SOFTWARE SIMULATOR OF AN ATM STATISTICAL MULTIPLEXER WITH VARIABLE PARAMETERS

Virgil Dobrota, Daniel Zinca, Mihai Cosma

Technical University of Cluj-Napoca, Department of Communications
26 Baritiu Street, 3400 Cluj-Napoca, Romania
Tel: +40-64-191689, +40-64-197083, Fax: +40-64-194831
E-mail: {dobrota, zinca, mcosma} @ utcluj.ro

Abstract

This paper presents a software simulator of an ATM statistical multiplexer designed to evaluate the performances of a broadband communication systems at a non-real time rates from 1 Mbps up to 1000 Mbps. The main objective of the proposed Windows simulator is to find the optimum design of the four queueing system multiplexer with priorities scheme using two or three levels.

1. Introduction

The high-speed networks implementation and the evaluation of the performances imply to setup very expensive trials with equipment which is not always available. Therefore, the simulating process is becoming an important step. This paper presents a software simulator of an ATM statistical multiplexer, designed for broadband communication systems (Windows 3.1x environment), at a non-real time rates from 1 Mbps up to 1000 Mbps. The inputs could provide a continuous flow of ATM cells or could have a uniform, Poisson or Bernoulli distribution. The data are provided by the files for each input and the results (meaning the output of the multiplexer) are stored into proper files, including the demultiplexing process too. The experiments were focused on evaluating the influence of variable parameters over CLR *(Cell Loss Ratio)*, calculated for each input and also globally.

Almost similar approach has been presented at ICSE'96 in Las Vegas [1], but the operating system was Linux and the real-time rates were envisaged. Obviously the output rate of the ATM multiplexer in this case was not higher than 100 Mbps, so the results of both simulators could be compared within the common range of throughputs only.

2. Software simulator of an ATM statistical multiplexer

Fig. 1. presents the functions of the project application. Each Windows-based program has a *WinMain* function as an enter point, having the following parameters: two variables (descriptors) for the current instant and the previous instant, a pointer to a string of characters for sending arguments to the program and an integer to indicate the window display type (minimized/maximized) when the application starts. This function initializes and registers the class of the window which will become available for all the instants of the application.

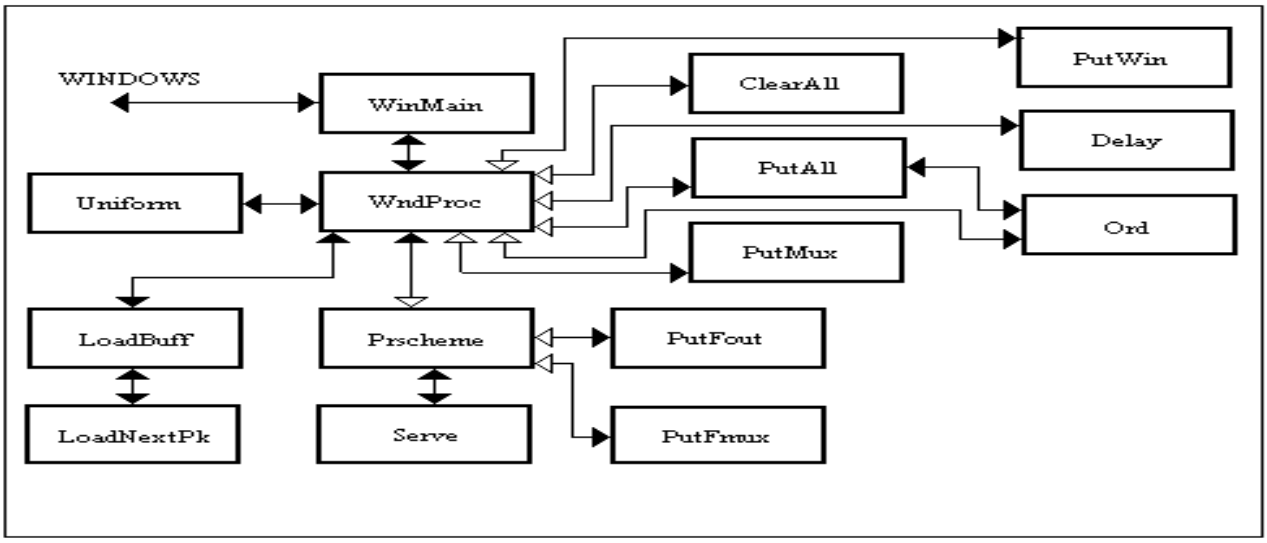

Fig. 1. Diagram block of the software simulator for ATM statistical multiplexer

The *CreateWindow*, *ShowWindow* and *UpdateWindow* functions will be used by the current instant in order to show the application's window. The external communication is realized through a queue of messages according to the external events (pressing a key or a mouse button). Another function, very important one, is designed for the application's window, particularly called herein *WndProc*, which does not necessarily handle any message addressed to it. At the *CreateWindow* call, the *WndProc* function receives the *WM_CREATE* message and initiates the global variables. For a popup menu-oriented execution of an application, the previous mentioned function receives *WM_COMMAND* message and an *IDM_<command>* identifier (defined within the program or within its header file). The messages which are not handled by the *WndProc* function are forwarded to *DefWindowProc*. The double-line frame of the window is drawn by *PutWin*, whilst the number of digits used for the representation of arguments as integers is provided by *Ord* function. The uniform input distribution for the queueing systems needs the *Uniform* function which returns an integer value of random number of waiting periods/cell until the next cell will arrive. A Boolean function called *LoadBuff* updates the global variable *Buffer* as

follows: the first argument is the index of the data input and the second one is the number of ATM cells arrived in the system. It returns either *TRUE* (if an end-of-file has been found) or *FALSE*. The function *LoadNextPk* reads an ATM cell (53 bytes, including the header) from the input files, according to its argument which is an index. It returns *FALSE* if an end-of-file has been found otherwise the value is *TRUE*. The following information are displayed using *ClearAll*, *PutAll*, *Delay*, *PutMux* functions: the buffer size, the threshold values, the input rates and distribution, the number of ATM cells (total and loss), the contents of the first cell in each buffer, the statistics of the queue length. There are two priority schemes that have been implemented: *PrScheme0* supposes that all the inputs have a low level of priority (1), being served according to a circulate token ring mechanism. If the number of cells in a queue is becoming greater than a given threshold L the buffer receives the higher priority level (0), and the token will not be shared with low priority inputs. As the congestion traffic cannot be avoided, a study of three level of priorities have been done using *PrScheme1*. The conclusion is that the higher the number of levels in the priority schemes is, the more severe are the constraints for a real-time traffic. Both mentioned functions call the *Serve* function to shift the buffer and the *PutFmux* writes the received cells in an output file as a proof of the multiplexer behaviour until the input flow is active. Finally the *PutOut* function acts as demultiplexer.

3. Simulation of random arrivals

One the most difficult part of the proposed implementation was the simulation of the random arrival of ATM cells (*L = 53x8 = 424 bits/cell*). For a better understanding of the principle adopted let suppose a uniform distribution for the *Uniform* procedure previously mentioned. According to the rate-setting condition for probability *P(CR)* and the definitions of *PCR (Peak Cell Rate)* and *PBR* (*Peak Bit Rate)*, presented in [2]:

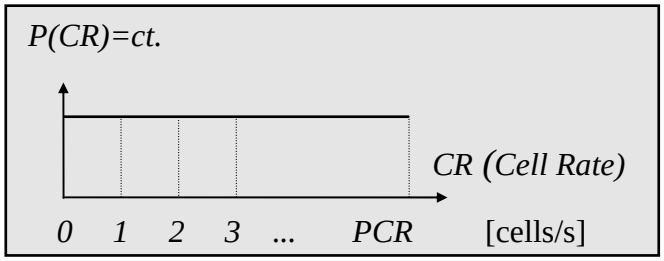

Fig.2. Uniform distribution

$$PCR = \frac{PBR}{L} \quad \text{[cells/s]} \qquad (1)$$

$$\sum_{CR=0}^{PCR} P(CR) = 1 \qquad (2)$$

$$P(CR) = \frac{1}{PCR} = ct. \qquad (3)$$

The average number of cells generated by a simulated source measured over a long interval of time divided by the minimum time between the arrival of two consecutive cells is called *AvCR (Average Cell Rate).*

$$AvCR = \sum_{CR=0}^{PCR} [CR \times P(CR)] = \frac{1}{PCR} \sum_{CR=0}^{PCR} CR = \frac{1}{PCR} \times \frac{PCR(PCR+1)}{2} = \frac{PCR+1}{2} \quad (4)$$

and

$$PCR = 2 \times AvCR - 1 \quad (5)$$

The problem is to find the random number *CR* for the given range *[0,PCR]*. Due to the fact that the C function *int random(int)* could return (in a specific situation) integer values far from the limit *PCR*, a new algorithm had to be proposed. Let *CR* be the number to be generated and *N1=random(1000)* a random value within *0...1000*. Let *N2=random(PCR/1000)* be a random value between 0 and the integer part of ratio *CR/1000* (two bytes).

$$CR = N2 \times 1000 + N1, \text{ for } P(CR) = ct., CR \in [0, PCR] \quad (6)$$

The ratio of the peak-to-average traffic generation rate is called β (burstiness) and the ratio *PCR/CR* represents the number of periods until the next cell arrives. Suppose the simulation has the input rate *IR[i]* as a parameter for input *i*, measured in [10^3cells/s]. The value returned by the function *Uniform* is the following:

$$\frac{AvCR}{IR[i]} \times \frac{PCR}{CR} \quad (7)$$

Conclusions:

1. If *AvCR/IR[i]<1* and because *CR* is a random value, it is possible to get a returned value *<1*. This means that the time until the next cell arrives is less than the cell period, or that more than one ATM cell should arrive in a period, which is not possible. In this case it is decided that the time is equal to the cell period (continuous flow or $\beta = 1$).
2. Due to the previous approximation the average input rate could be less than IR[i]. For these reasons the proposed algorithm should be used within the defined range [1,1000] Mbps only.
3. Selecting other type of distribution *P(CR)* the algorithm could be used too.

4. Experimental results

The experiments have been done for different types of distributions: continuous flow of a given rate, random arrival (uniform, Poisson, Bernoulli). The aim was to determine the influence over the CLR for each input and over the global CLR of the following parameters: the variable input/output rates, the average rate of arrivals (for random case) and the different priority schemes. As the simulator was designed for general purposes, an ATM cell is sometimes referred as a packet (of 53 bytes), without a standard header (the routing aspects were not envisaged). Anyway each of the first five bytes identify the origin of the cell, that is the number of the input line. The contents of the input files represent an ASCII text (as a test) which is splitted in 48 bytes parts, having as a prefix the previous mentioned type of header. This procedure is useful for a visual evaluation of the accuracy of multiplexing and demultiplexing processes (see *Fig.4.*). Obviously an objective evaluation is possible, as it is presented in *Fig.5.*

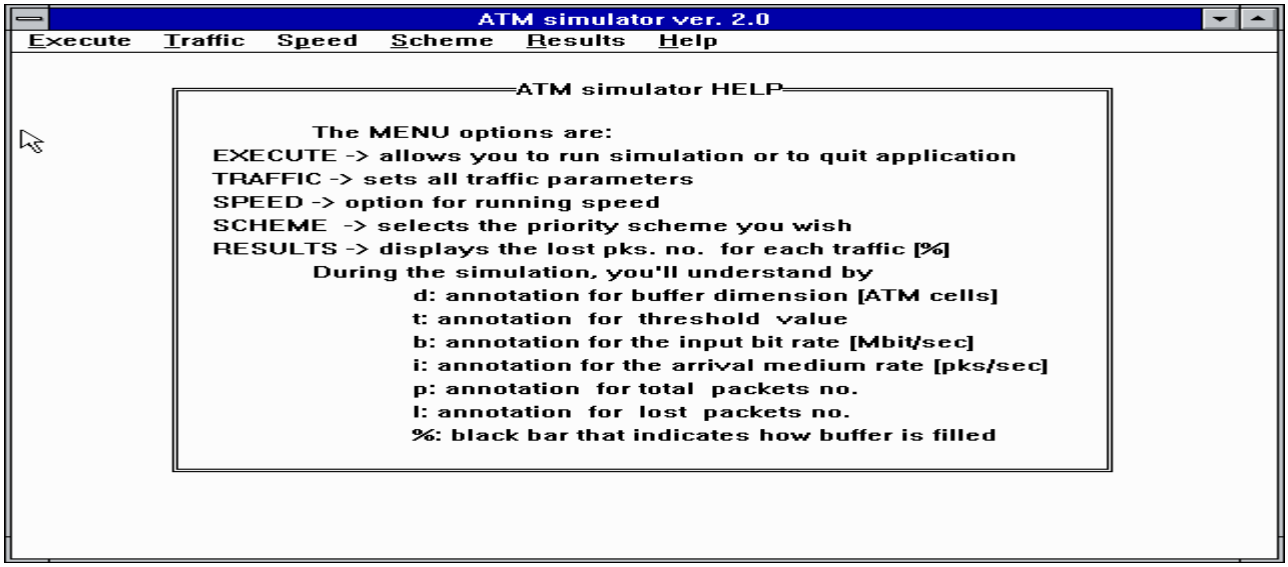

Fig.3. ATM Multiplexer Simulator: the menu options

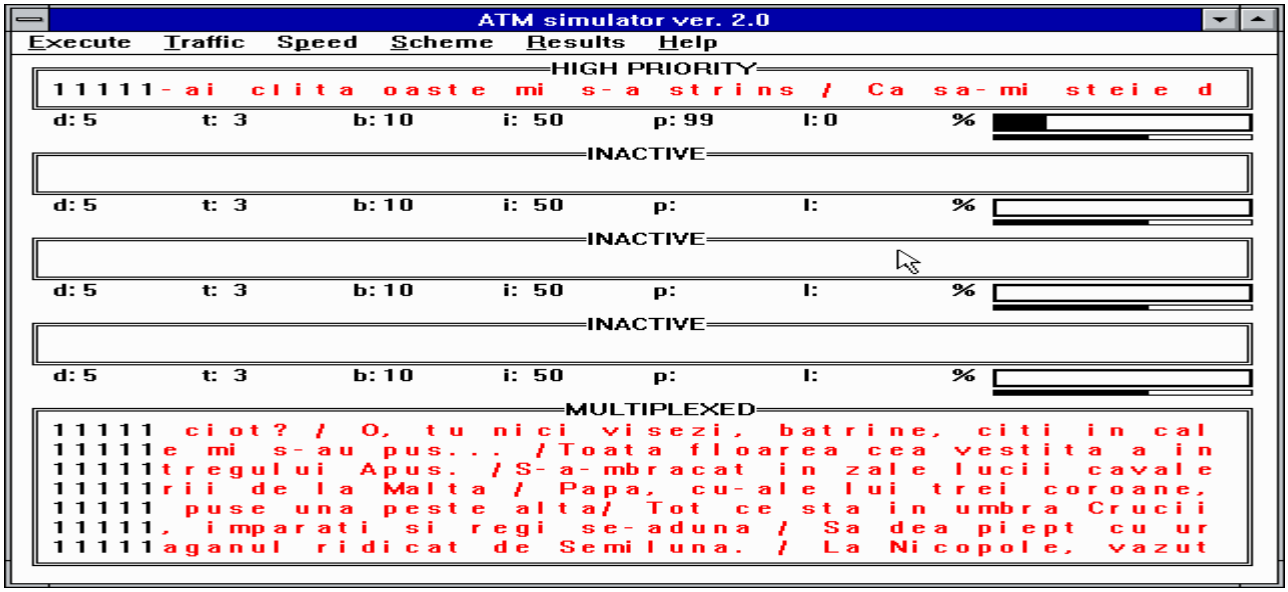

Fig.4. Experimental results (demo non-real time)

	OutBR=25Mbit/s										OutBR=100Mbit/s										OutBR=175Mbit/s				
BR#4 [Mbit/s]	#1 [%]		#2 [%]		#3 [%]		#4 [%]		G [%]		#1 [%]		#2 [%]		#3 [%]		#4 [%]		G [%]		#1 [%]	#2 [%]	#3 [%]	#4 [%]	G [%]
10	13	18	91	91	75	73	75	75	71		0	0	52	52	0	0	0	0	13		0	0	0	0	0
20	13	18	91	91	75	73	87	87	73		0	0	56	56	0	0	32	32	18		0	0	0	0	0
30	13	18	91	91	75	73	91	91	75		0	0	57	57	0	0	53	53	24		0	0	0	0	0
40	13	18	91	91	75	73	94	94	77		0	0	56	56	0	0	64	64	29		0	0	0	0	0
50	13	18	91	91	75	73	95	95	78		0	0	56	56	0	0	72	72	33		0	0	0	0	0
60	13	18	91	91	75	73	95	95	79		0	0	56	56	0	0	76	76	37		0	0	0	0	0
70	13	18	91	91	75	73	96	96	80		0	0	56	56	0	0	79	79	40		0	0	0	0	0
80	11	16	90	90	74	73	96	96	81		0	0	55	55	0	0	82	82	43		0	0	0	16	6
90	10	16	89	89	74	72	96	96	81		0	0	53	53	0	0	83	83	45		0	0	0	25	11
100	8	15	87	87	72	71	96	96	81		0	0	52	52	0	0	84	84	47		0	0	0	33	15
110	8	15	87	87	72	71	96	96	81		0	0	50	50	0	0	84	84	49		0	0	0	38	19
120	7	14	86	86	71	69	96	96	81		0	0	50	50	0	0	85	85	51		0	0	0	42	21
130	5	13	84	84	71	68	96	96	82		0	0	47	47	0	0	86	86	52		0	0	0	47	25
	S0	S1	S0	S1	S0	S1	S0	S1	S*																

Buffer size= 10, Threshold L =5, Input rates: #1=16 Mbps (priority),#2=32 Mbps (priority)
OutBR=Output Bit Rate #3=64 Mbps, #4=variable (no priority)

Fig. 5. Example of experimental results

Conclusions

This paper intended to present the principles for building a software simulator, running in Windows environment and suitable for evaluation of performances of an ATM statistical multiplexer. The parameters that could determine an optimum design (meaning a minimum CLR) are the following: the buffer size, the threshold, the priority scheme, the input/output rates, the type of traffic (continuous or random arrivals).

Acknowledgment

The authors would like to acknowledge the contribution to the initial version of the simulator of M. Emilio and B. Suciu from the Technical University of Cluj-Napoca. In part the work was supported by COPERNICUS #1529 project "Multimedia services over high-speed networks".

References

[1] DOBROTA, V., ZINCA, D., COSMA, M., VLAICU, A. - *Simulator for a Broadband Communication System Based on ATM Statistical Multiplexer/Switch with Fast Ethernet Link.* Eleventh International Conference on Systems Engineering ICSE'96, University of Nevada, Las Vegas, USA, July 9-11, 1996.

[2] DOBROTA, V., CRISTUREAN, F., COSMA, M., ZINCA, D. - *Software Evaluation Tool for ATM Traffic Parameters.* IEEE Symposium of Electronics and Telecommunications Etc'96, Timisoara, Sept. 26-27, 1996.

[3] DOBROTA, V. - *Digital networks in telecommunications. Volume I: digital switching, traffic analysis, ISDN* (Romanian language). Risoprint Publishers, Cluj-Napoca 1996.

[4] LEE, D-S., SENGUPTA, B. - *Queueing Analysis of a Threshold Based Priority Scheme for ATM Networks.* IEEE/ACM Transactions on Networking, Vol.1, No.6, December 1993, pp.709-717.

[5] MOLDOVEANU, F., HERA, G. - *Programming of Windows applications* (Romanian language). Teora Publishers, Bucharest, 1995