Modelling Packet Departure Times Using a Known PDF

Stanislav KLUCIK, Martin LACKOVIC, Erik CHROMY, Ivan BARONAK

Institute of Telecommunications, Faculty of Electrical Engineering and Informational Technology, Slovak University of Technology in Bratislava, Illkovicova 3, 812 19 Bratislava, Slovak Republic

klucik@ut.fei.stuba.sk, lackovic@ut.fei.stuba.sk, chromy@ut.fei.stuba.sk, baronak@ut.fei.stuba.sk

Abstract. *This paper deals with IPTV traffic source modelling and describes a packet generator based on a known probability density function which is measured and formed from a histogram. Histogram based probability density functions destroy an amount of information, because classes used to form the histogram often cover significantly more events than one. In this work, we propose an algorithm to generate far more output states of random variable X than the input probability distribution function is made from. In this generator is assumed that all IPTV packets of the same video stream are the same length. Therefore, only packet times are generated. These times are generated using the measured normalized histogram that is converted to a cumulative distribution function which acts as a finite number of states that can be addressed. To address these states we use an ON/OFF model that is driven by an uniform random number generator in (0, 1). When a state is chosen then the resulting value is equal to a histogram class. To raise the number of possible output states of the random variable X, we propose to use an uniform random number generator that generates numbers within the range of the chosen histogram class. This second uniform random number generator assures that the number of output states is far more larger than the number of histogram classes.*

Keywords

H.264, IPTV, MPEG2 TS, probability density function, traffic generators.

1. Introduction

Traffic generators are mainly used in simulators to test predefined scenarios. They are often used in real network environments for stress testing of paths, devices or QoS (Quality of Service) [1], [2] implementations within tested NGN networks.

This paper deals with an IPTV (Internet Protocol Television) traffic generator that can be applied in above-mentioned scenarios. It describes the proceeding how to derive and construct such a generator of packet times. In our scenario, we developed it as a MatLab function that uses a PDF and desired sequence length in seconds as inputs and inter-generation times are the output. Subsequently we preprogrammed this function to Java that uses a write function to generated packets of identical arbitrary content at the network interface of a network interface card. This generator generates only one IPTV stream. This means no streams are multiplexed together. In more detail, we propose this IPTV generator with following attributes: VBR (Variable Bit-rate), H.264, MPEG2 TS (Moving Pictures Experts Group) (Transport Stream). These properties are the result of examined traffic that we have received from our national IPTV provider. Because our provider uses only CBR (Constant Bit-rate) traffic and we are interested in VBR, only pre coded VBR video records were used. This means that the coding scheme was left to H.264 main profile and Full HD (High Definition) resolution and only one channel per multiplex was examined. Then the recorded video samples were stream streamed out using the following encapsulations: MPEG2 TS/RTP (Real-time Transport Protocol)/UDP (User Datagram Protocol)/IP (Internet Protocol)/Ethernet. With this proceeding, we obtain IPTV VBR traffic with the coding used as by the IPTV provider. Results obtained from this generator are compared to real traffic. Both, real and generated traffic are streamed to our simulator which was created within MatLab environment. Measurements were made at the Ethernet layer.

Traffic generators have to be constructed as simple as possible, but they also have to be precise in comparison with real traffic. Therefore, it is always a decision between the complexity of the generator and the resulting computer load. In this paper, we propose a one-level packet generator that imitates a defined source. It uses

a known PDF to generate only packet times. This approach minimizes the computer CPU and RAM usages.

Within the scope of the project „Support of Center of Excellence for SMART Technologies, Systems and Services II" funded by structural funds of European union we have built a modern IP Multimedia Subsystem [3] lab at the Institute of Telecommunications. In this lab, we use this generator to simulate IPTV traffic from several sources.

The rest of this paper is organized as follows. The second section describes the current state of the art. The third chapter is devoted to the generator itself. The fourth chapter is the conclusion.

2. State of the Art

Packet generators can be divided into several modeling approaches. Many packet generators are based on several mathematical models like semi-Markovian chains [4], wavelets [5], multifractal analyses [6] or combined Markovian models [7]. These models can be used to generate packet sizes, times between generating of packets, absolute times or they can even aim only a specific part of the whole generator. When the real traffic has some specific properties, the traffic model has to take these properties into account, but some of them are omitted. In [8] authors published a survey on different VBR traffic models. Following models were examined: Markov Modulated Gamma (MMG) model, the Discrete Autoregressive (DAR) model, the second order Autoregressive AR(2) model, and a wavelet-based model These models are compared to each other, IPTV and video conference traffic. In [9] authors use frame oriented generator that mix Normal and Log-normal distributions for frame sizes generation. In the case of H.264 video Burr or Gamma distributions fit better frame sizes distributions of real traffic. In our case, we use a recorded PDF, therefore, the resulting PDF of the generator matches the input PDF and there are no needs to examine frame structures.

IPTV VBR traffic of one stream is a time variant self-similar process, which can be described at several levels. In this paper we propose a one level IPTV packet generator, which does not take the self-similarity parameter into account at all. We assume the case where the self-similarity goes to the background because of flow switching. When tenths of IPTV flows are aggregated together and streamed out, the effect of self-similarity of one flow reduces. We studied this behavior using our MatLab simulator where 125 real and 125 generated switched streams were compared against each other. This simulator is a programmed function of a one network node that can simulate some queueing systems, works as a time driven simulator, and its accuracy was proven with analytically expressed Markovian chains or using other methods. Several nodes can create a network of nodes. Real traffic has its own self-similarity because it consisted from 125 separate recorded IPTV streams that were switched together to a one IPTV stream. The same was made with the generated traffic, where the generator we used was based on the Burr distribution function. We compared results from real and generated traffic using the same way as in our previous paper [10]. One-way delay, packet loss and jitter were observed. Results are a bit more correlated like the results in sub-chapter 3.3. Because of this partial success we decided to create a one-level packet generator that uses a known PDF (Probability Density Function) that is not analytically expressed. Proposed algorithm is also usable in many other applications where finite number of states at the input has to be extended to nearly unlimited number of output states without using any analytically expressed function.

3. One Level Packet Generator

This chapter describes procedures to create a packet generator from a measured PDF of times between generation of packets. It is assumed that the all generated packets have the same size. Therefore, only packet times are needed to be generated. The MPEG2 TS layer creates 7×188 bytes long packets that are encapsulated into RTP, UDP, IP and Ethernet layer. Therefore the generated packet is in real an Ethernet frame with the total length of $7 \times 188 + 12 + 8 + 20 + 38 = 1394$ B. This length is usable in our network simulator that works nearly on the L2/L1 RM OSI (Reference Model Open System Interconnection) layer.

3.1. Input – Known Probability Density Function

PDF that we use as an input to our generator was measured in our experimental laboratory where we streamed the traffic obtained from our IPTV provider. Inter-arrival times were observed. Because measuring time using a personal computer is not very precise, we had to filter the results, and the resulting PDF can be seen in Fig. 1. CDF (Cumulative Distribution Function) of this PDF is showed on Fig. 2. To stream out traffic this time we used the VLC player software. Time is a continuous variable. When generating time stamps, real values have limits defined by the variable type used within programming environment. In MatLab, we used the default double-precision floating-point data type. This data type is more precise than the floating-point values needed to represent time stamps.

From filtered results, we created a histogram using defined classes. Then we normalize the absolute counts of occurrences to the sum of occurrences, so the sum of resulting occurrences is equal to one. In this way, we meet the rule that the integral or sum of a PDF over the y axis has to be equal to one, where y axis represents the probability of occurrence of a given class. In using a recorded PDF, we do not need to look for a usable analytical expressible probability distribution function. However, using a PDF that was made from a histogram with limited number of classes decreases the absolute count of states that can be generated using a known recorded PDF.

Creating frame times on an Ethernet line with a defined clock where time is a continuous variable using a histogram with only about thousand of classes may not be enough. Therefore, we propose a method wherewith we do not need to look for an analytically expressible PDF but we can create nearly continuous states of this random variable described by the recorded PDF.

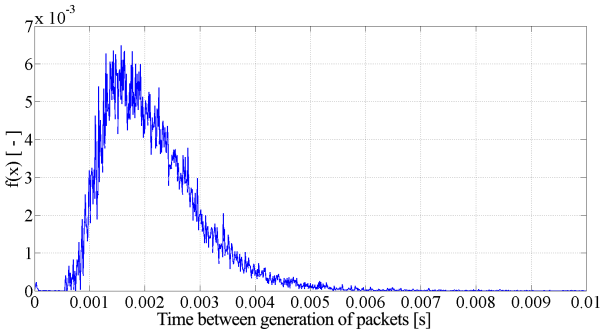

Fig. 1: PDF obtained from measurements on real IPTV traffic.

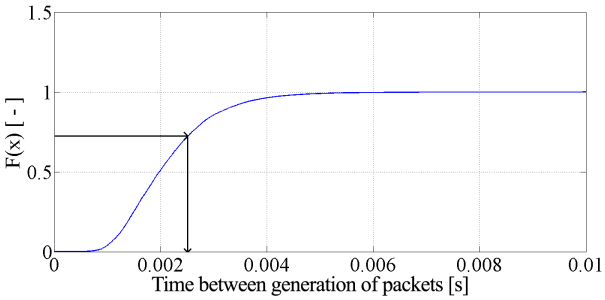

Fig. 2: CDF obtained from measurements on real IPTV traffic.

3.2. Defining the Generator

In this section, the process of finding the appropriate x value from a generated y value is described. When generating numbers according to a known PDF, firstly we create a CDF from this known PDF. Then we use an uniform pseudorandom generator with defined limits between $\langle 0, 1\rangle$ (MatLab rand() function). When the pseudorandom generator generates a value as a double-precision floating-point data type, it represents a real value with a limited number of numbers behind the comma sign. So this function generates a real number that is used to find the best value on the $F(x)$ vector representing the CDF y axis. Then a value from the x axis is assigned to the equivalent y value. This procedure is outlined in Fig. 1 and Fig. 2. As we already mentioned above, the PDF or CDF is a normalized histogram that uses a finite number of classes. If the histogram has 1001 boundaries, 1000 classes are created. Using the process of finding the appropriate class for an uniform random generated value representing a value on the $F(x)$ we only obtain 1000 finite times when a packet or frame is generated. These times are states that the resulting random variable can achieve. So we propose an algorithm to create nearly unlimited number of states distributed according to a recorded PDF.

Table 1 shows a PDF example that is created from a histogram. The first column shows centers of histogram classes, *f(x)* are the function values or probability of class occurrence, and $F(x)$ are CDF values. Centers of classes are used because histograms are often created with uniformly distributed bins. Using this PDF or CDF would result in only 7 states of random variable. We added a new step after the corresponding class was found. When a value with the center of 3.5 is created and the boundaries are from 3 to 4 (only uniform distributed classes are considered) an uniform distributed number is generated with limits $\langle 3, 4\rangle$. This generated number replaces the number representing the found class.

Tab. 1: Example PDF created from a histogram.

Center value of class (x)	f(x)	F(x)
0.5	0.1	0.1
1.5	0	0.1
2.5	0	0.1
3.5	0.5	0.6
4.5	0.2	0.8
5.5	0.1	0.9
6.5	0.1	1.0

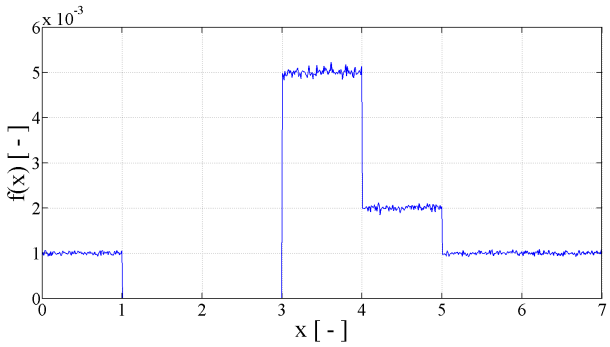

Fig. 3: PDF generated according to Tab. 1.

1) The Algorithm Presets

Input variables to our algorithm are the PDF and time. PDF consists of two columns where the first column contains the classes defined by the center values, and the second column contains probability of occurrences of equivalent classes. Classes are representing times between generations of IPTV packets. Time defines the length of the generated IPTV stream in seconds.

Algorithm is as follows. From the PDF, the mean value of time between the generation of packets is computed and from this value the number of output elements is computed according to Eq. (1):

$$E = \frac{time}{\mu}, \tag{1}$$

where, E is the number of elements to be generated and E also controls the number of cycles for generating elements, *time* is the requested time of the generated stream in seconds and μ represents the mean value of inter-generation time of packets. Then the CDF is computed from the input PDF using a cumulative sum of elements from the second column that contains the probability values. Assuming that classes have the same width, class width is computed according to Eq. (2):

$$I = x(2) - x(1), \tag{2}$$

where I is an interval or class width, *x(1)* and *x(2)* are the centers of two consecutive classes or the centers of the first and second class.

2) Algorithm - Searching for Classes

Generating values uses a simple for cycle with a nested search algorithm to search for best CDF values, which are used to find the class to generate time from. Every step a random number from an uniform interval of $\langle 0, 1\rangle$ is generated. This number represents the y axis of the CDF. Algorithm for searching the best $F(x)$ according to the generated y uses the condition defined in Eq. (3):

$$F(x) < y, \tag{3}$$

where, *F(x)* is a concrete probability value computed from the input PDF and y is the generated random value in $\langle 0, 1\rangle$.

In our generator PDF or CDF values are defined as matrixes. If there is no information about the placement of y values whether they are pointing to the middle or beginning of a corresponding class within the previous used histogram, we have to decide what the meaning of these values is. This information is needed because the generator has to decide what class should be used for elements (packet or time) generation. The condition in Eq. (3) denotes that the concrete *F(x)* value points to the beginning of the current class.

There is no use for finding the nearest neighbor of a corresponding class. Figure 4 shows what will happen when the nearest neighbor would be searched. This figure corresponds to the PDF defined in Tab. 2. The X on the Fig. 4 between the first and second classes represents a generated value of approximately 0.3 which was generated from the uniform interval of $\langle 0, 1\rangle$. If this value X is nearer from the point of view of y axis to the value $F(x) = 0.2$ as to the value $F(x) = 0.4$ then the resulting area would cover 1.5 classes as a limit from left (down), when searing the nearest value would be used. On the other hand, the last class would be used only in 1/2 of generated values that should correspond to that class. Therefore, it would be better to define the PDF and CDF as matrixes that are pointing to the beginning of corresponding classes. Also, the condition in Eq. (1) must be met.

Tab. 2: Example PDF – defining the condition of interest.

Center value of class (x)	Beginning value of class (x)	f(x)	F(x)
0.5	0	0.2	0.2
1.5	1	0.2	0.4
2.5	2	0.2	0.6
3.5	3	0.2	0.8
4.5	4	0.2	1.0

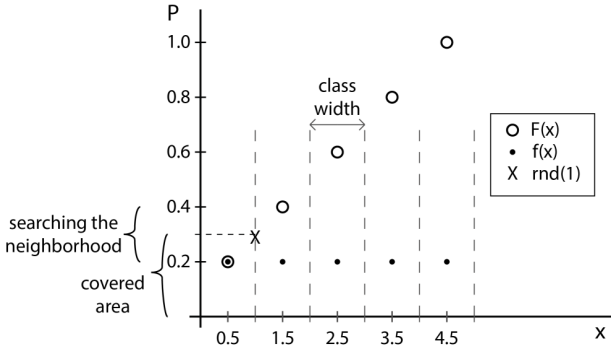

Fig. 4: PDF and CDF from Tab. 2.

Now is the class to generate from known, then the second generation of the current resulting time has to be done. The resulting time is generated between boundaries of two classes and is described in Eq. (4):

$$t = a + (b - a) \cdot rand(1), \tag{4}$$

where, t is the current generated time, a and b are boundaries of the current class and are defined in Eq. (5) and $rand(1)$ generates a random variable in

$\langle 0, 1\rangle$:

$$\begin{aligned} a &= x(j) - I/2, \\ a &= x(j) + I/2, \end{aligned} \tag{5}$$

where *x(j)* is the center of the current class j and *I/2* is the one-half of the computed interval defined in Eq. (2).

Figure 3 shows the resulting PDF after the uniform random number generator was applied to the input values from Tab. 1. Even when the input PDF uses 7 classes the resulting number of states is restricted only to data type used for the variable, which contains the generated uniform random value in limits of the concrete chosen class. Certainly the resulting curve of the PDF contains stairs, but when more classes are used, the shape at all should not be a problem when generating IPTV packets. This process of creating packets includes the need of generating 2 values using the uniform pseudo-number generator and a find function to find the best *F(x)*. Finding the best *F(x)* value depends on the representation of x axis for the *F(x)*.

After generating an appropriate time representing the inter-generation time of packets or frames, a final customization of this generator must be used. MPEG2 TS uses a PCR (Program Clock Reference) that has to be sent every 100 ms. Within this 100 ms interval, the inter-generation time has to be constant. So if a real number is generated, the number of packets that have to be sent out with this time is computed in Eq. (6):

$$NP = \left\lceil \frac{100 \cdot 10^{-3}}{t} \right\rceil, \tag{6}$$

where NP represents the number of packets that have to be sent out with the same time between the generation of packets and t represents the time between the generation of packets itself. When that time is smaller, more packets have to be sent out within the interval between sending of PCR samples. Because of this repetition in times the overall computer load stays low.

This type of packet generator does not take the self-similarity of an IPTV traffic source into account. Because of using a real PDF, the resulting mean values like mean output speed or variance in comparison with the real traffic are nearly the same. Therefore, we do not show any plots supporting this statements. However, when using this generator in a real network, the resulting packet loss or jitter would be smaller. It is because this generator does not consider sending I frames, which have a bigger size in comparison to B or P frames. Both frames have to be sent out with the same time interval. Therefore, this type of generator is more suitable for testing of a large set of IPTV streams that are multiplexed within the network. In using many streams, the resulting variance of streams smoothes-up and there is no such big difference in packet loss and jitter.

3.3. Results

Because we used a recorded PDF, the resulting mean and variance are nearly the same and slightly alter between separate usages of this generator. There is no point to show them. Therefore, higher level results are a better approach how to compare proposed generator with the recorded traffic.

In this tests VLC player was used to stream traffic, therefore, the PDF originates from this streaming. Using VLC brings a new issue into this chapter. VLC player does not meet the rule that every 100 ms is the bit-rate of the video constant, but rather uses its own rules. These rules are showed in Fig. 5.

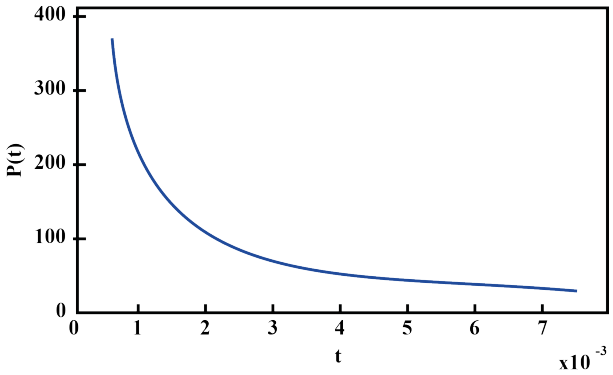

Fig. 5: Number of packets *P(t)* to be generated with the same inter-generation time as a function of the resulting generated inter-generation time *t*.

Normally the number of packets to be generated with the same time between the individual generations of packets would be computed according to Eq. (6). Because we used the VLC player, we had to adapt this step and we used an analytically expressed form of the curve represented in Fig. 5. This curve denotes that the 100 ms interval is not constant and depends on the current inter-generation time. Otherwise, this curve would be a straight line. This is only a problem of the current usage of this streamer, therefore, there is no point to concern about this issue more.

In our tests, no packet losses were observed and therefore we present only the resulting packet jitter within our tested network (Fig. 6). The results confirm the assumption that the generator produces a bit small jitter then the original traffic. As already mentioned, this is because the aggregated traffic that consists of 100 IPTV streams has no dependence to the self similarity of a real IPTV stream. However, aggregating traffic reduces this effect quite visible.

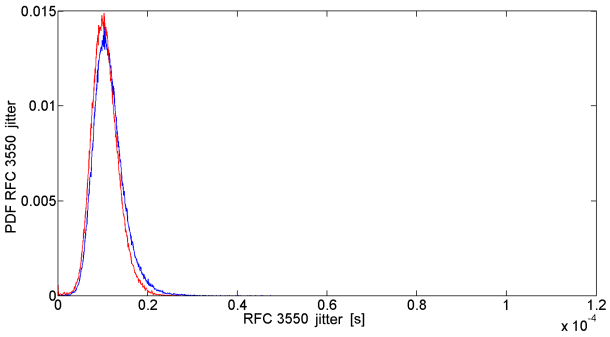

Fig. 6: RFC 3550 jitter in test network of recorded traffic (blue) and generated traffic (red).

This type of generator is very little computing intensive, because only two number generation per packet has to be done and a one searching function is used. However, using MPEG2 TS ensures that this generation is called approximately every $N \times 100^{th}$ packets depending on the input PDF. This generator is also slower in comparison to a generator that uses an analytically expressed function. However, in our case, the recorded PDF is desired as an input.

4. Conclusion

In this paper, we propose a one level packet generator used to generate IPTV packet times between generation of packets or inter-arrival times. This generator uses a known recorded PDF to generate packets from. The input PDF to the proposed algorithm is expressed as a matrix and has its origins in a histogram that uses classes within defined ranges. PDF with uniform classes reduces the number of usable states to generate from. Therefore, we propose an algorithm to generate nearly continuous states that are representing time. Resulting IPTV generator does not meet the rule of IPTV stream self-similarity and therefore it is usable as a generator for aggregating streams to higher speeds so the self-similarity does not apply so much. As the results show, the resulting generator is less compute intensive then our four-level IPTV generator and can delivers nearly similar results, but only when multiplexing is used.

Proposed algorithm is also applicable in other generators that have to use a recorded PDF with defined classes because the PDF originates from a histogram.

In further work, we intend to create an adjustable IPTV H.264 packet generator with the ability to choose the desired output bitrate of the IPTV stream. This generator will use a known recorded PDF and apply in this article introduced model to raise the number of output states of the generator.

Acknowledgment

This article was created with the support of the Ministry of Education, Science, Research and Sport of the Slovak Republic within the Research and Development Operational Programme for the project "University Science Park of STU Bratislava", ITMS 26240220084, co-funded by the European Regional Development Fund. This article is a part of research activities conducted at Slovak University of Technology Bratislava, Faculty of Electrical Engineering and Information Technology, Institute of Telecommunications, within the scope of the projects "Grant programme to support young researchers of STU - QoS in IMS networks".

References

[1] REZAC, F., M. VOZNAK and F. HROMEK. Delay Variation Model with Two Service Queues. *Advances in Electrical and Electronic Engineering.* 2010, vol. 8, no. 1, pp. 24–28. ISSN 1804-3119. DOI: 10.15598/aeee.v8i1.15.

[2] ROZHON, J., F. REZAC, J. SLACHTA and M. VOZNAK. Monitoring of Speech Quality in Full-Mesh Networks. In: *21st International Conference, CN 2014.* Brunow: Springer, 2014, pp 157–166. ISBN 978-3-319-07940-0. DOI: 10.1007/978-3-319-07941-7_16.

[3] JANEVSKI, T. *NGN architectures, protocols, and services.* United Kingdom: Wiley-Blackwell, 2014. ISBN 978-1-118-60720-6.

[4] KEMPKEN, S. and W. LUTHER. Modeling of H.264 high definition video traffic using discrete-time semi-markov processes. In: *Proceedings of the 20th International Teletraffic Congress, ITC20 2007.* Ottawa: Springer, 2007, pp. 42-–53. ISBN 978-3-540-72989-1. DOI: 10.1007/978-3-540-72990-7_8.

[5] DAI, M., Y. ZHANG and D. LOGUINOV. A Unified Traffic Model for MPEG-4 and H.264 Video Traces. *IEEE Transactions on Multimedia.* 2009, vol. 11, iss. 5, pp. 1010–1023. ISSN 1520-9210. DOI: 10.1109/TMM.2009.2021802.

[6] RELJIN, I., A. SAMCOVIC and B. RELJIN. H.264/AVC video compressed traces: multifractal and fractal analysis. *EURASIP Journal on Advances in Signal Processing.* 2006, vol. 2006, pp. 1–13. ISSN 1687-6180. DOI: 10.1155/ASP/2006/75217.

[7] WAN, F., L. CAI and T. A. GULLIVER. A Simple, Two-Level Markovian Traffic Model for

IPTV Video Sources. In: *IEEE GLOBECOM 2008 - 2008 IEEE Global Telecommunications Conference*. New Orleans: IEEE, 2008, pp. 1–5. ISBN 978-1-4244-2324-8. DOI: 10.1109/GLOCOM.2008.ECP.288.

[8] TANWIR, S. and H. PERROS. A Survey of VBR Video Traffic Models. *IEEE Communications Surveys & Tutorials*. 2013, vol. 15, iss. 4. pp. 1778–1802. ISSN 1553-877X. DOI: 10.1109/SURV.2013.010413.00071.

[9] REAZ, A., D. MURAYAMA, K.-I. SUZUKI, N. YOSHIMOTO, G. KRAMER and B. MUKHERJEE. Synthetic traffic generation for streaming video to model IPTV. In: *2011 Fifth IEEE International Conference on Advanced Telecommunication Systems and Networks (ANTS)*. Bangalore: IEEE, 2011, pp. 1–6. ISBN 978-1-4673-0093-3. DOI: 10.1109/ANTS.2011.6163638.

[10] KLUCIK, S. and M. LACKOVIC. Modelling of H.264 MPEG2 TS Traffic Source. *Advances in Electrical and Electronic Engineering*. 2013, vol. 11, no. 5, pp. 404-409. ISSN 1804-3119. DOI: 10.15598/aeee.v11i5.870.

About Authors

Stanislav KLUCIK was born in Bratislava, Slovakia, in 1984. He received the Master degree in telecommunications in 2009 from Faculty of Electrical Engineering and Information Technology of Slovak University of Technology in Bratislava (FEI STU). In 2012 he submitted a Ph.D. work from the field of Source and QoS traffic parameters modeling in NGN networks and his scientific research is focused on packet generators and modeling of packet networks. Nowadays he works as research at the Institute of Telecommunications of Faculty of Electrical Engineering and Information Technology of Slovak University of Technology in Bratislava.

Martin LACKOVIC was born in Trnava, Slovakia, in 1989. He received the Master degree in telecommunications in 2013 from Faculty of Electrical Engineering and Information Technology of Slovak University of Technology (FEI STU) Bratislava. His scientific research is focused on modeling of packet generators, traffic parameters and networks.

Erik CHROMY was born in Velky Krtis, Slovakia, in 1981. He received the Master degree in telecommunications in 2005 from Faculty of Electrical Engineering and Information Technology of Slovak University of Technology (FEI STU) Bratislava. In 2007 he submitted Ph.D. work. Nowadays he works as assistant professor at the Institute of Telecommunications of Faculty of Electrical Engineering and Information Technology of Slovak University of Technology in Bratislava.

Ivan BARONAK was born in Zilina, Slovakia, on July 1955. He received the electronic engineering degree from Slovak Technical University Bratislava in 1980. Since 1981 he has been a lecturer at Department of Telecommunications Slovak University of Technology in Bratislava. Nowadays he works as a professor at Department of Telecommunications of Faculty of Electrical Engineering and Information Technology of Slovak University of Technology in Bratislava. Scientifically, professionally and pedagogically he focuses on problems of digital switching systems, ATM, Telecommunication management (TMN), NGN, IMS, VoIP, QoS, problem of optimal modeling of private telecommunication networks and services.