato, Y. (2017) Real-time 10GbE data transmission over a converged RoF links and 96-GHz wireless bridge. *IEEE Photonics Technology Letters*, 29(1), 15-18. DOI: 10.1109/LPT.2016.2624339.
4. Buttazzo, G., Lipari, G., Abeni, L. and Caccamo, M. (2005) *Soft real-time systems: Predictability vs. efficiency*. New York: Springer.
5. Butterfield, A., Ngondi, G.E., and Kerr, A. (2016) *A dictionary of computer science*. Oxford: Oxford University Press.
6. Chen, C., Lu, Z., Xu, Y., Pang, S. and Ma, Y. (2024) Research on real-time video transmission in intelligent vehicle outdoor remote driving system based on 5G network. In: *Proceedings of the 5th International Seminar on Artificial Intelligence, Networking and Information Technology (AINIT)*, Nanjing, March 2024. Springer: Atlantis Highlights in Social Sciences, Education and Humanities, 16-21. DOI:10.2991/978-94-6463-574-4\_70.
7. Eckhardt, D. and Steenkiste, P. (1996) Measurement and analysis of the error characteristics of an in-building wireless network. *ACM SIGCOMM Computer Communication Review*, 26(4), 243–254. DOI: 10.1145/248157.248178.
8. Gnuplot. (2024) *gnuplot homepage*. Available at: <http://www.gnuplot.info/>.
9. Hanczewski, S., Stasiak, M. and Weissenberg, J. (2023) An analytical model of a system with compression and queuing for selected traffic flows. *MDPI Electronics*, 12(17), 3601. DOI:10.3390/electronics12173601.
10. Hanczewski, S., Stasiak, M., Weissenberg, J. and Weissenberg, M. (2025) Modelling of heterogeneous 5G network slice for smart real-time railway communications. *IEEE Transactions on Network and Service Management*, PP(99), 1-1. DOI: 10.1109/TNSM.2025.3547762.
11. ISO/IEC/IEEE. (2017) *Systems and software engineering – Vocabulary*. ISO/IEC/IEEE 24765:2017. Edition 2. Available at: <https://www.iso.org/standard/71952.html>.
12. Li D., Zhang H., Pan G. and Qiu S. (2022) Fault-tolerant scheduling method based on imprecise calculation for real-time system of aviation equipment. In: *Proceedings of the IEEE 6th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, Beijing, October 2022. IEEE, pp. 170-173. DOI: 10.1109/IAEAC54830.2022.9929787.
13. MicroSoft. (2023) *<chrono> - library description*. Available at: <https://learn.microsoft.com/en-en/cpp/standard-library/chrono?view=msvc-170>
14. Stasiak M., Głabowski M., Wiśniewski A., and Zwierzykowski P. (2011) *Modeling and dimensioning of mobile networks*. Hoboken, NJ, USA: Wiley.
15. Sujitha A., Lakshmi L., Mathan N. and Narmadha R. (2023) Analysis of an efficient linear congruential generator architecture for digital applications. In: *Proceedings of the Fifth International Conference on Electrical, Computer and Communication Technologies (ICECCT)*, Erode, February 2023. IEEE, 1-4. DOI: 10.1109/ICECCT56650.2023.10179609.
16. Szalay M., Mátray P. and L. Toka. (2021) Real-time task scheduling in a FaaS cloud. In: *Proceedings of the IEEE 14th International Conference on Cloud Computing (CLOUD)*, Chicago, September 2021. IEEE, 497-507. DOI: 10.1109/CLOUD53861.2021.00065.
17. Tian X. and Benkrid K. (2009) Mersenne twister random number generation on FPGA, CPU and GPU, In: *Proceedings of the 2009 NASA/ESA Conference on Adaptive Hardware and Systems*, San Francisco, July 2009 - August 2009. IEEE, 460-464. DOI: 10.1109/AHS.2009.11.
18. Tyszer J. (1999) *Object-oriented computer simulation of discrete-event systems*. New York: Kluwer Academic Publishers.
19. Zhang T., Huang, R., Hu, Y., Li, Y., Zou, Sh., Zhang, Q., Liu, X., Ruan, Ch. (2023) Load balancing with deadline-driven parallel data transmission in data center networks. *IEEE Internet of Things Journal*, 10(2), 1171-1191. DOI: 10.1109/JIOT.2022.3204908.
20. Zheng J., Qian Y. and Wang C. (2024) Design and cryptographic implementation of an evolutionary digital chaotic model. *IEEE Transactions on Circuits and Systems for Video Technology*, 34(12), Part 1, 12510–1252. DOI: 10.1109/TCSVT.2024.3429641.