Paper

FR/ASimJava: a federated approach to parallel and distributed network simulation in practice

Andrzej Sikora and Ewa Niewiadomska-Szynkiewicz

Abstract— The paper addresses issues associated with the application of federations of parallel/distributed simulators to large scale networks simulation. We discuss two principal paradigms for constructing simulations today. Particular attention is paid to an approach for federating parallel/distributed simulators. We describe the design and performance of frame relay network simulator (FR/ASimJava) implemented based on a Java-based library for distributed simulation – ASimJava. Six practical examples – six networks operating under frame relay – are presented to illustrate the operation of the given software tool. The focus is on the efficiency of presented network simulator.

Keywords— parallel simulation, computer networks simulation, frame relay, federated simulators.

1. Introduction

Network simulation is an important tool for researches that allows to analyze the behavior and performance of the considered network and verify new ideas. A variety of software environments simulating packets transmission through the network are available today. There are a number of possible sets of criteria that could be used for network simulators comparison, e.g., the model size, the execution time, memory requirements, scalability, programming interface, etc. Different tools are optimized for different purposes. The comparative study of some popular simulators are reported in many papers, e.g., the results of the performance study involving: JavaSim [7], ns-2 [19], SSFNet-Java [18], and SSFNet-C++ [17] are described in [13], the comparison of ns-2, JavaSim and OPNET [15] is concluded in [9].

We are involved in large heterogenous networks simulating in near real time. The main difficulty in packet level simulation is the enormous computational power, i.e., speed and memory requirements needed to execute all events involved by packets transmission through the network. Another problem is scalability, i.e., how a given simulator scales for large topologies and high speed links. Parallel and distributed simulation has already proved to be very useful when performing the analysis of different real complex systems [20]. It allow us to reduce the computation time of the simulation programme, to execute large programs that cannot be put on a single processor and to better reflect the structure of physical system. Last years a new paradigm for constructing parallel and distributed simulations was developed. It is based on the idea of federating disparate simulators, utilizing runtime infrastructure to interconnect them. In this paper we investigate issues concerning federations of parallel simulators. A novel approach to scalability and efficiency of parallel/distributed frame relay (FR) network simulation is described and discussed.

2. An approach for federating parallel/distributed simulators

Parallel/distributed discrete-event simulation can be described in terms of logical processes (LPs) and communicate with each other through message-passing. LPs simulate the real life physical processes (FPs). Each logical process starts processing as a result of event occurrence (from the event list or having received a new message). It performs some calculations and generates one or more messages to other processes.

The calculation tasks executed in parallel require explicit schemes for synchronization. Two simulation techniques are considered [9]: synchronous and asynchronous. Synchronous simulation is implemented by maintaining a global clock (global virtual time – GVT). Events with the smallest time-stamp are removed from the event lists of all LPs for parallel execution. Parallelism of this technique is limited because only events with time-stamps equal to that of the global clock can be executed during an event cycle. Asynchronous simulation is much more effective due to its potentially high performance on a parallel platform. In asynchronous simulation each logical process maintains its own local clock (local virtual time – LVT). Local times of different processes may advance asynchronously. Events arriving at the local input message queue of a logical process are executed according to the local clock and the local schedule scheme.

Synchronization mechanisms fall into two categories: conservative and optimistic. They differ in their approach to time management. Conservative schemes avoid the possibility of causality error occurring. These protocols determine safe events that can be executed. Optimistic schemes allow occurrence of causality errors. They detect such error and provide mechanisms for its removal. The calculations are rolled back to a consistent state by sending out antimessages. It is obvious that in order to allow rollback all results of previous calculations have to be recorded. Now, there are two basic directions to take when developing parallel and distributed simulators [2]:

- development of a problem dedicated (specialized) simulators, specific to the environment for which they were created;
- development of general purpose simulators designed as federations of disparate simulators, utilizing run-time infrastructure (RTI) software to interconnect them.

In the case of the first paradigm the simulation engine, interface, libraries and tools to create new high performance simulators are defined. It is difficult, in general, for the user to modify and apply such software to new environments. A second paradigm results in coarse-grained set of simulators creating federation. It is assumed that his entire simulators are viewed as black boxes. They are designated as federates. The runtime infrastructures used for federates interconnecting are typically designed for coarse granularity concurrency. This approach is utilized in high level architecture (HLA) [5] standard for distributed discrete-event simulation. The main advantage is high possibility of simulation models reuse. However, we pay for this universal applicability. This approach imposes certain restrictions concerning the structure of the federation members. In addition federates have to obey some rules of the federation that are included in.
The federated, distributed simulation consists of a collection of autonomous simulators that are interconnected using RTI software. The RTI implements relevant services required by the federated simulation environment. The most important services are: time synchronization among federated simulators, secure and efficient communication and scalable platform architecture.
Network analysis and modeling concentrate on studying network components (from devices to requirements and performance levels) and their inputs and outputs. We are interested to evaluate simulated system operation under various real-life conditions. Two important characteristics of networks: interconnectivity and levels of hierarchy (from network core to user access level) [10] have to be concerned. Based on these characteristics we can define a set of network submodels. We assume that the simulator of each submodel implements only a part of the network being simulated. The advantage of such application is that there is no need to provide the shared memory access to describe the whole simulated network system. Physical connectivity between federated simulators executes a set of logical connections between submodels. An example of this approach is shown in Fig. 1.
Using the paradigm of federated simulators, an implementation of a federation of asynchronous parallel software modules simulating frame relay network was developed and examined.

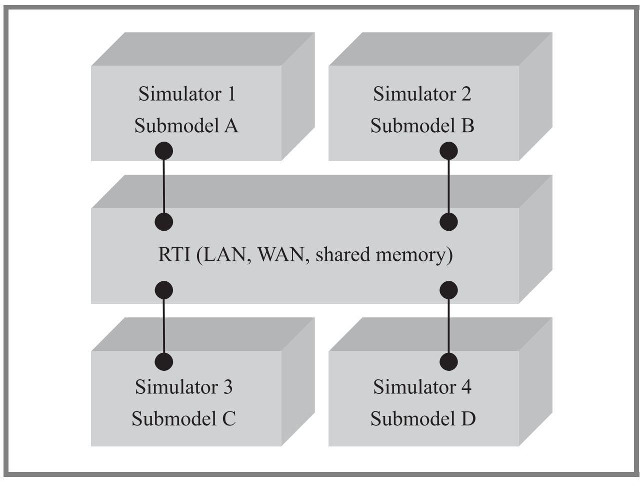

Fig. 1. An architecture of federated simulator.

3. Parallel simulation of frame relay networks

3.1. ASimJava library

Frame relay/asynchronous simulation Java (FR/ASimJava) network simulator is a parallel/distributed fast simulator of frame relay networks. It was implemented based on ASimJava, a Java-based library for large-scale systems simulation. Although ASimJava was described in [14], we provide a brief summary here, to make the paper self-contained. The ASimJava library admits to do parallel and distributed discrete-event simulations that can be described in terms of logical processes and communicate with each other through message-passing. LPs simulate the real life physical processes. Each logical process starts processing as a result of event occurrence (from the event list or having received a new message). It performs some calculations and generates one or more messages to other processes. The calculation tasks executed in parallel require explicit schemes for synchronization. The synchronous and asynchronous [20] variants of simulators are available. In the case of asynchronous approaches four synchronization protocols are provided: conservative protocol with null messages (CMB) [11], window conservative protocol [12], time warp (TW) [8], moving time window protocol (MTW) [16].
The simulator built upon ASimJava classes has hierarchical structure. The simulated system is partitioned into several subsystems (subtasks), with respect to functionality and data requirements. Each subsystem is implemented as LPs. Each LP can be divided into smaller LPs. Hence, the logical processes are nested (Fig. 2). Calculation processes belonging to the same level of hierarchy are synchronized. The module-oriented architecture of ASimJava library allows developers to add new components. One of these modules is bidirectional interface to XML configuration and state save file that uses ASimL language – XML schema[1] specification for building XML file with description of pa-

[1] See http://www.w3.org standard

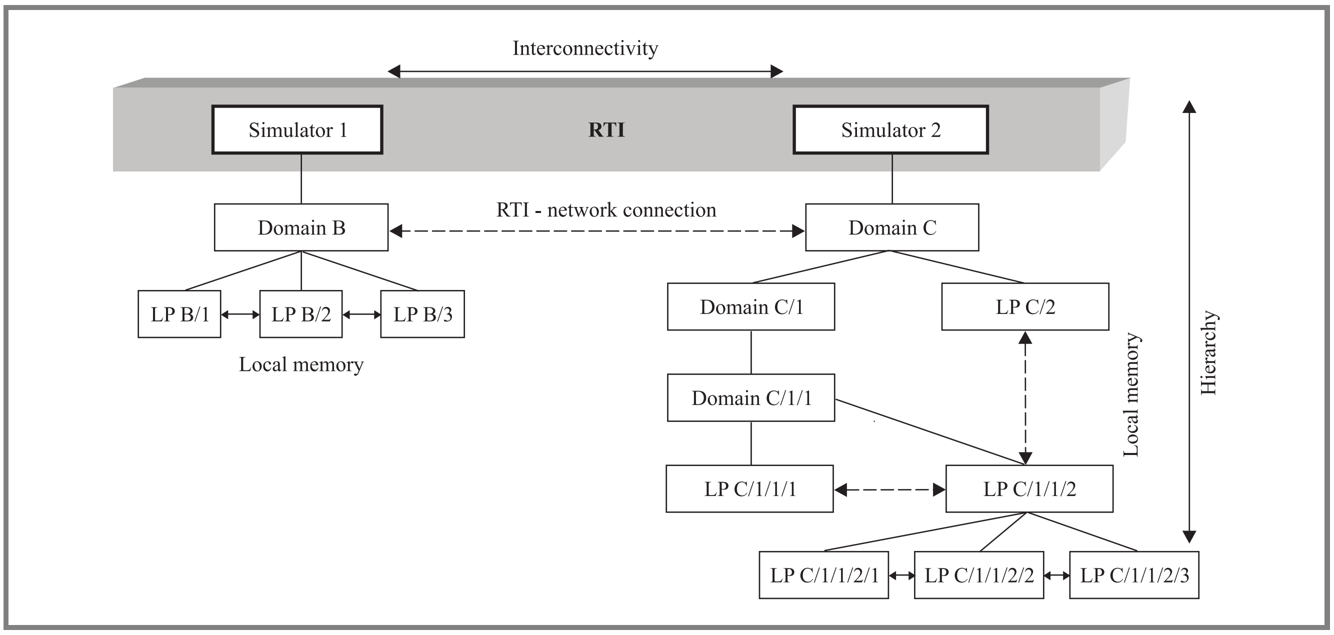

Fig. 2. A federation of network simulators consisting of two members: Simulator 1 and Simulator 2.

rameterized system model. Simulator configuration can be fully loading (run, re-run) from XML file, it may contain any number of user-defined parameters.
Two types of simulators can be distinguished:

1) the simulator consisting only of classes provided in ASimJava; the structure of the simulated system together with all model parameters is created using ASimJava graphical interface or may be read from an XML file;

2) new simulator – the user's task is to implement the subsystems' simulators responsible for adequate physical systems simulation; he can create his application applying adequate classes from the ASimJava libraries and including his own code – numerical part of the application.

As one of the ASimJava's principle goals was portability and usage in heterogeneous computing environments. Two versions of ASimJava are implemented: parallel and distributed. It is possible to join both of them in one simulator. The JXTA technology platform provided by Sun Microsystems was used to interprocess communication in the case of distributed version of the library. JXTA is a set of open, generalized peer-to-peer (P2P) protocols that allow any connected device on the network to communicate and collaborate as peers (see Fig. 3). The JXTA protocols are independent of any programming language, and multiple implementations exist for different environments. This technology enables developers to build and deploy interoperable P2P services and applications. The JXTA protocols standardize the manner in which peers:

– discover each other peer,

– self-organize into peer groups,

– advertise and discover network services,

– securely communicate with each other,

– monitor each other peer remotely.

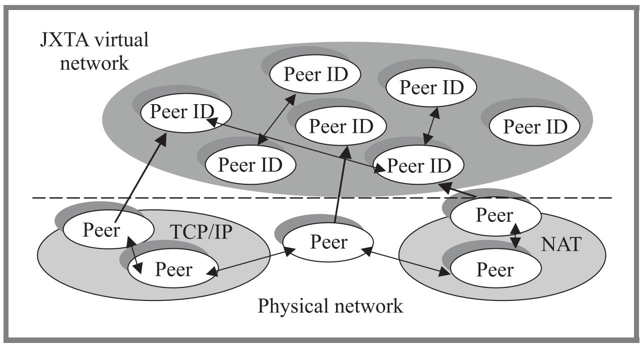

Fig. 3. JXTA logical network mapping. Explanations: ID – identification number, NAT – network address translation, TCP/IP – transmission control protocol/Internet Protocol.

The ASimJava software framework is suitable to solve many small and large scale problems, based on simulation. The package is flexible and can be easily extended by software modules, which are specific to a chosen application.

3.2. Description of FR/ASimJava simulator

Frame relay is a high-performance wide area network (WAN) protocol that operates at the physical and data link layers of the open system interconnection (OSI) reference model. This is a standard protocol for local area network (LAN) internetworking which provides a fast and efficient method of transmitting information from a user

device to LAN bridges and routers. Frame relay is an example of a packet-switched technology. Variable-length packets are used for more efficient and flexible data transfers. These packets are switched between various segments in the network until the destination is reached. Internationally, frame relay was standardized by the International Telecommunication Union – Telecommunications Standards Section (ITU-T) [4]. In the United States, frame relay is an American National Standards Institute (ANSI) standard [6].
Frame relay traffic is described based on several characteristic parameters. The detailed information about FR parameters one can find on frame relay forum webside [3].

The FR/ASimJava simulator implements the following parameters describing traffic characteristics in frame relay networks:

- committed information rate (*CIR*);
- excess information rate (*EIR*);
- committed information size (*Bc*):
  $Bc = CIR \times Tc$,
  where Tc denotes assumed time interval;
- excess information size (*Be*):
  $Be = EIR \times Tc$;
- a physical line speed of the interface connecting to the frame relay network (*access rate*):
  $\sum_i^I CIR_{ij} \leq access\ rate_j$,
  where $i \in I$ denotes client and j connection.

The following features of FR protocol are taken into considerations in our implementation:

- permanent virtual circuit (PVC);
- switched virtual circuit (SVC);
- data terminal equipment (DTE) – an edge, access routers:
  - classifying traffic – any number of class that can be used in quality of service (QoS) mechanism,
  - measuring and marking excess traffic (if greater than *CIR*) with bit discard eligibility DE = 1,
  - shaping input traffic (leaky bucket, token bucket),
  - dropping "bursty" traffic (if greater than $CIR + EIR$),
  - backward explicit congestion notification (BECN) and forward explicit congestion notification (FECN) reaction,
  - SVC negotiation,
  - stochastic traffic generators for all virtual circuits;
- Data circuit-terminating equipment (DCE) – switches (see Fig. 4):
  - quality of service: input and output buffers management: first in, first out (FIFO), priority queuing (PQ), class-based weighted fair queuing (CBWFQ), PQ-CBWFQ discipline,
  - BECN and FECN signaling,
  - SVC negotiation,
  - switching and routing.

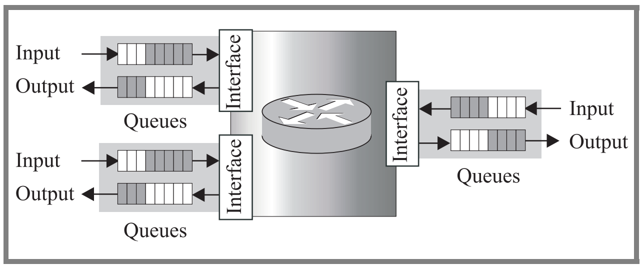

Fig. 4. Exampled architecture of switch used in FR/ASimJava simulator.

All parameters of simulation model, network topology, characteristics of data flows (traffic) and frame relay mechanism are saved in XML configuration file. This file can be simply modified and reused in many simulations.

4. Case study results

In the presented case study we evaluate the complexity of frame relay network simulation. In this paper the results of experiments performed for four network configurations, examples E1–E4 describing different model size, and two variants of implementation – S (sequential) and D (distributed) are discussed. The detailed descriptions, i.e., network models and traffic characteristics are given in Table 1. During the tests, we measured the simulation time (the execution time of each experiment). We assumed in all tests the same simulated time – 30 seconds of physical network operation. The objective of presented case study was to compare the efficiency of parallel, federated simulators with the sequential realization. To compare the performance of packet-level simulators we used two characteristics, i.e., simulation time (execution time) in miliseconds and average simulator speed simulated packets transmissions per second (PTS) [4] defined in Eq. (1):

$$PTS \approx \left(\frac{N_F \cdot P_F \cdot H_F}{T} \right), \tag{1}$$

where T denotes the execution time, N_F – the number of flows (edge router to edge router), P_F – the number of packets sent per flow, H_F – the average hops per flow (queuing, transmitting over link, etc.). The presented definition ignores lost packets, protocol generated packets.

Table 1
Results of experiments: four exampled frame relay networks (variant S)

Example	Network model	LPs number	Packets number	Simulation time [ms]	*PTS*
E1	1 switch 2 interfaces 2 edge routers 2 links 1.544 Mbit/s	7	$(\sim 13500 \cdot 2) = 27000$	4800	22.5
E2	1 switch 6 interfaces 6 edge routers 6 links 1.544 Mbit/s	19	$(\sim 13500 \cdot 6) = 81000$	12900	25.1
E3	2 switches 14 interfaces 12 edge routers 12 links 1.544 Mbit/s 1 link 44.736 Mbit/s	42	$(\sim 13500 \cdot 12) = 162000$	24300	46.6
E4	3 switches 22 interfaces 18 edge routers 18 links 1.544 Mbit/s 2 links 44.736 Mbit/s	65	$(\sim 13500 \cdot 18) = 243000$	34100	71.2

Table 2
Computer systems used during experiments

Computer systems	AMD Athlon-M 1.2 GHz, 512 RAM	AMD Sempron 1.67 GHz, 512 RAM	AMD Sempron 1.67 GHz, 512 RAM
C1	X		
C2	X	X	
C3	X	X	X

The results of simulation experiments performed on single machine (a computer system C1 described in Table 2) are presented in Table 1. It can be observed that the execution time of experiment performed for exampled network E4 exceeds the real time operation of the physical network (the simulation time is greater than simulated, virtual time).

The second series of experiments was performed in the network of computers. Two hardware platforms were considered: C2 – the network of two machines, C3 – the network of three machines (see Table 2).

Two exampled networks E3 and E4 were taken into considerations. The simulator of the whole network was composed of two federated simulators in the case of example E3 and three federated simulators in the case of E4 (see Fig. 5). The calculations of each member of federation were performed by separate computer. The window conservative scheme described in [12, 14] was applied to federated simulators synchronization.

The submodels configurations, execution time of each experiment and simulators speeds are given in Table 3.

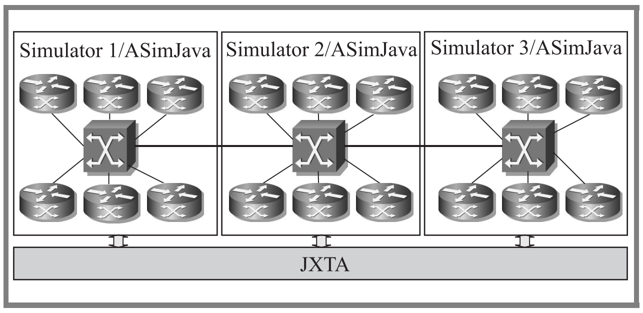

Fig. 5. Federated simulator of exampled network E4.

As expected, we can observe that federated, distributed simulation can seriously speed up simulations of network operation w.r.t. sequential implementation. The calculation speed-up depends on the size of considered network model and assumed degree of parallelism. It should be indicated that in the case of distributed implementation the reserve of efficiency to meet real time requirements is quite large (see E4 in Table 3).

Table 3
Results of experiments: two exampled frame relay networks (variant D)

Example	Network model	LPs number	Packets number	Simulation time [ms]	*PTS*
E3	1 switch 7 interfaces 6 edge routers 6 links 1.544 Mbit/s 1 link 44.736 Mbit/s	21	$(\sim 13500 \cdot 12) = 162000$	19300	58.7
	1 switch 6 interfaces 6 edge routers 1 link 1.544 Mbit/s	21			
E4	1 switch 7 interfaces 6 edge routers 6 links 1.544 Mbit/s 1 link 44.736 Mbit/s	21	$(\sim 13500 \cdot 18) = 243000$	19300	**125.2**
	1 switch 6 interfaces 6 edge routers 6 links 1.544 Mbit/s 1 link 1.544 Mbit/s	21			
	1 switch 8 interfaces 6 edge routers 6 links 1.544 Mbit/s	23			

5. Conclusions

In this paper we described the federated approach to parallel and distributed simulation of frame relay networks. We demonstrated that this approach is suitable to perform fast simulations of large-scale networks. Our experiences with federated, distributed network simulations confirm the ability of the federated simulation approach to achieve large and detailed simulation models. JXTA peer-to-peer technology and ASimJava library allow us to use Internet and other computer networks as a secure simulation platform.

References

[1] American National Standards Institute, http://www.ansi.org

[2] S. L. Ferenci, K. S. Perumalla, and R. M. Fujimoto, "An approach for federating parallel simulators", in *Proc. 14th Worksh. Parall. Distrib. Simulat. PADS 2000*, Bologna, Italy, 2000, pp. 63–70.

[3] Frame relay forum, http://www.frforum.com

[4] R. M. Fujimoto, K. S. Perumalla, A. Park, H. Wu, M. Ammar, and G. Riley, "Large scale network simulation: how big? how fast?", in *Proc. Symp. Modell., Anal. Simulat. Comput. Telecommun. Syst.*, Orlando, USA, 2003, pp. 116–125.

[5] High level architecture (HLA) homepage, http://www.dmso.mil/public/transition/hla/

[6] International Telecommunication Union, http://www.itu.int/home/

[7] JavaSim homepage, http://www.javasim.org

[8] D. A. Jefferson, "Virtual time", *ACM Trans. Programm. Languag. Syst.*, vol. 7, no. 3, pp. 404–425, 1985.

[9] M. Małowidzki, "Network simulators: a developer's perspective", in *Proc. Int. Symp. Perform. Eval. Comput. Telecommun. Syst. SPECTS'04*, San Jose, USA, 2004.

[10] J. D. McCabe, *Network Analysis, Architecture, and Design*. San Francisco: Morgan Kaufman, 2003.

[11] J. Misra, "Distributed discrete-event simulation", *Comput. Surv.*, vol. 18, no. 1, pp. 39–65, 1986.

[12] D. M. Nicol and R. Fujimoto, "Parallel simulation today", *Ann. Oper. Res.*, vol. 53, pp. 249–285, 1994.

[13] D. M. Nicol, "Utility analysis of network simulators", *Int. J. Simulat. Syst., Sci. Technol.*, vol. 4, pp. 55–69, 2003.

[14] E. Niewiadomska-Szynkiewicz and A. Sikora, "ASimJava: a Java-based library for distributed simulation", *J. Telecommun. Inform. Technol.*, no. 3, pp. 12–17, 2004.

[15] OPNET modeler homepage, http://www.opnet.com/products/modeler/home.html

[16] L. M. Sokol, D. P. Briscoe, and A. P. Wieland, "MTW: a strategy for scheduling discrete simulation events for concurrent execution", in *Proc. SCS Multiconf. Distrib. Simulat.*, San Diego, USA, 1988, pp. 34–42.

[17] SSFNET-C++ homepage, http://www.cs.dartmouth.edu/ghyan/dassfnet/overview.htm

[18] SSFNET-Java homepage, http://www.ssfnet.org

[19] The network simulator ns-2 homepage, http://www.isi.edu/nsnam/ns/

[20] B. P. Zeigler, H. Praehofer, and T. G. Kim, *Theory of Modeling and Simulation*. San Diego: Academic Press, 2000.

Andrzej Sikora received the M.Sc. degree in 2002 from the Warsaw University of Technology, Poland. Currently he is a Ph.D. candidate in computer science in the Institute of Control and Computation Engineering at the Warsaw University of Technology. He is a specialist in the application of control and simulation of distributed systems. He has experience in the use of parallel asynchronous computation, modeling, computer simulation, computer networks, Internet techniques, optimization and decision support, workflow application (Lotus Notes&Domino). His computer skills include programming using Java, C, C++, SQL, HTML, XML, Perl, and RMI.
e-mail: a.sikora@elka.pw.edu.pl
Research and Academic Computer Network (NASK)
Wąwozowa st 18
02-796 Warsaw, Poland

Ewa Niewiadomska-Szynkiewicz received the M.Eng., Ph.D., and Ds.C. degrees in control and computer engineering from the Warsaw University of Technology, Poland. Since 1987 with Warsaw University of Technology, she is a lecturer on simulation technologies parallel computing and optimization techniques. Since 2001 she is also with Research and Academic Computer Network (NASK), where she is leading a group in the field of IP network modeling and simulation. She was involved in a number of research projects including three EU projects, the author of over 60 journal and conference papers and co-author of one book. Her current interests are: computer simulation, parallel computing, complex systems optimization and control and global optimization.
e-mail: ens@ia.pw.edu.pl
Institute of Control and Computation Engineering
Warsaw University of Technology
Nowowiejska st 15/19
00-665 Warsaw, Poland