Paper

# Multi Queue Approach for Network Services Implemented for Multi Core CPUs

Marcin Hasse, Krzysztof Nowicki, and Józef Woźniak

*Gdańsk University of Technology, Gdańsk, Poland*

**Abstract—Multiple core processors have already became the dominant design for general purpose CPUs. Incarnations of this technology are present in solutions dedicated to such areas like computer graphics, signal processing and also computer networking. Since the key functionality of network core components is fast package servicing, multicore technology, due to multi tasking ability, seems useful to support packet processing. Dedicated network processors characterize very good performance but at the same time high cost. General purpose CPUs achieve incredible performance, thanks to task distribution along several available cores and relatively low cost. The idea, analyzed in this paper, is to use general purpose CPU to provide network core functionality. For this purpose parameterized system model has been created, which represents general core networking needs. This model analyze system parameters influence on system performance.**

***Keywords—generic purpose CPU, multi core, network, queue, network services.***

## 1. Introduction

In the recent years, CPU technology has turned into multi core to brake the clock barrier and improve applications performance. Higher solutions performance require much faster medium to transfer data. Computer networking remains not only a medium, but also network software stack, which is a significant part of the network performance. Once application efficiency is improved by using CPU technology, networking stack software could also be considered as area, were multi core can increase productivity. There are many hardware CPU architectures used for high speed network packet processing. Network processors Intel IXP28XX series [1] introduce hardware micro engines able to process pipeline oriented traffic with significant performance improvement. More recent design from Cavium Networks called OCTEON Multi-Core Processor Family [2] or Freescale [3] are providing hardware driven opportunities to divide multi flow traffic into separated cores. These sophisticated hardware designs characterize high cost of implementation and deployment. Cost of these high performance solutions are significant but still worth their price to satisfy network needs.

From the software point of view there are multiple implementations of network stacks, protocols and solutions provided by the market today. These solutions characterize high performance, modular architecture and relatively easy integration. Due to this they already have strong market position for dedicated network solutions. There are also lots of research going on to optimize hardware support for multiple software threads [4] and to provide possible highest performance for computer network nodes. In personal computers segment multi core CPUs have also strong presence. More and more end user applications taking advantage of this solution and increasing their performance by adding software support for multi core hardware architecture.

In this paper, multi queue approach to network services implemented in multi core general usage CPUs [5] is presented. This approach has been proposed to verify, if this is reasonable to optimize network performance for solutions with standard CPUs. Most of new software architectures, which support multi core environments, are dedicated to data processing but not to the packet processing. It important to identify how much software architecture design can influence on network application performance.

## 2. Multi Core Architecture Approach

Nehalem is the codename for an Intel processor microarchitecture [6], [7]. The most popular available in end user market is Intel's Core i7 [8] processor. Higher performance is achieved on this CPU by processing multiple data in the same time using parallel cores. Most of operating systems are providing multi core support and assigning tasks to be processed by hardware with possible highest performance. An operating system is not focusing on the type of application beside I/O bound or CPU bound types distinguished by the task scheduler [9]. This approach might not be enough to support high performance networking and low performance management and monitoring activities, running at the same time. In the server area this problem has been solved by introducing visualization on both application and platform levels [10], [11]. Many visualization solutions including hypervisor [12], became standard parts of the operating systems [13] providing opportunities to work with several contests at the same time. The software architecture described in this paper also account visualization approach in order to work with different contests on multiple CPU cores. Such solutions like that are available now - for instance Sun xVM Virtual Box [14], which allows multiple operating systems run the same time on a single PC.

Network applications are responsible for servicing data streams. Network streams are transporting significant amount of data, which needs to be processed and depending on application functionality either consumed (provided to the end user) or transmitted (back to the network). Effective stream processing could have significant impact on networking application performance.

For this research purposes, there has been defined a network stream consisted of a limited sequence of data packets $D^1, D^2, \ldots$. To simplify the mathematical model we assume, that each packet represents different amount of data but has the same, permanent defined size. Such approach is often used to specify application throughput for defined packet size

$$D_k = \{D_k^1, D_k^2, ..., D_k^n\}, n \in N. \tag{1}$$

The network application can get streams from several different sources

$$S = \sum_{k=1}^{K} D_k, k \in N. \tag{2}$$

Each stream, responsible for delivery of potentially different sets of information, is directed to different applications for different sorts of processing. This diversity can be a good approach for the system design. In our model we additionally accept, that some streams require more privilege (priority) service and task delivery to the end user.

Multi core CPUs, ensure better performance by redirecting tasks to be executed in the same time on multiple cores. Each core offers the same execution condition including access to peripherals, memory etc. [15]. Properly configured operating system is able to manage tasks to assure possible best performance. In this paper authors propose to change the task approach for multi core CPUs provided by most modern operating system to the network stream approach. The intention on this research is to verify, whether it is reasonable enough to consider general purpose CPU as hardware platform to systems dedicated for networking.

Standard operating system (like, e.g., Linux kernel 2.6.X) in general treats I/O coming from networking card as interrupts coming from any other computer peripherals. Analyzing different communication devices from the data rate per second point of view one can differ between devices (PCI bus 528 MB and gigabit Ethernet 128 MB), however communication expectation is to service significant amount of data in shorter possible time. The network card driver in the Linux kernel associated with networking stack, is able to provide packet delivery services to dedicated application. In cases of many different streams, which need to be serviced by different applications existing in OS together with another, not strict network related tasks, can meet computer performance critical point. When critical point is crossed, it could manifest long tasks queues and delays in service (see Fig. 1).

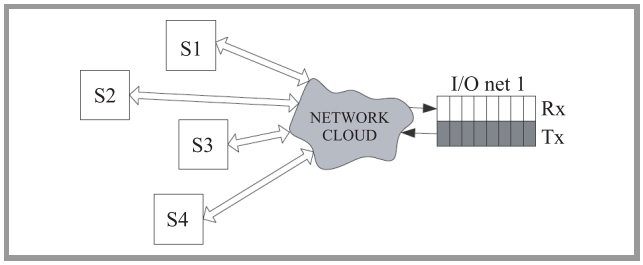


*Fig. 1.* Single package queue for single physical interface.

Considered system approximation assumes single package queue associated with physical interface. Packages in the queue are arriving from a multiple resources. Each data stream $D_k$ in queue income, specified by amount of data $S$, should be serviced by different application. Performance of networking system from this approximation can be determine by specifying time $t$ in which dedicated networking application completed service of $S$

$$P_s = S(t). \tag{3}$$

In the recent days networking technology is based either on networking pipeline [1] or dedicated multi core solutions [2]. These solution offers different approaches for networking applications than standard applications service on generic purpose CPU. In the networking area application can be divided into the following areas:

- Management/Controlling – responsible for controlling network settings, managing network events; type of traffic – control plane able to work with full Linux stack;
- Preprocessing – responsible for redirecting packets to exceptions or forwarding; type of traffic – slow path/fast path able to work with limited L3 stack to classify packages;
- Exceptions – responsible for servicing packages, dedicated to particular node – type of traffic – slow path able to work with full Linux stack;
- Forwarding – responsible for fast forwarding packages to additional network segments; type of traffic – fast path able to work with basic/limited Linux stack.

Each of these areas has different requirements for throughput and different user expectations related to accessibility, stability and manage ability. A general purpose operating system is not providing a different set of system resources for networking application areas, beside the ability to configure different application with different priorities [9]. For example performance of Linux application depends on number tasks executed on CPU in specified amount of time, and scheduler latency can cause unexpected delays for application, which require quick system reaction time.

Multicore CPUs, when hypervisor is used, provides ability to distinguish different system expectation and execute each networking application area on separated core [16]. In this model physical network interface can be assigned directly to OS running with fast and limited Linux stack responsible for preprocessing. This networking application role is to

classify packet and as soon as possible send it to exception or forwarding applications running on additional cores and with dedicated networking stacks. Last core in the system is responsible for dealing with preprocessing, exceptions and forwarding rules. Its management and control interface should be easy available for system administrator though slow port management interface (Fig. 2).

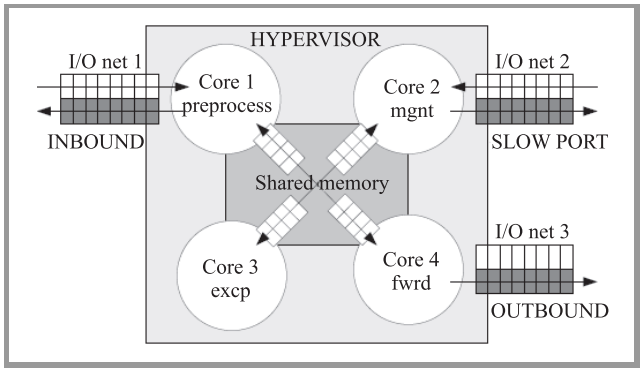


***Fig. 2.*** Network system architecture for general purpose CPU.

Hypervisor role is to physically assign networking interfaces to dedicated cores. Communication between cores is delivered trough shared memory available from each core. This approach allows to install on each core [17] dedicated version of OS (with dedicated networking stack) where physical access to I/Os is pre configured by hypervisor. Interface access latency should be much lower than for solutions with master OS [13], [14].

## 3. Common Network Traffic Models

Generic approach for traffic modeling is to mathematically describe the physical arrival of packets as a point function of countable values. Points describe packets arrival instances starting from $T_0 = 0$ and is limited only to model assumed time frame: $T_0, T_1, \ldots, T_n, \ldots$. Countable values are usually dependent on two point stochastic processes – one, describing time distance between arrival instances and second, describing actual value/number of delivered packets. Packet arrival time (PAT) $\{PAT_n\}_{n=1}^{\infty}$ process is non negative random sequence:

$$PAT_n = T_n - T_{n-1}\,. \tag{4}$$

The point process is describing each instance of packet arrival:

$$T_n = \sum_{k=1}^{n} PAT_k\,. \tag{5}$$

Number of delivered packages (NDP) $\{NDP_n\}_{n=1}^{\infty}$ is associated with amount of work – workload (WLD), which has to be done to service traffic in a specified time. It can be limited by bandwidth bottleneck and can be described via stochastic function $D_n$

$$NDP_n = \begin{cases} D_n(T_n, \tau) & \text{if } \tau <= \tau_{max} \\ D_n(T_n, \tau_{max}) & \text{otherwise} \end{cases}\,. \tag{6}$$

It can be useful in traffic modeling to incorporate also function describing workload $WLD_n$ associated with n-th delivered NDP. This workload can be dependent on queue delays in case traffic is redirected between multiple queues.
Both stochastic functions $PAT_n$ and $NDP_n$ are characterized by independent distributions. In some systems also the workload factor can be described by a stochastic distribution.
In related works authors describe different stochastic functions in each component of the traffic model. One of the oldest traffic model, which has been used in many analyzes uses Poisson process [18]. This process assumption random sequence arrive independently from one another and depends on constant value. This model can be successfully used to model different types of traffics like ex. VoIP [19]. There are also studies, which proves that Poisson modeling can fail [20] and using fractal (self similar) model [21] based on results analyze of hundreds of million packets observations in LAN and WAN area, can be more effective. Self-similar model cumulates traffic volume as self-similar process with increments that are strictly stationary to specified shifts in time. Alternative model for Poisson independence in arrival can be Markov process, which introduces dependence into random sequence. Markov chains can be used for example for TCP traffic classification [22], however the most commonly used Markov model is Markov-modulated Poisson process (MMPP) model, which can be used for modeling self-similar traffic [23], [24].

## 4. Networking System Model

The easiest way to determine possible system performance is to crate parameterized mathematical model and calculate system throughput indicators. There are many factors, which can affect system performance, in the multicore system architecture. Some parameters remain constant as shared memory access latency and some depend on other aspects like number of additional tasks executed on OS-es dedicated to separated cores.
*QIRx* is an inbound queue associated with I/O net1 in Fig. 2. This interface delivers set of $K$ data streams $D_k$ to the Core 1 OS networking stuck for preprocessing. Accepting burstiness limitation of Poisson process model, traffic can be described through a counting process:

$$D_k\{N(t+\lambda) - N(t) = n\} = \frac{(\tau\lambda)^n e^{-\tau\lambda}}{n!}\,, \tag{7}$$

where $N(t)$ is the number of packet arrivals at time $t$ from single source.

$$S(t) = \sum_{k=1}^{K} D_k(n_k), k \in N \tag{8}$$

For each stream $k$ number of arrival $n_k$ is called stream activity level (SAL) and should be represented by different constant value. This model parameter correspond to $NPD_n$ and could be parametrized by stochastic function.

Preprocessing core is responsible for saving packages from *QIRx* queue in dedicated shared memory queues: *SSME* – exception and *QSMF* – forwarding. This model assumes that in *QIRx* for every 10 packages $\eta$ is classified as forwarding and $10-\eta$ as exception. Shared memory queue can service $p$ packages in time $t$, but core ability to read and write to the queue also depends on several other factors. Our model limits these factors to number of tasks in CPU queue. Higher number of tasks in queue can increase CPU average waiting time [25] and writing/reading operation can be delayed. Little's formula describes relationship between $L$ – average number of processes in the queue and $W$ – CPU average waiting time. Parameter $\alpha$ can be associated with average arrival time, which is proportional to traffic rate.

$$W = \frac{L}{\alpha}. \tag{9}$$

System performance is often determined by traffic processing latencies. Sum of whole latencies specified by system model is able to show how long traffic will be processed by the system. In this system model $\delta L_1$ is latency specified between *QIRx* and *QSMF*/*QSME*

$$\delta L_1(t,\tau,L) = \frac{L_p}{S(t)}. \tag{10}$$

Latency $\delta L_2$ is specified between *SSME*, *QSMF* and its target interface. For exception core target interface would be $QITx$ and for forwarding it would be $QFTx$.

$$\delta L_2(t,\tau,L,\eta) = \frac{\frac{\eta}{10}L_f}{S(t)} + \frac{(1-\frac{\eta}{10})L_f}{S(t)} \tag{11}$$

Assuming, that traffic exception service suppose to notify packet sender about exception condition, there need to be considered additional process in the CPU preprocessing, responsible for dealing with back to sender traffic.

$$\delta L_2(t,\tau,L,\eta) = \frac{\frac{\eta}{10}L_f}{S(t)} + \frac{(1-\frac{\eta}{10})(L_f+L_p+1)}{S(t)} \tag{12}$$

Total system latency should also consider hypervisor latency. For this model this is constant value $\delta L_h$.

$$\delta L(t,\tau,L,\eta) = \delta L_1(t,\tau,L) + \delta L_2(t,\tau,L,\eta) + \delta L_h \tag{13}$$

For the sake of simplicity the model has been limited to consider only significant factors, which can influence the system performance. It can be extended to provide more detailed data, if this accuracy level is not satisfied enough to determine its value against standard general purpose CPU system. The authors' intention was to show how fluctuations of commonly accepted factors can affect the modeled system performance, e.g., influence the packet service latency.

## 5. Model Parameterization Results

Networking system model performance and scalability depends on several configurable system parameters, starts from incoming traffic, through different latencies and finish at operating system process utilization ability. Value of model presented in this paper is ability to verify, how system model parameters can influence whole system performance.

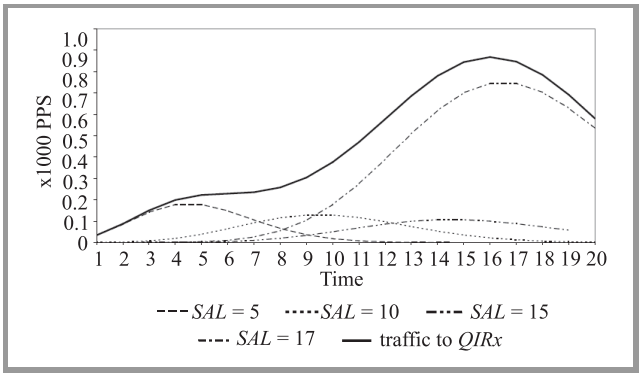


***Fig. 3.*** Data stream distribution associated with QIRx.

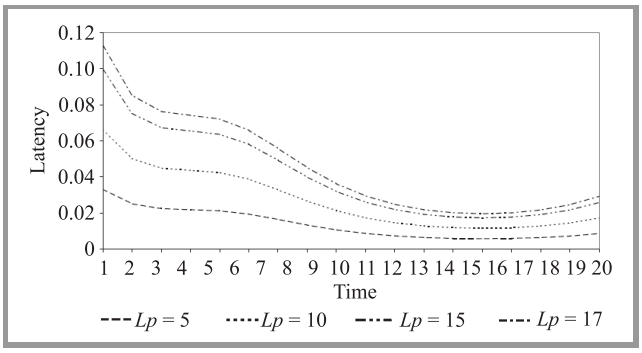


***Fig. 4.*** Latency between QIRx and QSMF/QSM.

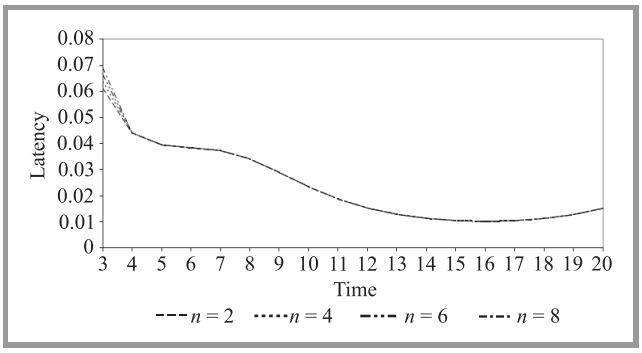


***Fig. 5.*** Latency between SSME, QSMF and its target interface, $L_p = 5$, $L_f = 5$ – approach 1.

Sample charts presented on Figs. 3–8 provides overview of system model parameters influence on measured latency. Multi queue approach assumes several parameters affecting performance value in more or less significant way. For example, number of task, which need to be serviced by OS in presented system model plays marginal role. Another parameters can affect time, in which packages are serviced, in more significant way. Data stream distribution represented by stochastic model and parametrized by system activity

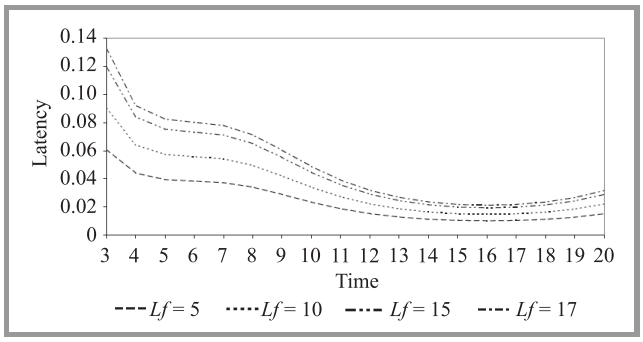


***Fig. 6.*** Latency between SSME, QSMF and its target interface, $n = 2$, $L_p = 5$ – approach 2.

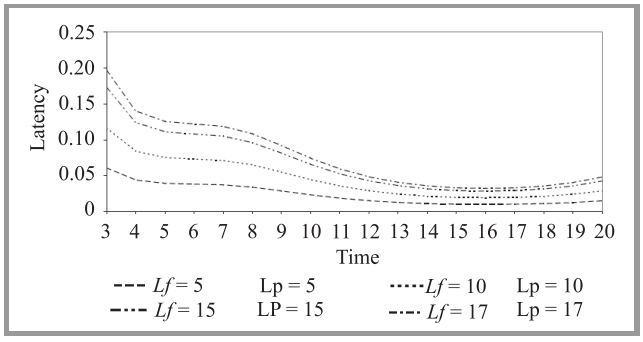


***Fig. 7.*** Latency between SSME, QSMF and its target interface, $n = 2$ – approach 3.

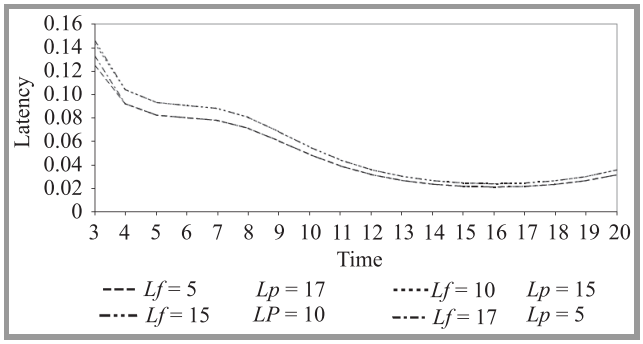


***Fig. 8.*** Latency between SSME, QSMF and its target interface, $n = 2$ – approach 4.

level can directly influence exception and forwarding queue latencies. Naturally higher number of transmitted packets (average arrival time) affect latencies and can cause system delays. Size of calculated lag seems to be reasonable small and its effect to whole system for most of the cases should be negligible, however system designers/architects should be aware, that in case of adverse stream distribution unexpected delays can happen.

## 6. Summary and Conclusions

The need for flexibility and performance and cost reduction in the networking systems make general purpose CPUs worth to be considered as valuable alternative for expensive solutions designed for packet processing. If system architecture agrees for general purpose computing limitations (like ex. us speed), there can be considered many system designs, which would make general purpose personal computer dedicated networking solutions. Fast development of CPU technologies allows to assumes, that CPUs dedicated to common marked are more capable to play valuable roles in dedicated (not generic) solutions. Good example can be presented in this paper usage of hypervisor technology in designing dedicated system model. The only limitation in the possible system architecture designs can be imagination of system architects. System model can help verify usability of dedicated solution assuming hardware/software limitation of model parameters. Networking system presented in this paper can be a good reference for further work, which could bring more detailed model providing more complex analyze of system performance indicator like queues delays, latencies as well as more self similar delivery distribution to better present ex. Ethernet characteristic. It has been proved that idea with specialized core functions (forwarding, exception), due to relatively small latencies caused by between core communication, could open easy way for generic purpose CPU usability in niches like eg. computer networking. Solutions based on generic purpose, multicore CPUs could be truly considered in complex, functionality oriented system designs.

## Acknowledgment

The work was partially supported by the Polish National Center for Research and Development under the PBZ grant MNiSW-02/II/2007.

## References

[1] "Intel IXP4XX product line of network processors", http://www.intel.com

[2] "OCTEON Multi-Core Processor Family", http://www.cavium.com/OCTEON_MIPS64.html

[3] "Semiconductors overview", http://www.freescale.com/

[4] "Improving network performance in multi-core systems", in Intel Corporation White Paper, http://www.intel.com

[5] "Intel CPU documentation", in Intel CPU, http://www.intel.com/design

[6] "Press release", in Intel Corporation, March 2007, http://www.intel.com/pressroom

[7] L. Shimpi, "AnandTech", June 2008, in The Nehalem Preview: Intel Does It Again, http://www.intel.com

[8] "Intel Core i7" in The Nehalem Preview: Intel Does It Again, http://www.intel.com

[9] M. Hasse and K. Nowicki, *Linux Scheduler Improvement for Time Demanding Network Applications, Running on Communication Platform Systems*. Gdańsk, Polska: Politechnika Gdańska, 2011.

[10] P. Barham *et al*., "Xen and the art of virtualization", in *Proc. ACM Symp. Operat. Sys. Principles*, New York, USA, 2003.

[11] A. Gavrilovska *et al*., "High-performance hypervisor architectures: virtualization in HPC systems", in *Proc. 1st Worksh. System-level Virtu. High Perform. Comput. HPCVirt 2007*, Lisbon, Portugal, 2007.

[12] "A Performance comparison of hypervisors", in *VMWare Performance Study*, Technical paper VMWare.

[13] "Guide to virtualization on Red Hat enterprise Linux", in *Virtualization Guide*, http://docs.redhat.com

[14] "Virtual box reference", in *Sun xVM Virtual Box*, http://www.virtualbox.org

[15] C. Pitter and M. Schoeber, "Time predictable CPU and DMA shared memory access", in *Proc. FPL 2007*, Amsterdam, The Netherland, 2007, pp. 317–322.

[16] K. Kolyshkin, "Virtualization in Linux", September 2006, http://www.pdfmob.com

[17] R. Ennals, R. Sharp, and A. Mycroft, "Task partitioning for multi-core network processors", in *Proc. Eur. Symp. Programming ESOP*, LNCS, 2005, vol. 3443, SpringerLink.

[18] L. Moddelmog and P. Johnson, "Poisson distribution", February 2006 [Online]. Available: http://pj.freefaculty.org/stat/Distributions/Exponential_v2.lyx

[19] I. Al Ajarmeh, J. Yu, and M. Amezzine, "Framework of applying a non-homogeneous Poisson process to model VoIP traffic on tandem networks", in *Proc. 10th WSEAS Int. Conf. Applied Informatics and Communications AIC 2010*, Taipei, Taiwan, 2010, pp. 164–169.

[20] V. Paxson and S. Floyd, "Wide-area traffic: the failure of Poisson modeling", *IEEE/ACM Trans. Netw.*, vol. 3, no. 3, pp. 226–244, 1995.

[21] W. Leleand, M. Taqqu, W. Willinger, and D. Wilson, "On the self similar nature of Ethernet traffic", *IEEE/ACM Trans. Netw.*, vol. 2, no. 1, pp. 1–15, 1994.

[22] G. Munz, H. Dai, L. Braun, and G. Carle, "TCP traffic classification using Markov models", in *Proc. Traffic Monitoring and Analysis Workshop TMA 2010*, Zurich, Switzerland, 2010, pp. 127–140.

[23] A. Nogueira, P. Salvador, R. Valadas, and A. Pacheco, "Modeling self-similar traffc through Markov modulated Poisson processes over multiple time scales", *Telecommun. Sys.*, vol. 17, no. 1–2, pp. 185–211, 2001.

[24] S. Scott and P. Smyth, *The Markov Modulated Poisson Process and Markov Poisson Cascade with Applications to Web Traffic Modeling.* Oxford University Press 2003.

[25] J. F. Brady, "Virtualization and CPU wait times in a Linux guest environmnet", January 2008.

**Marcin Hasse** received the M.Sc. degree in Telecommunication from the Gdańsk University of Technology, Poland in 2005. Currently he is working for embedded computing leading company providing solutions for telecommunication market. His research interest and current work are related to operating system improvements for networking/telecommunication usage scenarios. He is author of several publications in computer networking mechanisms improvements for end user services.
E-mail: marcin@hasse.pl
Gdańsk University of Technology
G. Narutowicza st 11/12
80-952 Gdańsk, Poland

**Krzysztof Nowicki** received his M.Sc. and Ph.D. degrees in Electronics and Telecommunications from the Faculty of Electronics, Gdańsk University of Technology (GUT), Poland, in 1979 and 1988, respectively. He is the author or co-author of more than 150 scientific papers and author and co-author of five books, e.g., "LAN, MAN, WAN – Computer Networks and Communication Protocols" (1998), "Wired and Wireless LANs" (2002) (both books were awarded the Ministry of National Education Prize, in 1999 and 2003, respectively), "Protocol IPv6" (2003), "Ethernet-Networks" (2006), Ethernet End-to-End. Eine universelle Netzwerktechnologie (2008). His scientific and research interests include network architectures, analysis of communication systems, network security problems, modeling and performance analysis of cable and wireless communication systems, analysis and design of protocols for high speed LANs.
E-mail: krzysztof.nowicki@eti.pg.gda.pl
Gdańsk University of Technology
G. Narutowicza st 11/12
80-952 Gdańsk, Poland

**Józef Woźniak** is a Full Professor in the Faculty of Electronics, Telecommunications and Computer Science at Gdańsk University of Technology. He received his Ph.D. and D.Sc. degrees in Telecommunications from Gdańsk University of Technology in 1976 and 1991, respectively. He is the author or co-author of more than 250 journal and conference papers. He has also co-authored 4 books on data communications, computer networks and communication protocols. In the past he participated in research and teaching activities at Politecnico di Milano, Vrije Universiteit Brussel and Aalborg University, Denmark. In 2006 he was Visiting Erskine Fellow at the Canterbury University in Christchurch, New Zealand. He has served in technical committees of numerous national and international conferences, chairing or co-chairing several of them. He is a member of IEEE and IFIP, being the vice chair of the WG 6.8 (Wireless Communications Group) IFIP TC6 and. For many years he chaired the IEEE Computer Society Chapter at Gdańsk University of Technology. His current research interests include modeling and performance evaluation of communication systems with the special interest in wireless and mobile networks.
E-mail: jozef.wozniak@eti.pg.gda.pl
Gdańsk University of Technology
G. Narutowicza st 11/12
80-952 Gdańsk, Poland