Simulation and Performance Analysis of Distributed Internet Systems Using TCPNs

Slawomir Samolej and Tomasz Rak
Department of Computer and Control Engineering, Rzeszow University of Technology, Poland
E-mail: ssamolej, trak@prz-rzeszow.pl and http://ssamolej, trak.prz-rzeszow.pl

Keywords: distributed systems modeling, performance analysis, timed coloured Petri nets

Received: December 6, 2008

This paper presents a Timed Coloured Petri Nets based programming tool that supports modeling and performance analysis of distributed World Wide Web environments. A distributed Internet system model, initially described in compliance with Queueing Theory (QT) rules, is mapped onto the Timed Coloured Petri Net (TCPN) structure by means of queueing system templates. Then, it is executed and analyzed. The proposed distributed Internet systems modeling and design methodology has been applied for evaluation of several system architectures under different external loads.

Povzetek: Predstavljeno je orodje na osnovi Petri mrež za modeliranje spletnih okolij.

1 Introduction

One of modern Internet (or Web) systems development approaches assumes that the systems consist of a set of distributed nodes. Dedicated groups of nodes are organized in layers (clusters) conducting predefined services (e.g. WWW service or data base service) [2, 6, 8]. This approach makes it possible to easily scale the system. Additionally, a distributed structure of the system assures its higher dependability. Fig. 1 shows an example cluster–based Internet system structure. The Internet requests are generated by the clients. Then they are distributed by the load balancer among set of computers that constitute the front-end or WWW cluster. The front–end cluster offers a system interface and some procedures that optimize the load of the next system layer–the database servers cluster. In the standard scenario the client produces the request by e.g. filling out the formula on the web side. Then the request is converted into e.g. a SQL query and forwarded to the database layer. The result of the query is sent back to the front-end layer. Finally, the client receives the result of his request on the website.

Simultaneously, for a significant number of Internet applications some kind of soft real-time constraints are formulated. The applications should provide up-to-date data in set time frames [20]. Stock market or multimedia applications may be good examples of hardware/software systems that may have such timing requirements.

The appearing of new above mentioned development paradigms cause that searching for a new method of modeling and timing performance evaluation of distributed Internet systems seems to be an up-to-date research path.

One of intensively investigated branch of Internet systems software engineering is formal languages application for modeling and performance analysis. Amid suggested solutions there are: algebraic description [11], mapping

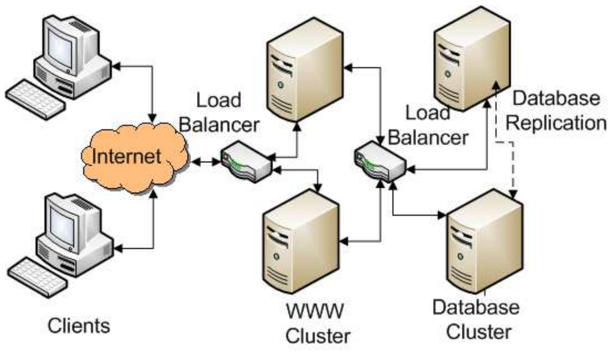


Figure 1: Example distributed cluster–based Internet system.

through Queueing Nets (QNs) [8, 18], modeling using both Coloured Petri Nets (CPNs) [12] and Queueing Petri Nets (QPNs) [6, 7].

Our approach proposed in this paper[1] may be treated as extension of selected solutions summed up in [6, 7], where Queueing Petri Nets (QPNs) language [1] has been successively applied to the web–cluster modeling and performance evaluation. The final QPNs based model can be executed and used for modeled system performance prediction. In our solution we propose alternative Queueing Systems models defined as in [3] expressed into Timed Coloured Petri Nets (TCPNs) [4]. The models has been used as a background for developing a programming tool which is able to map timed behavior of queueing nets by means of simulation. Subsequently we developed our individual TCPNs–based method of modeling and analysis

[1]An earlier version of the manuscript was published in Proceedings of the International Multiconference on Computer Science and Information Technology, 2008, pp. 559–566.

of distributed Internet systems. The well known software toolkits as Design/CPN or CPN Tools can be naturally used for our models simulation and performance analysis [10, 19]. The preliminary version of our software tool was announced in [13], the more mature its description can be found in [16].

The remaining work is organized as follows. In section 2 we informally introduce basic concepts of TCPNs and then in section 3 we provide rules of mapping queueing systems into TCPNs. In the next section, we present a method of applying the TCPNs based queueing systems models (TCPNs templates) to distributed Internet system modeling. Section 5 focuses on results of simulation some detailed Internet system models while section 6 sums up the paper and includes our future research plans.

2 Hierarchical timed coloured Petri nets' basic concepts

As TCPNs is the main formal language exploited in the paper we decided to briefly introduce it. The introduction will have an informal form focusing only on the most important TCPNs features. The more thorough TCPNs informal introductions can be found in [9, 5]. The detailed TCPNs features and some applications are presented in [4].

Informally, a Timed Coloured Petri Net is a bipartite graph consisting of "place" nodes and "transition" nodes. The places, drawn as circles or ellipses, are used to represent conditions; the transitions, drawn as bars, are used to represent events.

The places can have some "tokens" associated with. Each token is equipped with an attached data value - the token "colour". The data value may be freely complex (e.g. integer number or record of data). For each place, a colour set is defined which characterizes acceptable colours of the token in the place. All declarations concerning the behavior of the net and colours of tokens are written in the CPN ML language. It is possible to define colours, variables, expressions and functions connected to the elements of the net. The distribution of tokens in the places is called marking. The initial marking determines the initial state of the net and is specified by "initialization expressions". The marking of each place is a multi-set over the colour set attached to the place (compare [4]).

Directed arcs (arrows) connect the places and transitions, with some arcs directed from the places to the transitions and other arcs directed from the transitions to the places. An arc directed from place to transition defines the place to be an input of the transition. Multiple inputs to a transition are indicated by multiple arcs from the input places to the transition. An output place is indicated by an arc from the transition to the place. Again, multiple outputs are represented by multiple arcs. The tokens can "use" the arcs to move from place to place. Moreover, "arc expressions" (the expressions that may be attached to arcs) decide on the token flow in the net. Arc expressions may denote the number and the colour set of flowing tokens, as well as may be functions that manipulate tokens and their colours.

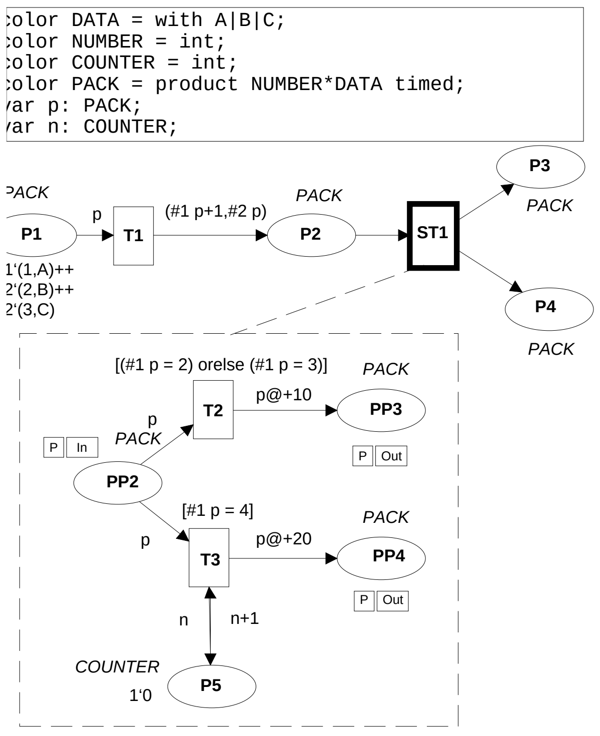
The marking of a Petri net changes by the "occurrence" of transitions. Before the occurrence of a transition, the variables defined in the net are bounded to colours of corresponding types, which is called a binding. A pair `(t, b)` where `t` is a transition and `b` a binding for `t` is called a binding element. For each binding, it can be checked if the transition is enabled to fire in a current marking. An enabled transition may occur. The occurrence of a transition is an instantaneous event during which tokens are removed from each of transition's input places and tokens are deposited in its output places.

The transitions of the net may have "guards" attached to them. The guards are boolean expressions that may define additional constraints that must be fulfilled before the transition is enabled.

For the effective modeling TCPN enable to distribute parts of the net across multiple subnets. The ability to define the subnets enables to construct a large Hierarchical TCPN by composing a number of smaller nets. Hierarchical TCPNs offer two mechanisms for interconnecting TCPN structure on different layers: "substitution transitions" and "fusion places". A substitution transition is a transition that stands for a whole subnet of net structure. A fusion place is a place that has been equated with one or more other places, so that the fused places act as a single place with a single marking. Each hierarchical TCPN can be translated into behaviorally equivalent non-hierarchical TCPN, and vice versa. Thanks to the ability to handle additional information or data in a TCPN structure and thanks to introduction of the hierarchy, the nets manage to model complex systems in a consistent way.

To model a timing behavior of systems TCPNs posses the following features. There exist a global clock whose values represent the current model time. Tokens may carry a time value, also denoted as a time stamp. The time stamp represents the earliest model time at which the token can be used. Hence, to occur, a transition must be both colour enabled and ready. It means that the transition must fulfill the usual enabling rule and all the time stamps of the removed tokens must be less than or equal to the current model time. When a token is created, the time stamp is specified by an expression. This means it is possible to specify all kinds of delays (e.g. constant, interval, or probability distribution). Moreover, the delay may depend upon the binding of the transition that creates the token.

Fig. 2 includes an example HTCPN modeling a simple data distribution system. The declaration node at the top of the figure defines 4 token types (colours). `DATA` token type defines 3 possible enumerated values: A, B or C. `NUMBER` and `COUNTER` token types rename the basic integer type. `PACK` colour is a tuple including two elements: the number and the value of the data package distributed over the modeled system. Two variables: `p` and `n` are also defined in the declaration node. They may carry `PACK` and `COUNTER` data values, respectively.



```
olor DATA = with A|B|C;
olor NUMBER = int;
olor COUNTER = int;
olor PACK = product NUMBER*DATA timed;
ar p: PACK;
ar n: COUNTER;
```

Figure 2: Example HTCP Net

For `P1` and `P5` places the initial markings are defined. For example, initially place `P1` will include 1 token of (1,A) value, 2 tokens of (2,B) value and 2 tokens of (3,C) value. As `P1` place has some tokens `T1` transition may occur. During the occurrence of `T1` transition arc inscription associated to the arc connecting `T1` transition and `P2` place modifies the currently transported token. It increments by 1 each first element of the `PACK` type tuple. Then the packets are distributed over the `P3` and `P4` places. The distribution procedure is "hidden" under the `ST1` substitution transition. Guard function connected to `T2` transition allows to "pass" only these data packets whose `NUMBER` field is 2 or 3. The remaining tokens may go through `T3` transition. Moreover the firing of `T3` transition increments a token value stored in `P5` place. As a result `P5` stores the number of data packets received by `P5` place. Data packets that are transported over the `T2` and `T3` transitions acquire additionally some new time stamps. This may be interpreted that the event connected to `T2` or `T3` transition occurrence takes 10 or 20 time units, respectively.

The remaining HTCPNs structures presented in this paper use some similar techniques to manage the token distribution. Some arc inscriptions are defined as "CPN ML functions". The selected parameters of the modeled systems are stored as "values" not mentioned in the example.

3 Queueing system implementation

Queueing Net usually consists of a set of connected *queueing systems*. Each *queueing system* is described by an arrival process, a waiting room, and a service process. In the proposed programming tool, we worked out several TCPNs based *queueing system templates* (e.g. – /M/PS/∞, – /M/FIFO/∞) most frequently used to represent properties of distributed Internet system components. Each template represents a separate TCPN net (subpage) which may be included in the model of the system as a substitution transition (using hierarchical CP nets mechanisms [4]).

3.1 Queueing system mapping

Queueing system properties are mapped to the TCPNs net as follows. At a certain level of system description, a part of hardware/software is modeled as a TCPN, where some dedicated *substitution transitions* are understand as queueing systems (compare fig. 3). To have the queueing functionality running "under" selected transitions the mapping to adequate TCPNs subpages must be done. The corresponding subpages include the implementation of the adequate queueing system. In fig. 3 an example -/M/1/PS/∞ (exponential service times, single server, Processor Sharing service discipline and unlimited number of arrivals in the system; the queue's arrival process in our modeling approach is defined outside of queueing system model) queueing model definition path is presented.

Packets to be served by the example queueing system are delivered by port place `INPUT_PACKS`. Then, they are scheduled in a queue in `PACK_QUEUE` place. Every given time quantum (regulated by time multiset included in `TIMERS` place) the first element in the queue is selected to be served (execution of transition `EXECUTE_PS`). Then, it is placed at the end of the queue or directed to leave the system (execution of transition `ADD_PS1` or `REMOVE_PS` respectively). Number of tokens in `TIMERS` place represents number of servers for queueing system.

3.2 Internet requests modeling

Full description of the model requires colors and functions definition in CPN ML language connected to the net elements:

```
(*———————System parameters:—————————*)
val ps_ser_mean_time=100.0;
val pack_gen_mean_time=220.0;
(*——Internet request definition:————*)
color ID=int;   color PRT=int;
color START_TIME=int; color PROB=int;
color AUTIL=int; color RUTIL=int;
color INT=int; color PACKAGE=
product ID*PRT*START_TIME*PROB
        *AUTIL*RUTIL timed;
color PACK_QUEUE=list PACKAGE;
(*——Auxiliary types and variables:—*)
color random_val=int with 1..100;
var tim_val:INT; var n:INT;
color TIMER=int timed;
```

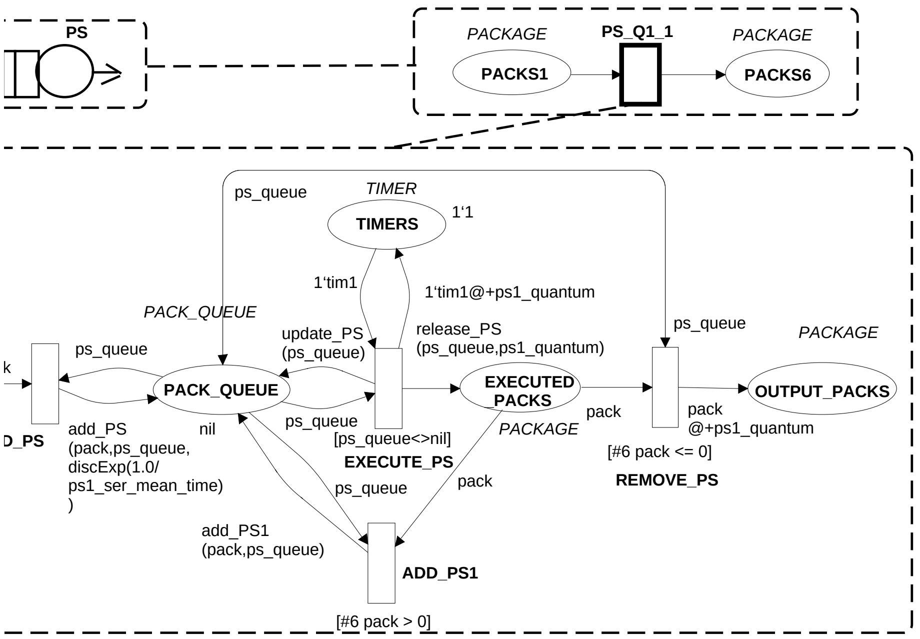


Figure 3: TCPNs model of -/M/1/PS/∞ queuing system.

```
var tim1:TIMER;   var pack:PACKAGE;
var ps_queue: PACK_QUEUE;
```

Corresponding arc functions (`add_PS()`, `add_PS1()`, `update_PS()`, `release_PS()`) release or insert tokens within the queue:

```
(*--Add request to queue(1):--------------*)
fun add_PS(pack:PACKAGE, queue:PACK_QUEUE,
ser_time:int)=if queue = nil
then [(#1 pack,#2 pack,#3 pack,#4 pack,
ser_time,ser_time)]
else (#1 pack,#2 pack,#3 pack,#4
pack,ser_time,ser_time)::queue;
(*--Add request to queue(2):--------------*)
fun add_PS1(pack:PACKAGE, queue:PACK_QUEUE)=
if queue=nil
then [pack] else pack::queue;
(*--Withdraw request from queue (1):-----*)
fun update_PS(queue:PACK_QUEUE)=
rev(tl(rev queue));
(*--Withdraw request from queue (2):-----*)
fun release_PS(queue:PACK_QUEUE,
ps_quantum:INT)=let
val r_pack=hd (rev queue) In
(#1 r_pack,#2 r_pack,#3 r_pack,
ran'random_val(), #5 r_pack,
#6 r_pack-ps_quantum) end;
```

As it was mentioned above, the main application of the software tool presented in the paper is modeling and evaluation of distributed Internet systems. To effectively model the Internet requests (or data packets) from the clients a separate token type (or token colour) has been proposed. The state of the system is determined by the number and distribution of the tokens representing data packets within the TCPN model. Each of the tokens representing a packet is a tuple `PACKAGE = (ID, PRT, START_TIME, PROB, AUTIL, RUTIL)` (compare with source code including color's definitions), where: `ID` - token identification (allowing token class definition etc.), `PRT` - priority, `START TIME` - time of a token occurrence in the system, `PROB` - probability value (used in token movement distribution in the net), `AUTIL` - absolute value of token utilization factor (for PS queue) and `RUTIL` - relative value of token utilization factor. Tokens have *timed* attribute scheduling them within places which are not queues.

While packets are being served, the components of a tuple are being modified. At the moment the given packet leaves the queueing system, a new `PROB` field value of `PACKAGE` tuple is being generated randomly (`release_PS` function). The value may be used to modify the load of individual branches in the queueing system model. Generally, the queueing system template is characterized by the following parameters: average tokens service time (`ps_ser_mean_time`), number of servers (number of tokens in `TIMERS` place) and service discipline (the TCPN's structure).

In the software tool developed, it is possible to construct queueing nets with queueing systems having PS and FIFO disciplines. These disciplines are the most commonly used for modeling Internet systems. Some our previous works include the rules of mapping TCPNs into queues of tokens scheduled according priorities [14, 15]. The presented templates have been tested on their compatibility with mathematical formulas determining the average queue length and service time as in [3].

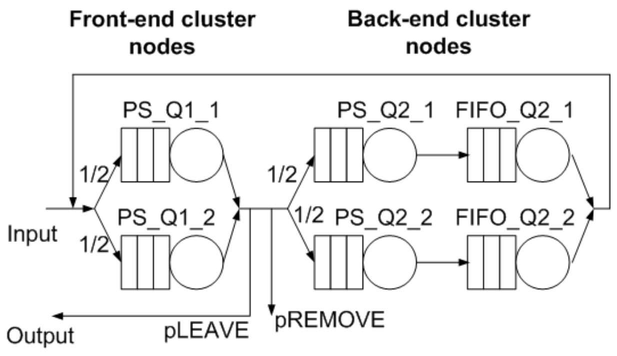


Figure 4: Queuing model of an example distributed Internet system environment.

4 Internet system modeling and analysis approach

Having a set TCPN based queueing systems models a systematic methodology of Internet system modeling and analysis may be proposed. Typically, modern Internet systems are composed of layers where each layer consists of a set of servers - a server cluster. The layers are dedicated for proper tasks and exchange requests between each other.

4.1 System structure modeling

To efficiently model typical Internet systems structures we proposed 3 modeling levels:

- superior - modeling of input process, transactions between layers and requests removal,
- layer - modeling of cluster structure,
- queue - modeling of queueing system.

To explain our approach to Internet system modeling a typical structure of distributed Internet system structure will be modeled and simulated. The example queueing model of the system informally sketched in fig. 1 is presented in fig. 4. It consists of two layers of server clusters and is constructed following the rules introduced in [6, 7, 8].

The front-end layer is responsible for presentation and processing of client requests. Nodes of this layer are modeled by PS queues. The next layer (back-end) implements system data handling. Nodes of this layer are modeled by using the serially connected PS and FIFO queues. The PS queue models the server processor and FIFO models the hard disc drive of server. Requests are sent to the system and then can be processed in both layers or removed after processing in front-end layer. The successfully processed requests are turned to the front-end layer and then send back to the customer.

Figure 5 shows the TCPN based model of above mentioned queueing network. The superior level of system description is presented in fig. 5a, whereas in fig. 5b and 5c detailed queueing systems topologies at each layer of the system are shown (compare fig. 4). Server cluster of the first layer (e.g. WWW servers; fig. 5b) as well as the second layer cluster (e.g. database; fig. 5c) have been demonstrated on the main page of TCPN net as substitution transitions: `front-end_cluster` and `back-end_cluster`.

On the superior level of system description (fig. 5a) we have also defined arrival process of the queueing network (`T0` transition with `TIMER0` and `COUNTER` places). `TIMER0` place and `T0` transition constitute a clock-like structure that produces tokens (requests) according to random, exponentially distributed frequency. These tokens are accumulated in a form of timed multiset in `PACKS1` place and then forwarded into the queueing-based model of the Internet system. When each token is being generated its creation time is memorized in the `PACKAGE` tuple. This makes it possible to conduct an off-line analysis of the model.

`T1` and `T5` as well as `T2` and `T3` transitions (compare fig. 5a) are in conflict. Execution of transition `T5` removes a serviced request from the net (modeling the system answer). If `T1` fires the request needs next transaction with the back-end system layer. Similarly, execution of transition `T2` removes a token from `T1` the net (modeling the possible loss of data packet). However, if `T3` fires, the data packet is transferred for processing in the second layer of the system. Guard functions connected to the mentioned transitions determine proportions between the tokens (packets) rejected and the ones remaining in the queueing net (in the example model approximately `30%` of the tokens are rejected or send back to the client).

Consequently, an executable (in a simulation sense) queueing network model is obtained. Tokens generated by arrival process are transferred in sequence by models of WWW server layer, by the part of the net that models loss (expiration) of some packets and by database layer. Provided that the system is balanced and has constant average arrival process, after some working time, the average values of the average queue length and response time are constant. Otherwise, their increase may occur.

The main parameters of the system modeled are the queue mean service time, the service time probability distribution function and the number of servicing units defined for each queueing system in the model. In the demonstrated model it has been assumed that queues belonging to a given layer have identical parameters.

4.2 System rerformance analysis

At this stage of our research it has been decided that simulation will be the main mechanism used to do analysis of the constructed model. In our simulations we applied the performance analysis. It allows collecting selected elements of the net state at the moment of an occurrence certain events during simulation. It has been assumed that in

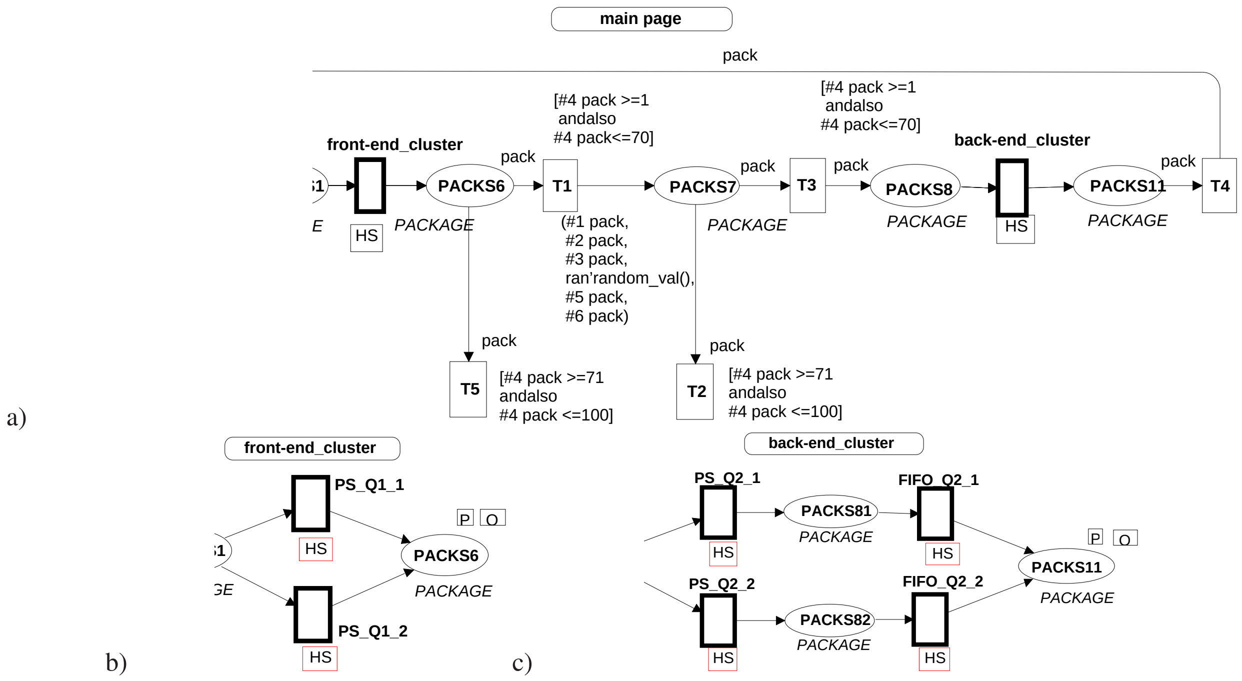


Figure 5: TCPNs based queueing system model: a) main page, b) front-end_cluster subpage and c) back-end_cluster subpage.

each of the model layers, queue lengths and response time will be monitored. Monitoring of the above mentioned parameters helps to determine whether the model of the system is balanced. Fig. 6 shows example plots obtained in the simulation of the discussed model.

The example experiment covered model time range from `0` to `100 000` time units. Fig. 6a shows the state of selected queue when the modeled system was balanced. Response time does not increase and remains around average value. System is regarded as balanced if the average queue lengths in all layers do not increase. In fig. 6b response time for unbalanced system was shown. The results concern the same layer as previously and identical time range for the simulation. It is clear that response time (fig. 6b) increase during the experiment. On the basis of the plot in fig. 6b, it can be concluded that the modeled system under the assumed external load would be overload and probably appropriate modifications in the structure of the system would be necessary. The software tool introduced in our paper makes it possible to estimate the performance of developing Internet system, to test and finally to help adjust preliminary design assumptions.

Having the possibility to capture the net's state during the simulation within a certain time interval, it can be possible to select model parameters in such a manner that they meet assumed time restrictions. Additionally, the parameters of real Internet system can be used to fit parameters of the constructed model.

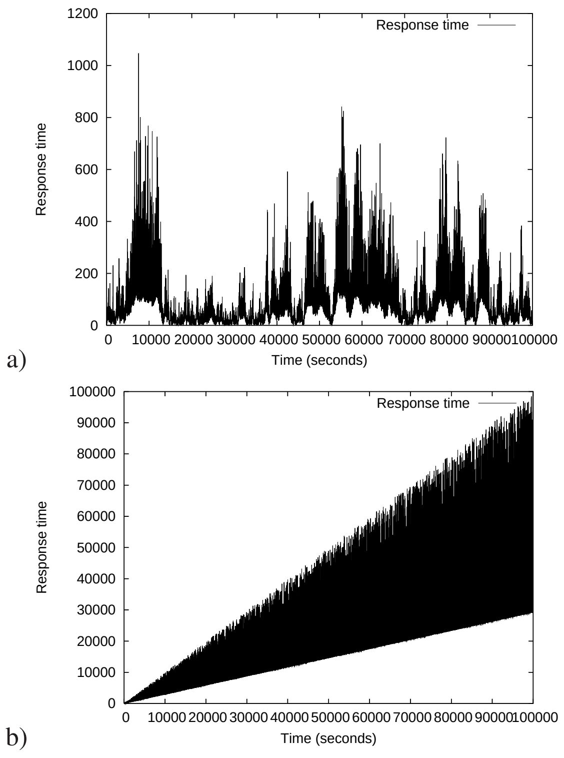


Figure 6: Sample system response time history: a) system balanced and b) system unbalanced.

5 Detailed vs. simplified Web cluster modeling

The worked out modeling and analysis methodology was used for construction and evaluation of several detailed models of architectures of distributed Internet systems. The analysis of the models were executed with the use of performance analysis tools for TCPN nets [20]. CSIM [17] simulating environment and experiments on real Internet system were used for TCPNs simulations evaluation. The overview of typical TCPNs based Internet system models analyzed so far can be found in [12].

In the remaining part of the paper we would like to discuss the following issue. During our research we proposed a *detailed TCPNs-based model of an example distributed Internet system.* The model makes it possible to express such phenomena as suspension of database service or database replication. The accuracy of the model was checked by comparison to a physical experimental computer cluster and a CSIM model. Obviously, the detailed modeling process requires the adequate knowledge of a database engine properties and consumes an amount of time. On the other hand, some authors (e.g. [18, 6, 8, 7]) ”suspends” the modeling process on the level similar to one presented in section 4. This modeling level seems to be more acceptable for a broaden amount of Web systems developers. Taking into consideration the aforementioned rationale we decided to compare three models of an example distributed Internet system and evaluate the level of inaccuracy that simpler models contribute.

5.1 Detailed model

In fig. 7 a detailed queueing model of the example system has been presented. It consists of two cluster layers of servers. Customer requests are sent to the chosen node of front-end cluster with `1/2` probability. Then they are placed in the queue to get service. The service in the service unit (processor) can be suspend many times, if for example the requests need the database access. When the database access occurs, requests are sent to back-end layer. Any request can also be removed following `pREMOVE` path. In case of sending to the database, a requests steers itself to service in one of back-end nodes with `1/2` probability. The service in the database service unit may be suspend if access to the input/output subsystem of the database storage device is necessary. Requests are returned to the database service unit after the storage device is served. This operation can be repeated many times. After finishing servicing in the back-end layer, requests are returned to front-end servers (`pDB`). Requests can visit back-end layer during processing many times. After finish servicing in front-end server requests are sent to the customer (`pLEAVE`).

If the necessity of the database replication appears the requests are sent to next database node (`pREP`). Unless none of these two situations appear the requests are send to front-end layer (`pDB`). Replication can also cause resig-

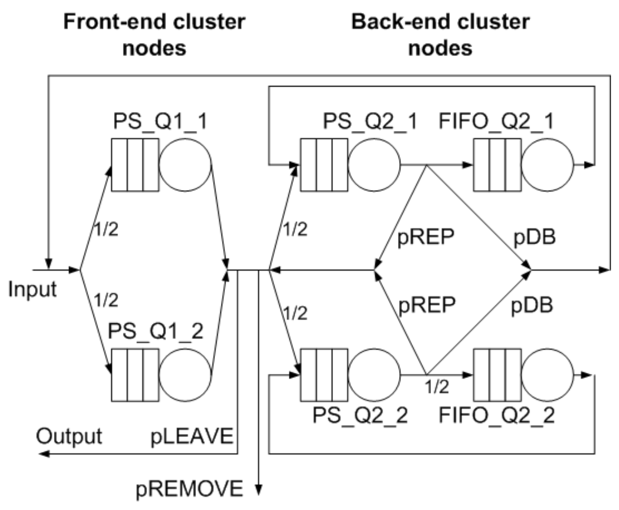


Figure 7: Detailed queueing model of the example distributed Internet system with suspension of transactions and database replication.

Probability	Probability values for model [%]
pREMOVE	30
pLEAVE	30
pDB	55
pREP	10

Table 1: The value of probabilities for model

nation from transaction. Both the replication and the rejection of realizing transaction are modeled in simplistic way as delivery of task to the next location of replication.

In this model:

- `PS_Q1_1\2` is a queue `PS` modeling element that processes in front-end layer,
- `PS_Q2_1\2` is a queue `PS` modeling element that processes in back-end layer,
- `FIFO_Q2_1\2` is a queue `FIFO` modeling device that stores data in back-end layer.

Tab. 1 includes the probabilities values for Internet requests distribution for considered model. The parameters assumed for discussed model are as follows:

- external load,
- the identical parameters of queues,
- request distribution probabilities,
- the number of nodes in experimental environment.

In fig. 8 the detailed TCPNs-based model of the back-end cluster is presented. The structure of the TCP net corresponds with the detailed queing model depicted in fig. 7. Its most essential properties are:

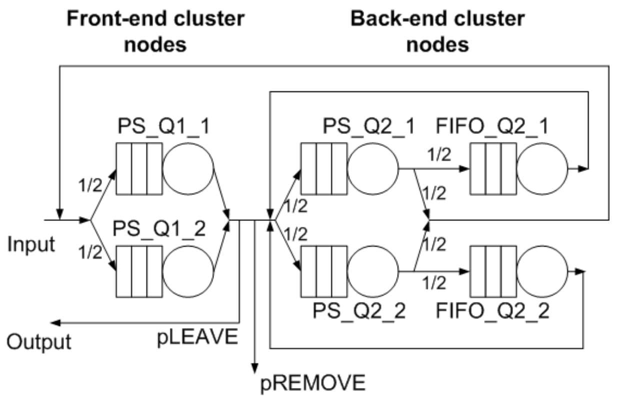


Figure 9: Semi-detailed queueing model of the example distributed Internet system.

- the possibility of directing tokens to any node (the location of replication),
- the return of tokens at the beginning of the layer,
- the realization of data synchronization in individual locations.

It is worth to notice that the complete detailed TCPNs-based model of the example Internet system can be derived as follows. The superior page of the model is identical at the one in fig. 5a. The front-end cluster model corresponds with the one from fig. 5b.

5.2 Semi-detailed model

In fig. 9 a semi-detailed queueing model of the example Internet system has been proposed. At this level of model simplification a database replication has been omitted. Still the service in the database service unit may be suspend if access to the input/output subsystem of the database storage device is necessary (compare subsection 5.1). A separate TCPNs-based semi-detailed model of the example system has been constructed, but due to limited size of the manuscript it has not been mentioned.

5.3 Simplified model

The so-called simplified model was thoroughly presented in section 4. Its queueing implementation can be seen in fig. 4, whereas its TCPNs-based implementation in fig. 5. The model does not takes into consideration suspension during the database access, nor database replicattion phenomenon.

5.4 Detailed model evaluation

Experimental environment and the CSIM packet were used to verify the detailed TCPNs-based model of the example system. The experimental system consisted of a net segment (100Mb/s), set of computers (Pentium 4, 2.8 GHz, 256 MB RAM) with Linux operating system (kernel 2.4.22) and Apache2 software (for WWW servers) as well as MySQL, version 4.0.15 (for database servers) [12].

The verification model was written by using CSIM simulator. This is a process oriented discreet event simulation package used with C or C++ compilers [17]. It provides libraries that a program written can be used in order to model a system and to simulate it. The models created by using CSIM [12] were based on presented queueing models (fig. 7) (similarly as TCPN models).

As a result we obtained the evaluated TCPNs based model of the Internet system discussed. The model made it possible to predict response time of the system developed. Tab. 2 shows the comparison of the detailed TCPNs-based model, detailed CSIM model of the example system, and a real experimental system. The system response times in each of the layer were measured during simulation (TCPNs model) and execution of the real environment. The experiments were conducted under different average external load values. In fig. 10 the response time of the layers during TCPNs-based model simulation under 100 [req./s] load are shown. It can be easily noticed, that system was balanced and the second layer may be the potential bottleneck due to higher response time.

5.5 Models evaluation

As we mentioned before, we treat the ”real” example system as the reference for the set of models to evaluate. In our experiments we assumed identical model parameters (e.g. queuing systems parameters, external load, probability of choosing different paths by the requests). Only the structures of modeled system were subsequently ”simplified”.

Tab. 3 includes the example response time statistics for the same external load for all types of models proposed. Tab. 4 includes the list of percentage of inaccuracy of the TCPNs-based and and CSIM models with respect to the real experimental system, respectively.

It can be easily noticed, that the simpler system model, the faster answer it gives. What is more, if we assume, that the detailed model is the ”closest” to the real system, an attempt to simplify it to the level discussed in section 5.3 without any modifications of queueing systems parameters is purposeless. Simultaneously, the semi-detailed model for some developers can have the quality sufficient to follow the real system behavior.

In fig. 11 an average response times history under the same external load for all types of the proposed models has been included. The average response times seem to oscillate around three constant values. It seems that a kind of equation showing the dependence between detailed, semi-detailed and simplified models may be derived.

6 Conclusions

It is still an open issue how to obtain an appropriate distributed Internet system. The demonstrated research results

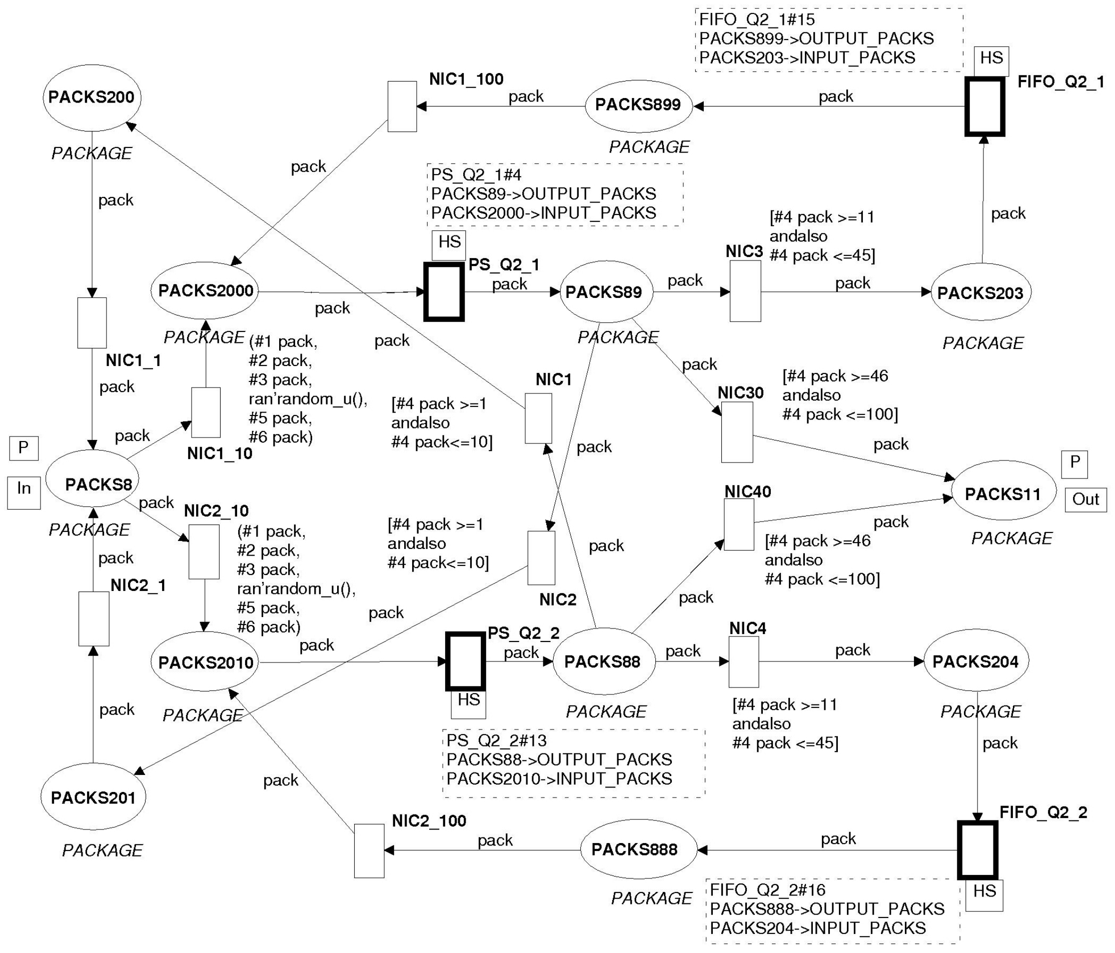


Figure 8: Back-end_cluster subpage of the detailed TCPNs-based queueing system model.

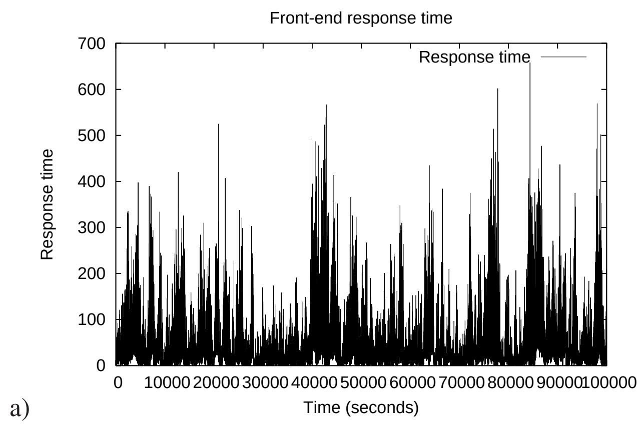
Load	Layer	TCPN Model	CSIM Model	Experiments	TCPN Model	CSIM Model
[req./s]		[ms]	[ms]	[ms]	Error [%]	Error [%]
100	front-end	49	50	36	26.5	28.0
100	back-end	567	598	409	27.5	31.0
300	front-end	701	743	579	17.4	22.1
300	back-end	813	798	648	20.4	39.7
500	front-end	1001	1109	921	8.0	16.4
500	back-end	1050	1083	973	7.3	10.0

Table 2: Layers response time for TCPNs model, CSIM model, and for experimental reference system

Layer	TCPN Simply [ms]	CSIM Simply [ms]	TCPN Semi-detailed [ms]	CSIM Semi-detailed [ms]	TCPN Detailed [ms]	CSIM Detailed [ms]	Experiments [ms]
front-end	93	114	123	159	167	197	153
back-end	115	135	157	192	222	257	201

Table 3: Layers response time for three TCPN, CSIM models and experimental results (the same load)

are an attempt to apply Queueing Theory (QT) and TCPNs formalism to the development of a software tool that can support distributed Internet system design. The idea of linking Queueing Nets Theory and Coloured Petri Nets was



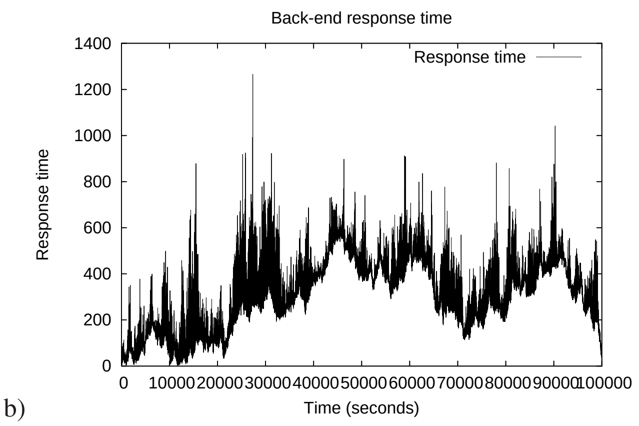


Figure 10: Detailed TCPNs-based model response time for layers a) - front-end layer, b) - back-end layer.

Layer	TCPN-Exp. Simply [%]	CSIM-Exp. Simply [%]	TCPN-Exp. Semi-detailed [%]	CSIM-Exp. Semi-detailed [%]	TCPN-Exp. Detailed [%]	CSIM-Exp. Detailed [%]
front-end	-57.73	-68.13	-22.40	-28.57	18.18	23.11
back-end	-63.41	-71.79	-34.89	-40.55	22.98	27.17

Table 4: Error - TCPN-CSIM-experiments

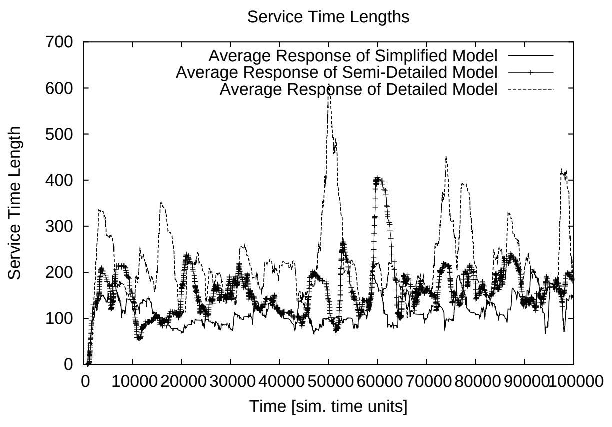


Figure 11: Average service times for Simplified, Semi-Detailed and Detailed models of the example system.

proposed previously by other authors in [6, 7]. However, in the presented approach queueing systems have been implemented using TCPNs formalism exclusively. As a consequence, alternative implementation of Coloured Queueing Petri Nets has been proposed. What was more, the rules of modeling and analysis of distributed Internet systems applying described net structures was introduced.

This paper deals with the problem of calculating performance values like the response time in distributed Internet systems environment. The values are calculated by using TCPNs. It is shown how the Coloured Petri Net model of a distributed Internet system is created with some of its data structures and functions, and gives an examples of system analysis. Finally, the paper discuses whether the detailed Internet system modeling brings substantial improvement in a real system mapping. The preliminary results show that probably adequate modification of queueing systems parameters can produce acceptable level of compatibility between ”simplified” models and the real systems.

Our future research will focus on dealing and analyzing another structures of distributed Internet systems using the software tool developed. It will be also dedicated to demonstrate compatibility of the models with the real systems. TCPN features such as tokens distinction will be of more extensive use. We will also make an attempt to create queueing model systems with defined token classes and consider a possibility to use state space analysis of TCPN net to determine properties of the system.

References

[1] F. Bause (1993) Queueing Petri Nets – a formalism for the combined qualititative and quantitative analysis of systems, *PNPM'93*, IEEE Press, pp. 14-23.

[2] V. Cardellini, E. Casalicchio, M. Colajanni (2002) The State of the Art in Locally Distributed Web-Server Systems, *ACM Computing Surveys, Vol. 34, No. 2*, ACM, pp.263–311.

[3] B. Filipowicz (2006) *Modeling and optimize queueing systems*, POLDEX, Krakow, (In Polish).

[4] K. Jensen (1996) *Coloured Petri Nets, Basic Concepts, Analysis Methods and Practical Use*, Springer.

[5] K. Jensen, L.M. Kristensen, L. Wells, (2007) Coloured Petri Nets and CPN Tools for modelling and validation of concurrent systems, *International Journal on Software Tools for Technology Transfer (STTT)*, Vol. 9, Numbers 3-4, pp. 213-254.

[6] S. Kounev (2006) *Performance Engineering of Distributed Component-Base Systems, Banchmarking, Modeling and Performance Prediction*, Shaker Verlag.

[7] S. Kounev (2006) Performance Modelling and Evaluation of Distributed Component–Based Systems Using Queuing Petri Nets, *IEEE Trans. on Soft. Eng., Vol 32. No. 7*, IEEE, pp. 486-502.

[8] S. Kounev, A. Buchmann, (2003) Performance Modeling and Evaluation of Large-Scale J2EE Applications, *Proceedings of the 29th Int. Conf. of the Comp. Meas. Group on Res. Manag. and Perf. Eval. of Enterprise Comp. Syst.*, ACM, Dallas, Texas, pp. 486-502.

[9] L. M. Kristensen, S. Christensen, K. Jensen, (1998) The Practitioner's Guide to Coloured Petri Nets *International Journal on Software Tools for Technology Transfer*, Vol. 2, pp. 98-132.

[10] B. Linstrom, L. Wells (1999) *Design/CPN Perf. Tool Manual*, CPN Group, Univ. of Aarhus, Denmark.

[11] T. Rak (2003) Model of Internet System Client Service, *Computer Science*, AGH Krakow, Vol. 5, pp. 55-65.

[12] T. Rak (2007) *The Modeling and Analysis of Interactive Internet Systems Realizing the Service of High-Frequency Offers*, PhD dissertation supervised by J. Werewka, Krakow, AGH, (In Polish).

[13] S. Samolej, T. Rak (2005) Time Properties of Internet Systems Modeling Using Coloured Petri Nets, *Real Time Systems* WKL, pp. 91-100, (In Polish).

[14] S. Samolej, T. Szmuc (2005) Time Constraints Modeling And Verification Using Timed Colored Petri Nets, *Real-Time Programming 2004*, Elsevier, pp. 127-132.

[15] S. Samolej, T. Szmuc (2005) TCPN-Based Tool for Timing Constraints Modelling and Validation, *Software Engineering, Evolution and Emerging Technologies, Volume 130*, IOS Press, pp. 194-205.

[16] S. Samolej, T. Szmuc (2008) Coloured Petri Nets based WWW Cluster Modeling and Development Method, *Inzynieria oprogramowania-od teorii do praktyki*, WKL, Warszawa, pp. 49-59 (In Polish).

[17] H. Schwetman (2001) CSIM19: A Powerfull Tool for Bilding System Models, *Proceedings Winter Simulation Conference*, IEEE, pp. 250-255.

[18] B. Urgaonkar, et. all (2005) An Analytical Model for Multi-tier Internet Service and Its Applications, *Proceedings of the ACM SIGMETRICS Inter. Conf. on Measurement and Model. of Comp. Sys.*, ACM, New York, pp. 291–302.

[19] L. Wells (2006) Performance analysis using CPN tools *Proc. of the 1st Inter. Conf. on Performance Evaluation Methodolgies and Tools, Pisa, Italy*, Article No. 59.

[20] L. Wells, et. all (2001) Simulation Based Performance Analysis of Web Servers, *Proc. of the 9th Inter. Workshop on Petri Nets and Performance Models*, IEEE, pp. 59–68.