e-Informatica Software Engineering Journal, Volume 3, Issue 1, 2009

Web–Server Systems HTCPNs-Based Development Tool Application in Load Balance Modelling

Slawomir Samolej*, Tomasz Szmuc**
*Department of Computer and Control Engineering, Rzeszów University of Technology
**Institute of Automatics, AGH University of Science and Technology
ssamolej@prz.edu.pl, tsz@agh.edu.pl

Abstract

A new software tool for web–server systems development is presented. The tool consist of a set of predefined Hierarchical Timed Coloured Petri Net (HTCPN) structures – patterns. The patterns make it possible to naturally construct typical and experimental server–systems structures. The preliminary patterns are executable queueing systems. A simulation based methodology of web–server model analysis and validation has been proposed. The paper focuses on presenting the construction of the software tool and its application for selected cluster–based web–servers load balancing strategies evaluation.

1. Introduction

Gradually, the Internet becomes the most important medium for conducting business, selling services and remote control of industrial processes. Typical modern software applications have a client–server logical structure where predominant role plays an Internet server offering data access or computation abilities for remote clients. The hardware of an Internet or web–server is now usually designed as a set of (locally) deployed computers. The computers are divided into some layers or clusters where each layer executes separate web–system task [4], [12], [24], [34], [39], [37], [5], [2], [22], [9], [29], [23]. This design approach makes it possible to distribute services among the nodes of a cluster and to improve the scalability of the system. Redundancy which intrinsically exists in such hardware structure provides higher system 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 server.

To improve the quality of service of web–server clusters two main research paths are followed. First, the software of individual web–server nodes is modified to offer average response time to dedicated classes of consumers [11], [18], [19]. Second, some distribution strategies of cluster nodes are investigated [4], [29] in conjunction with searching for load balancing policies for the nodes [6], [32], [39], [37], [5], [2], [22]. In several research projects reported in [12], [30], [34] load balancing algorithms and modified cluster node structures are analyzed together.

It is worth noticing that in some of abovementioned manuscripts searching for a solution of the problem goes together with searching for the adequate formal language to express the system developed [3], [12], [30], [32], [34], [39]. In [3], [32], [34], [39] Queueing Nets whereas in [30] Stochastic Petri Nets are applied for

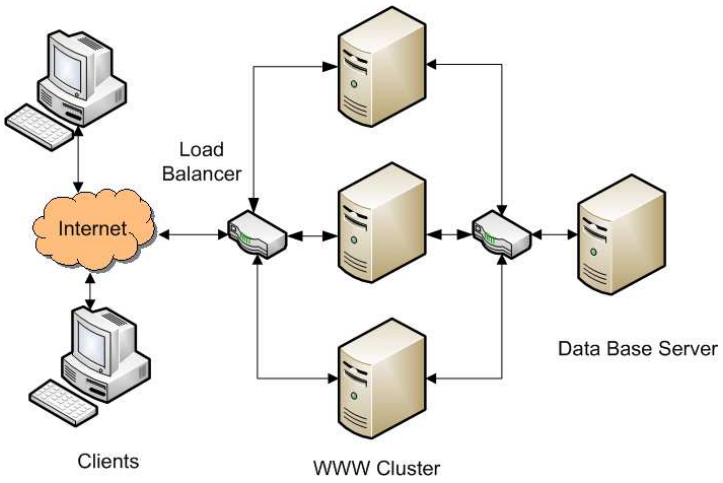

Figure 1. Example distributed cluster–based Internet system

system model construction and examination. However, the most mature and expressive language proposed for the web–cluster modelling seems to be Queueing Petri Nets (QPNs) [12]. The nets combine coloured and stochastic Petri nets with queueing systems [1] and consequently make it possible to model relatively complex web–server systems in a concise way. Moreover, there exists a software tool for the nets simulation [13]. The research results reported in [12] include a systematic approach to applying QPNs in distributed applications modelling and evaluation. The modelling process has been divided into following stages: system components and resources modelling, workload modelling, intercomponent interactions and processing steps modelling, and finally – model parameterization. The final QPNs based model can be executed and used for modelled system performance prediction.

The successful application of QPNs in web–cluster modelling become motivation to research reported in this paper. The aim of the research is to provide an alternative methodology and software tool for cluster–based hardware/software systems development. The main features of the methodology are as follows:

– The modelling language will be Hierarchical Timed Coloured Petri Nets (HTCPNs) [7],
– A set of so called HTCPNs design patterns (predefined net structures) will be prepared and validated to model typical web cluster components,
– The basic patterns will be executable models of queueing systems,
– A set of design rules will be provided to cope with the patterns during the system model creation,
– The final model will be an executable and analyzable Hierarchical Timed Coloured Petri Net,
– A well established Design/CPN and CPN Tools software toolkits will be used for the design patterns construction and validation,
– The toolkits will also be used as a platform for the web–server modelling and development,
– Performance analysis modules of the toolkits will be used for capturing and monitoring the state of the net during execution.

The choice of HTCPNs formalism as a modelling language comes from the following prerequisites. First, HTCPNs have an expression power comparable to QPNs. Second, the available software toolkits for HTCPNs composition and validation seem to be more popular than“SimQPN” [13]. Third, there exists a reach knowledge base of successful HTCPNs applications to modelling and validation of wide range software/hardware systems [7] including web–servers [24], [27], [36]. The rest named features of design methodology introduced in this paper results from both generally known capabilities of software toolkits for HTCPNs modelling and some previous experience gained by the authors in applica-

tion HTCPNs to real–time systems development [25], [26].

This paper is organized as follows. Section 2 describes some selected design patterns and rules of applying them to web–server cluster model construction. An example queueing system, web–server subsystem and top–level system models are presented. Then the simulation based HTCPNs models validation methods are discussed. Section 3 presents HTCPNs models of selected experimental and applied load balancing strategies for computer clusters. The load balancing models construction and some simulation results are discussed. Conclusions and future research program complete the paper.

It has been assumed that the reader is familiar with the basic principles of Hierarchical Timed Coloured Petri Nets theory [7], [8], [14]. All the Coloured Petri Nets in the paper have been edited and analysed using Design/CPN tool [21], [36]. Equivalent HTCPNs models may be developed using CPN Tools [8], [35] software toolkit.

2. Cluster Server Modelling Methodology

The main concept of the methodology lies in the definition of reusable timed coloured Petri nets structures (patterns) making it possible to compose web–server models in a systematic manner. The basic set of the patterns includes typical *queueing systems* TCPNs implementations, eg. –/M/PS/∞ , –/M/FIFO/∞ [24], [27]. *Packet distribution* TCPNs *patterns* constitute the next group of reusable blocks. They preliminary role is to provide some predefined web–server cluster substructures composed from the queueing systems. At this stage of subsystem modelling the queueing systems are represented as substitution transitions (compare [24], [27]). The separate models of system arrival processes are also the members of the group mentioned. The *packet distribution patterns* represented as substitution transitions are in turn used for the general *top–level system model* composition. As a result, the 3–level web–server model composition has been proposed. The top–level TCPN represents the general view of system components. The middle–level TCPNs structures represent the queueing systems interconnections. And the lowest level includes executable queueing systems implementations.

The modelling methodology assumes, that the actual state of the Internet requests servicing in the system can be monitored. Moreover, from the logical point of view the model of the server cluster is an open queueing network, so the requests are generated, serviced and finally removed from the system. As a result an important component of the software tool for server cluster development is *the logical representation of the requests*.

In the next subsections the following features of the modelling methodology will be explained in detail. First, the logical representation of Internet requests will be shown. Second, queueing system modelling rules will be explained. Third, an example cluster subsystem with an individual load–balancing strategy will be proposed. Fourth, Internet request generator structure will be examined. Fifth, top–level HTCPNs structure of an example cluster–server model will be shown. Finally, model analysis capabilities will be discussed.

2.1. Logical Request Representation

In the server–cluster modelling methodology that is introducing in the paper the structure of the HTCPN represents a hardvare/software architecture of web–server. Yet, the dynamics of the modelled system behavior is determined by state and allocation of tokens in the net structure. Two groups of tokens has been proposed for model construction. The first group consists of the so–called local tokens, that "live" in individual design patters. They provide local functions and data structures for the patterns. The second group of tokens represents Internet requests that are serviced in the system. They are transported throughout several cluster components. Their internal state carries data that may be used for timing and performance evaluation of the system modelled. As the tokens represent-

ing the requests have the predominant role in the modelling methodology, they structure will be explained in detail.

Each token representing an Internet request is a tuple

$$PACKAGE = (ID, PRT, START\_TIME, PROB, AUTIL, RUTIL),$$

where *ID* is a *request identifier*, *PRT* is a *request priority*, *START_TIME* is a *value of simulation time when the request is generated*, *PROB* is a *random value*, *AUTIL* is an *absolute request utilization value*, and *RUTIL* is a *relative request utilization value*. *Request identifier* makes it possible to give the request an unique number. *Request priority* is an integer value that may be taken into consideration when the requests are scheduled according priority driven strategy [11]. *START_TIME* parameter can store a simulation time value and can be used for the timing validation of the requests. *Absolute request utilization value*, and *relative request utilization value* are exploited in some queueing systems execution models (e.g. with processor sharing service).

2.2. Queueing System Models

The basic components of the software tool for web–server clusters development introduced in this paper are the executable queueing systems models. At the current state of the software tool construction the queueing systems models can have *FIFO*, *LIFO*, *processor sharing* or *priority based* service discipline. For each queue an arbitrary number of service units may be defined. Additionally, the basic queueing systems has been equipped with auxiliary components that are responsible for monitoring of internal states of the queue during its execution.

An example HTCPNs based –/1/FIFO/∞ queueing system model is shown in Fig. 2. The model is a HTCPNs subpage that can communicate with the parent page via *INPUT_PACKS*, *OUTPUT_PACKS* and *QL* port places. Request packets (that arrive through *INPUT_PACK* place) are placed into a queue structure within *PACK_QUEUE* place after *ADD_FIFO* transition execution. *TIMERS* place and *REMOVE_FIFO* transition constitute a clock–like structure and are used for modelling of duration of packet execution. When *REMOVE_FIFO* transition fires, then the first packet from the queue is withdrawn and directed to the service procedure.

The packets under service acquire the adequate time stamps generated according to the assumed service time random distribution function. The time stamps associated with the tokens prevent from using the packet tuples (the tokens) for any transition firing until the stated simulation time elapses (according to firing rules defined for HTCPNs [7]). The packets are treated as serviced when they can leave *OUTPUT_PACKS* place as their time stamps expired. The number of tokens in TIMERS place defines the quantity of queue servicing units in the system.

Main parameters that define the queueing system model dynamics are queue mean service time, service time probability distribution function and number of servicing units. Capacity of the queue is not now taken into consideration and theoretically may be unlimited.

For future applications the primary queueing system design pattern explained above has been equipped with an auxiliary "plug–in". *COUNT_QL* transition and *TIMER_QL*, *QL* and *COUNTER* places make it possible to measure the queue length and export the measured value to the parent CPNs page during the net execution. *TIMER_QL* place includes a timer token that can periodically enable the *COUNT_QL* transition. *QL* port place includes a token storing the last measured queue length and an individual number of a queueing system in the system. The *COUNTER* place includes a counter token used for the synchronization purpose.

2.3. Packet Distribution Models

Having a set of *queueing systems design patterns* some *packet distribution HTCPNs structures* may be proposed. In [24] a typical homogeneous multi–tier web–server structure pat-

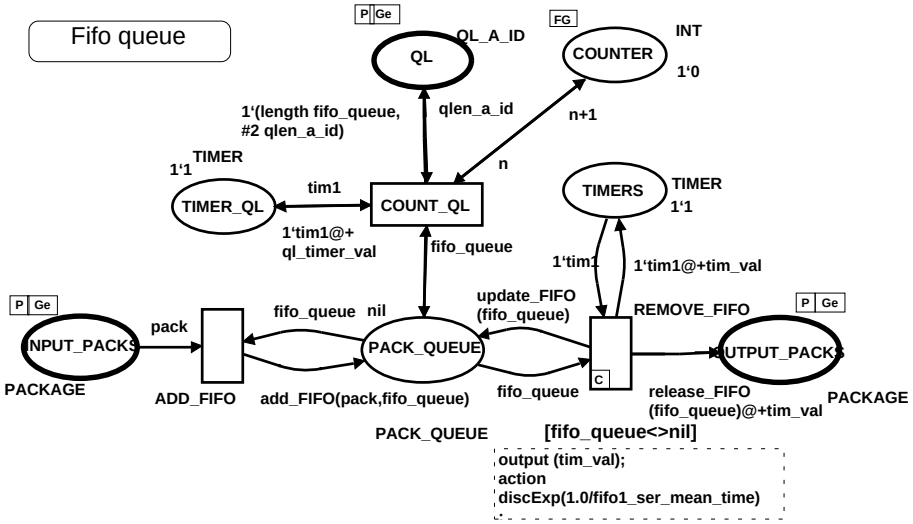

Figure 2. HTCPNs based –/1/FIFO/∞ queueing system model

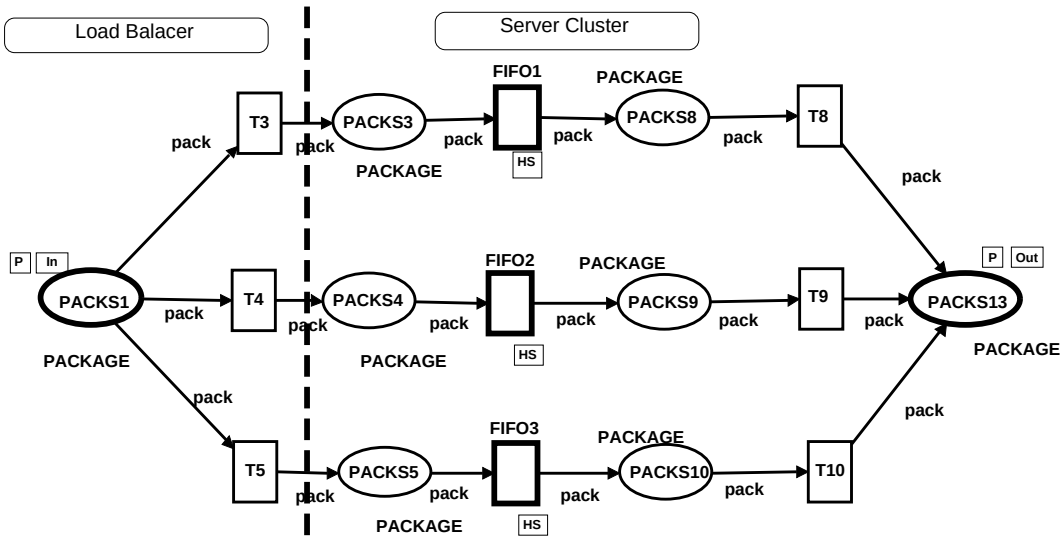

Figure 3. WWW cluster with stochastic load balancer

tern was examined, in [23] a detailed distributed database cluster model was proposed, whereas in [27] a preliminary version of server structure with feedback like admission control of Internet requests was introduced. The packet distribution patterns presented in this paper are also related to the load balancing in web–server cluster problem. The detailed discussion of some selected load balancing strategies models is included in section 3. In this section a simple WWW cluster model with stochastic packed distribution policy is concerned.

Figure 3 includes an example of cluster load–balancing HTCPNs model. The cluster consists of 3 computers (compare Fig. 1) represented as *FIFO1*...*FIFO3* substitution transitions, where each transition is attached to the corresponding *FIFO* queueing pattern. The Internet requests serviced by the cluster arrive through *PACKS1* port place. A *load balancer* decides where the currently acquired request should be send. When a token arrives in *PACKS1* place, transitions *T3*...*T5* are in conflict. According to CPN properties, a transition is randomly chosen for firing. Consequently, the stochastic packet distribution policy is naturally modelled.

2.4. Request Generator Model

According to one of main assumptions of the web–server cluster modelling methodology presented in this paper, the system model can be treated as an open queueing network. Consequently, the crucial model component must be a network arrival process simulating the Internet service requests that are sent to the server.

Figure 4 shows an example HTCPNs subpage that models a typical Internet request generator. The core of the packet generator is a clock composed from *TIMER0* place and *T0* transition. The code segment attached to the *T0* transition produces values of time–stamps for tokens stored in *TIMER0* place. The values are defined by the defined probability function. As a result the Internet requests appear into *PACKS1* place at random moments in simulation time. The frequency at which tokens appear in *PACKS1* place is determined by the mentioned above distribution function. *PACKS1* place has a port place status and thereafter tokens appearing in it can be consumed by other model components (e.g. server cluster model).

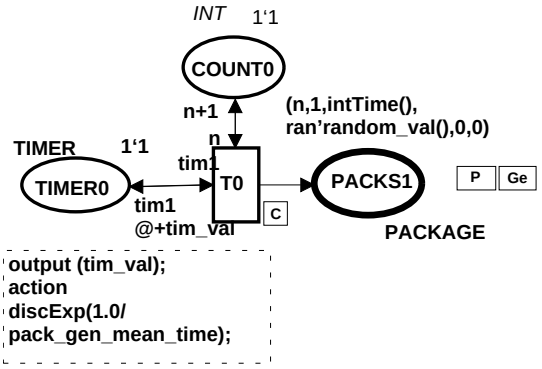

Figure 4. Web–server arrival process model

The Internet request frequency can have any standard probability distribution function or can be individually constructed as it was proposed in [36].

2.5. Example Top–Level Multi–Tier Server Model

Having the adequate set of design patterns, a wide area of server cluster architectures can be modelled and tested at the early stage of development process. At the top–level modelling process each of the main components of the system can be represented as a HTCPNs substitution transition. The modelling methodology presented in the paper suggest that at the top–level model construction the arrival process and main server cluster layers should be highlighted. After that each of the main components (main substitution transition) should be decomposed into an adequate packed distribution subpage, were under some of transitions queueing system models will be attached. It is easily to notice that a typical top–down modelling approach of software/hardware system modelling has been adapted in the web server modelling methodology proposed in the paper.

Figure 5 includes an example top–level HTCPN model of cluster–based server (compare Fig. 1) that follows the abovementioned modelling development rules. The HTCPN in Fig. 5 consists of 3 substitution transitions. *Input_Procs* transition represents the arrival process for the server cluster, whereas *WWW_Server_Cluster* transition represents the first–layer of multi–tier web–server, and finally *DataBaseServer* transition represents the data base server.

The modelling process can be easily extended by attaching the request generator model as in section 2.4 under the *Input_Procs* transition and by attaching the WWW cluster model with load balancing module as in section 2.3 under *WWW_Server_Cluster* transition. The final executable model can be acquired by attaching FIFO design patterns under $FIFO1$, $FIFO2$ and $FIFO3$ transitions in the load balancing module (compare sections 2.2 and 2.3). A separate model should be proposed for the packet distribution and queueing models layers of the data base server.

2.6. Model Validation Capabilities

Typical elements of HTCPNs modelling software tools are performance evaluation routines, e.g.: [16], [35] . The routines make it possible to capture the state of dedicated tokens or places

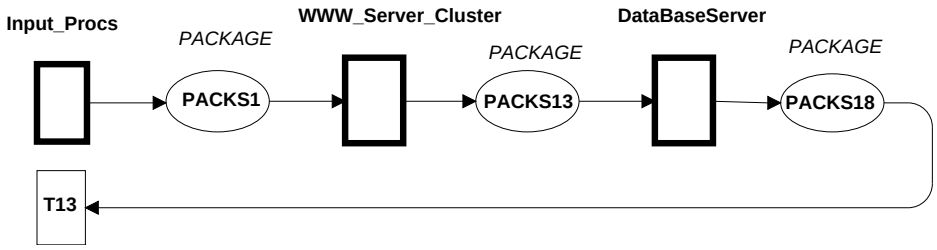

Figure 5. Example top–level multi–tier server model

during the HTCPN execution. A special kind of log files showing the changes in the state of HTCPN can be received and analyzed offline.

At the currently reported version of web–server cluster modelling and analysis software tool, queue lengths and service time lengths can be stored during the model execution. Detecting the queue lengths seems to be the most natural load measure available in typical software systems. The service time lengths are measurable in the proposed modelling method because of a special kind *PACKAGE* type tokens construction (compare section 2.1). The tokens "remember" the simulation time at which their appear in the cluster and thereafter the time at each state of their service may be captured. In real systems the service time is a predominant quality of service parameter for performance evaluation.

The performance analysis of models of web servers constructed according the proposed in the paper methodology can be applied in the following domains.

First, the system instability may be easily detected. The stable or balanced queueing system in a steady state has an approximately constans average queue length and correspondingly average service time. On the contrary, when the arrival process is to intensive for the queueing systems to serve, both queue lengths and service times increase. This kind of analysis is possible when there are no limitations for queue lengths in the proposed modelling method. Fig. 6 shows the queue lengths (Fig. 6 (left)) and service time lengths (Fig. 6 (right)) when the considered web server cluster model is permanently overloaded.

Second, the average values of queueing system systems parameters such as average queue lengths and average servicing times for the balanced model can be estimated. Provided that the arrival process model and the server nodes models parameters are acquired from the real devices as in [18], [30], [34], [36], the software model can be used for derivation the system properties under different load conditions. In the Fig. 7 queue lengths (Fig. 7 (left)) and service times (Fig. 7 (right)) under stable system execution are shown. The cluster had a heterogeneous structure, where server 2 (FIFO2 model) had 4 times lower performance. FIFO1 and FIFO3 average queue length was 1.7, whereas FIFO3 queue length was 4.4. The average service time for FIFO1 and FIFO3 cluster nodes was 811 time units whereas for FIFO2 was 7471 time units.

Third, some individual properties of cluster node structures or load balancing strategies may be observed. Some selected load balancing algorithms properties derived from simulation experiments will be discussed in section 3.

3. Example of Load Balancing Strategies Evaluation

Load balancing is an important issue in parallel and distributed systems. In traditional computation systems load balancing procedures were used to distribute the computation task among system nodes. It improved the general system utilisation and usually led to faster processing. In recent years, the load balancing algorithms elaborated for general parallel and distributed systems [31] were naturally re-applied in the emerging locally distributed Internet or web systems. The first load balancing strategies successively applied in Internet systems were static random distribution policy [28] and static modulus-based round-robin

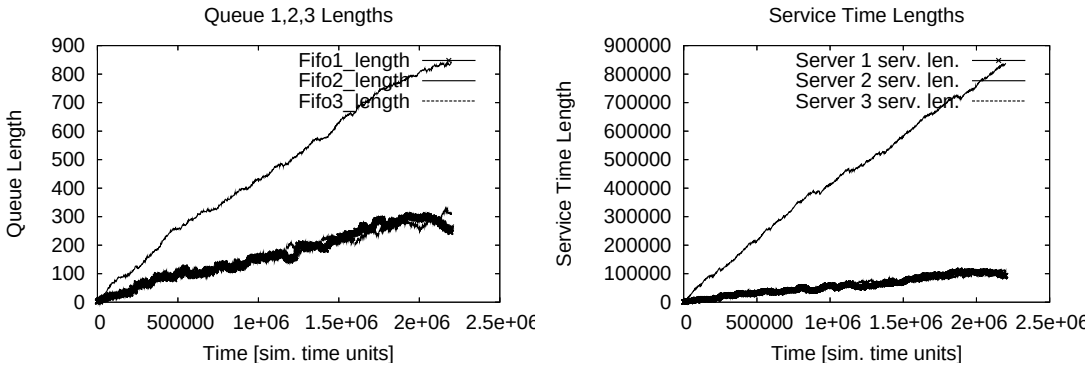

Figure 6. Queue lengths (left) and service times (right) under overload condition

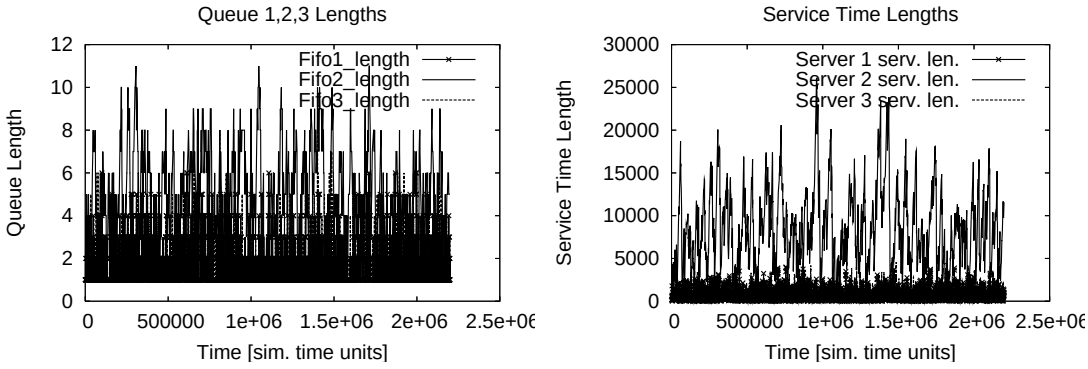

Figure 7. Queue lengths (left) and service times (right) under stable system execution

policy [15]. There were described in the paper [38] successful implementations of round-robin, weighted round-robin, least-connection and weight least-connection load balancing strategies in Linux Virtual Server [17] . The abovementioned strategies are now standard algorithms in commercial load balancing solutions as mentioned in [20], [33].

The rapid development of web–server oriented load balancing policies was tentatively systematised in [5], [2]. Paper [5] classifies distributed web-server architectures regarding the entity which distributes requests among servers. It defines 4 approaches of request distribution: client–based, DNS–based, dispatcher–based, and server–based. In [2] an attempt of simulation based on comparison of selected load balancing algorithms such as "round robin", "least connection first", "round-trip" and "Xmitbyte" was carried out.

During last few years research has focused on the so-called dynamic load balancing strategies for web–oriented systems. Generally, the dynamic load balancing polices use some kind of feedback information from the cluster nodes to redirect the incoming request among the nodes. In [30] the so called "fewest server processes first" and "extended fewest server processes first" dynamic load balancing policies were compared. The fewest server processes first policy concept is (in our opinion) comparable to least–connection policy. In both algorithms the least loaded server gets the next incoming request. The "extension" of the preliminary algorithm lies in the fact that the request can have priorities. The paper presents high-level Petri net approach to efficiency analysis of the policies in different priority levels scenarios. A dynamic web system load balancing policy presented in [37] (AdaptLoad policy) adopts the load of the servers according to size of documents requested. The policy builds a discrete data histogram encoding empirical size distribution of batches of K requests as they ar-

rive in the system. Each server "offers" files within a certain range of size. The range depends of "popularity" of the files derived from the histogram. The paper presents simulation results of the policy behaviour under historical load conditions. Paper [6] at first discusses such so called nonprediction–based load balancing techiques as "first fit", "stream–based mapping" and "adaptive load sharing" correspondingly. The techniques are examined with respect to possible application in multimedia applications. The prediction–based load balancing techniques as "least load first", "prediction–based least load first", "adaptive partition" and "prediction–based adaptive partition" are introduced and experimentally evaluated. In [22] a sum of weighted factors such as CPU usage, memory usage, number of processors, number of I/O operations, amount of free local storage and network I/O usage are taken into consideration to compute the load of a cluster node. The load of the node may then be applied to a load balancing policy.

At the current state of the development of the HTCPNs–based tool, some selected load balancing HTCPNs templates were modelled. Three of the most widely applied polices such as "random", "round-robin" and "fewest server processes first" were implemented. Additionally one experimental–"adaptive load sharing" policy was chosen for the implementation, because as it was claimed in [6], this policy offers reasonable balance between the throughput and out–of–order departures of the external requests. In the following subsections the HTCPNs–based models of the mentioned load balancing policies will be presented. The final subsection will include some simulation results of the HTCPNs–based load balancing algorithms. The model of the simplest– "random" load balancing policy was presented in subsection 2.3.

3.1. Round-robin Load Balancing Policy Model

Figure 8 presents HTCPNs–based model of the computer cluster similar to the cluster model in Fig. 3. The cluster consists of 3 computers servicing requests incoming via $PACKS1$ port place. The incoming Internet requests are redistributed among the cluster nodes according to round-robin load balancing policy. The model of the policy works as follows. Each incoming packet "passes" $T2$ transition and after the transition firing the forth element of the tuple modelling the requests (see subsection 2.1) is modified. The element includes a number of the server where the packet will be serviced. Guard functions attached to $T3$, $T4$, $T5$ transitions "check" the fourth element of each packed model and "pass" the related requests. The presented load balancing policy model can be easily extended to "weighted round-robin" policy by extending the numbers generated for the forth element of the packet tuple and by the corresponding modifications of the guards.

3.2. Fewest Server Processes First Load Balancing Policy Model

Figure 9 includes HTCPNs–based model of the computer cluster similar to the cluster model in Fig. 3 and Fig. 8. The incoming Internet requests are redistributed among the cluster nodes according fewest serwer processes first load balancing policy. The the model of the policy works as follows. During the model execution, the lengths of the queues in the queueing systems modelling servers are periodically monitored. The monitoring is possible due to appropriate construction of queueing systems models (compare subsection 2.2). *QL1*, *QL2*, and *QL3* places include the current values of queue's lengths. The queue's lengths are compared during *BALANCE* transition execution and *FEWEST* place acquires a number of the serwer which serves the fewest number of requests (the server with the shortest request queue). Guard functions associated to *T3*, *T4*, and *T5* transitions "open" (for the incoming requests) only this branch of the cluster which includes the least loaded server. The frequency of the queue's lengths measurement can be adjusted to derive the balance between additional system load caused by the measurement and the

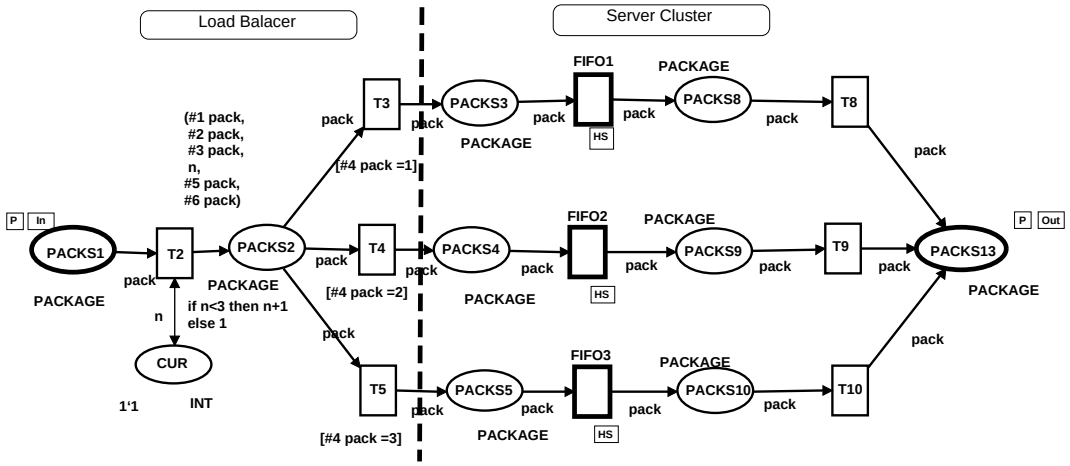

Figure 8. WWW cluster with round-robin load balancing policy

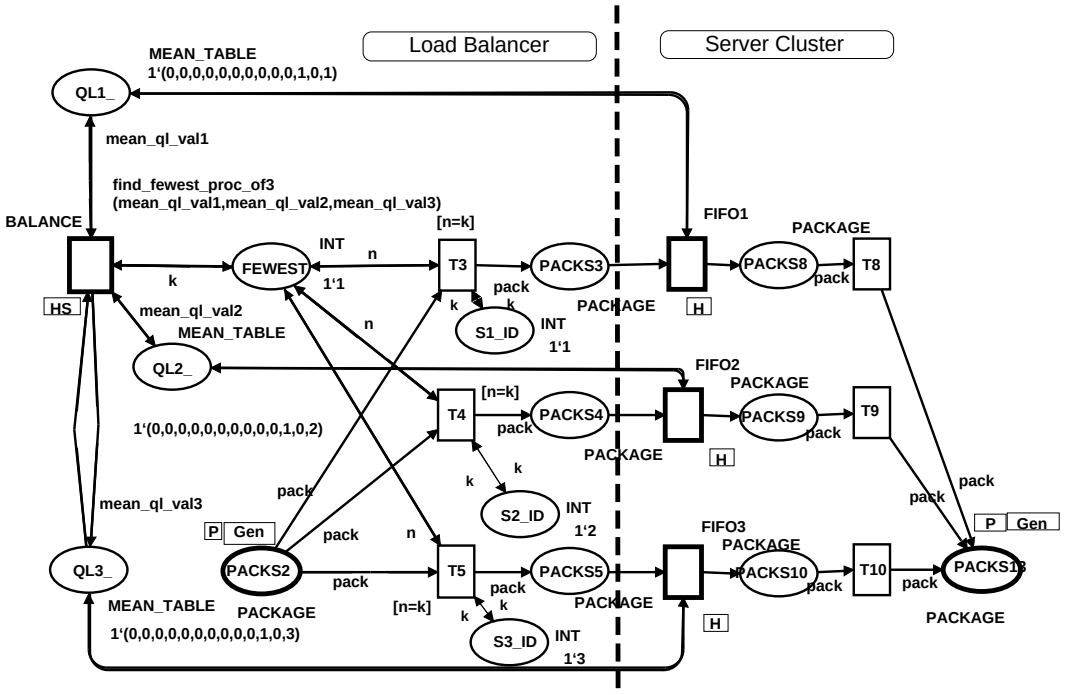

Figure 9. WWW cluster with fewest server processes first load balancing policy

quality of the balance process. It is possible to define digital filters to "smooth out" the queue length "signal".

3.3. Adaptive Load Sharing – Load Balancing Policy Model

Fig. 10 presents HTCPNs–based model of the computer cluster where the incoming Internet requests are redistributed among the cluster nodes according to adaptive load sharing load balancing policy inspired by [6], [10]. The model of the policy works as follows. During the system model execution lengths of the queues in the queueing systems modelling servers are periodically monitored. *QL1*, *QL2*, and *QL3* places include the current values of queue's lengths. During execution of the *BALANCE* transition the utilisation of each node of the cluster is calculated. Depending on the utilisation values some ranges of amounts of the Internet requests that each server may serve are calculated. The cal-

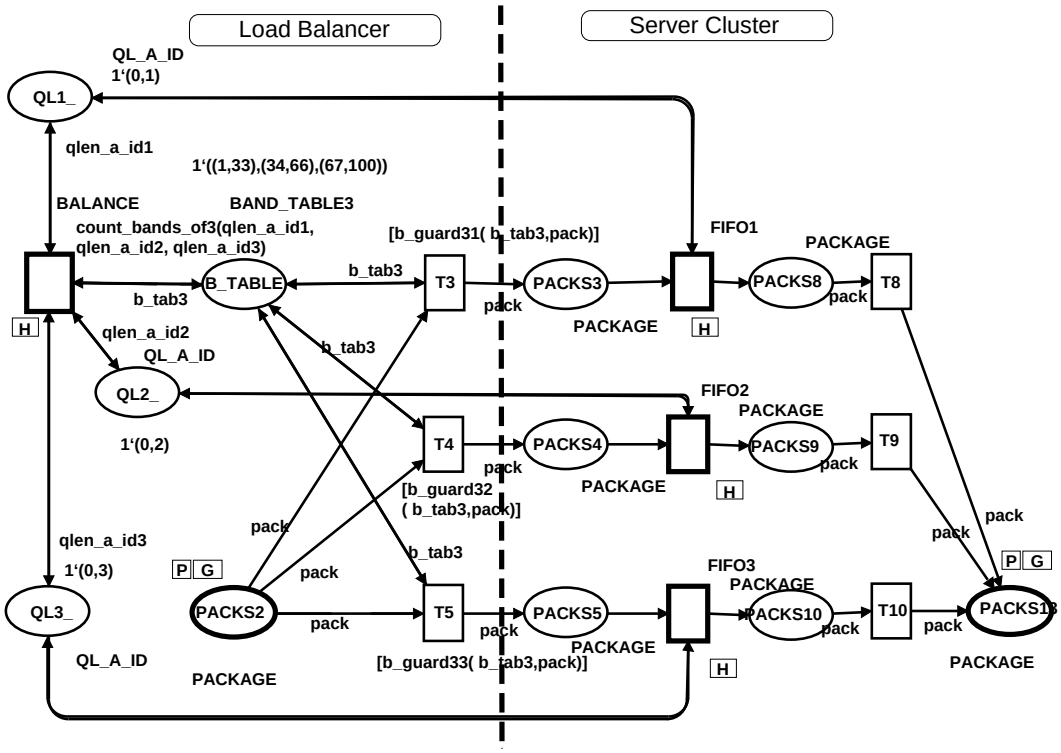

Figure 10. WWW cluster with adaptive load sharing load balancing policy

culated ranges are stored in *B_TABLE* place. Generally, servers having lower utilisation values will be given more chances to acquire the Internet requests in the future. Guard functions associated to *T3*, *T4*, and *T5* transitions can be understand as "valves" that adjust the amounts of the requests to be passed through to the servers according the range table stored in *B_TABLE* place. The frequency of the queue's lengths measurement can be adjusted to derive the balance between additional system load caused by the measurement and the quality of the balance process. It is possible to define digital filters to "smooth out" the queue length "signal".

3.4. Simulation Based on Load Balancing Policies Evaluation

For the above mentioned models of load balancing policies a set of simulation analysies was carried out. In Figure 11 results of 2 different simulations are shown. Figure 11 (left) includes queue lengths of 3 balanced servers where load balancing policy followed round-robin algorithm. In the (right) simulation a performance degradation of server 2 in 200000 time units was modelled. It can be easily seen, that the load balancer does not "notice" the performance degradation. The requests directed to second server are serviced later than the others.

The queue lengths for clusters where fewest server processes first and adaptive load sharing load balancing policies are coping with server 2 performance degradation are shown in Fig. 12. Both polices (fewest server processes first – Fig. 12 (left), adaptive load sharing – Fig. 12 (right)) "reconfigured" the loads for the cluster nodes and managed to keep the average queue lengths for all cluster nodes at the same level. The simulation experiment proved that dynamic load balancing polices better cope with dynamic changes during system execution. However, it must be noticed that the dynamic load balancing policies need some feedback information collected from the nodes of the cluster. To fulfill such requirement both modern load balancers and cluster nodes (e.g. WWW servers) software must be modified. Additionally the feedback data collection can increase the load of the system.

Figure 13 shows more possibilities for design of dynamic load–balancing policies. The simula-

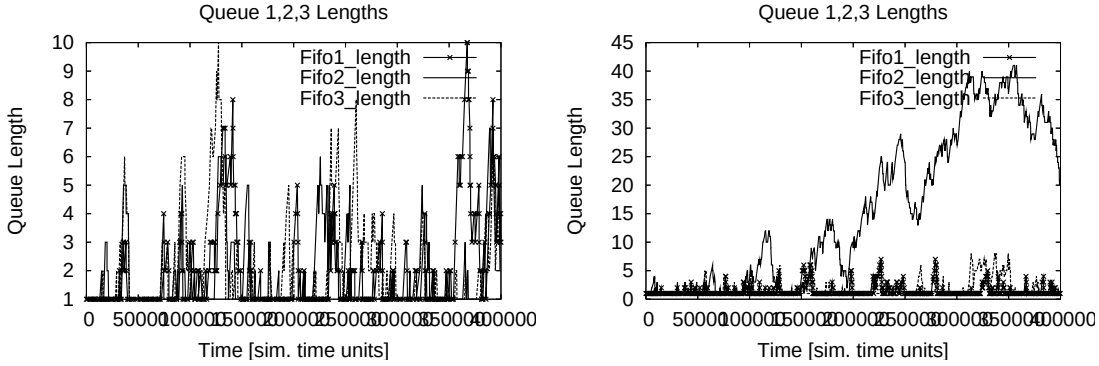

Figure 11. Queue lengths under round-robin load balancing policy (left) balanced (right) unbalanced after server 2 performance reduction

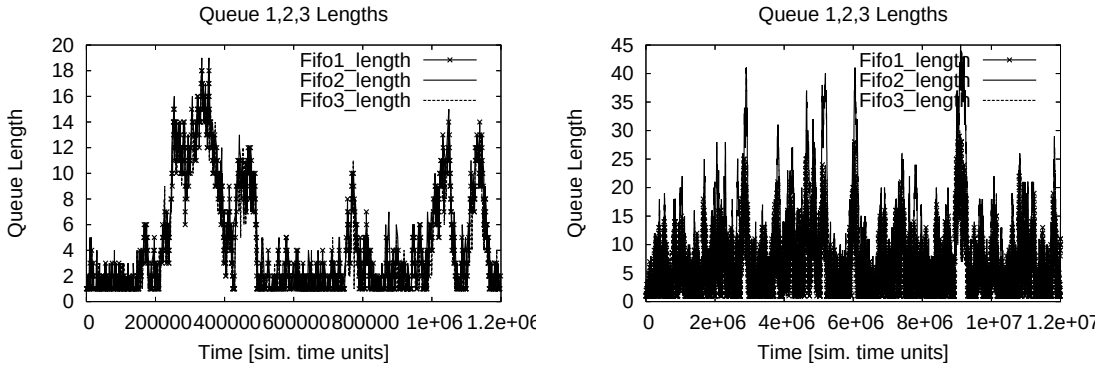

Figure 12. Queue lengths under fewest server processes first (left) and adaptive load sharing (right) load balancing policy after server 2 performance reduction

tion results show that the application of some feedback data to cluster state modification may cause system's behaviour similar to control–loop systems. In Figure 13 (left) the queue lengths oscillations caused by an inadequate data collection frequency may be noticed. The system in Fig. 13 (right) seems to "suffer" from the high sensibility that may in consequence lead to the instability.

4. Conclusions and Future Research

The first part of the paper introduces the HTCPNs–based software tool providing support for development and validation of web–server clusters executable models. The main concept of the tool lies in the definition of reusable HTCPNs structures (patterns) involving typical components of cluster–based server structures. The preliminary patterns are executable models of typical queueing systems. The queueing systems templates may be arranged into server cluster subsystems by means of packet distribution patterns. Finally, the subsystems patterns may be naturally used for top level system modelling, where individual substitution transitions "hide" the main components of the system. The final model is a hierarchical timed coloured Petri net. Simulation and performance analysis are the predominant methods that can be applied for the model validation. Queueing systems templates was checked whether they meet theoretically derived performance functions. The analysis of HTCPNs simulation reports enables to predict the load of the modelled system under the certain arrival request stream; to detect the stability of the system; to test a new algorithms for Internet requests redirection and for their service within cluster structures.

The second part of the paper includes the review of recently published research results con-

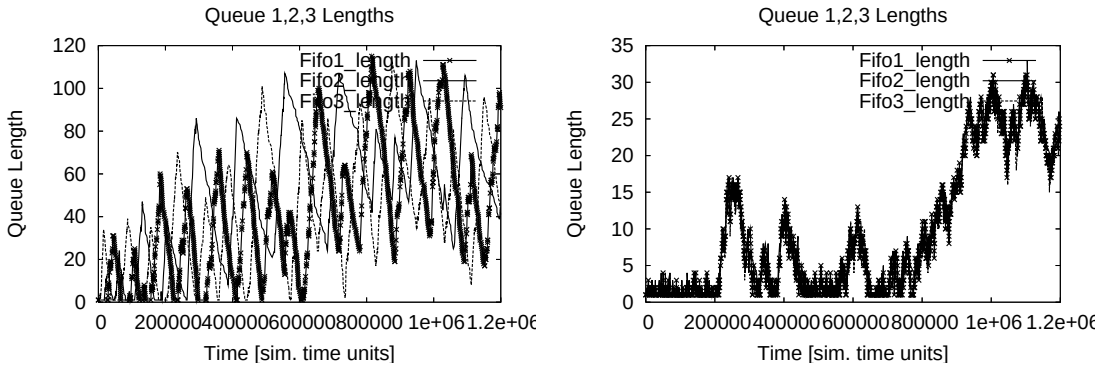

Figure 13. Queue lengths under fewest server processes first load balancing policy: queue lengths oscillation (left), high system sensitivity (right)

cerning application load balancing policies in Internet systems. Subsequently, the HTCPNs based models of some selected polices have been proposed. The most popular load balancing policies models such as "random", "round robin", and "fewest server processes first" as well as one experimental–"adaptive load sharing" have been applied and evaluated. The worked out HTCPNs structures become the integrated modules of the HTCPNs based software tool presented in the first part of the paper.

Currently, the software tool described in the paper can be applied for a limited web–server cluster structures modelling and validation. Thereafter the main stream of author's future research will concentrate on developing next web–server node structures models. This may result in following advantages. First, an open library of already proposed web–server cluster structures could be created and applied by the future web–server developers. Second, some new solutions for distributed web–server systems may be proposed and validated.

References

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

[2] H. Bryhni, E. Klovning, and O. Kure. A comparison of load balancing techniques for scalable web servers. *IEEE Network*, Volume 14(4):58–64, Jul./Aug. 2000.

[3] J. Cao, M. Andersson, C. Nyberg, and M. Khil. Web server performance modeling using an M/G/1/K*PS queue. In *CT 2003, 10th International Conference on Telecommunications*, pages 1501–1506. IEEE, 2003.

[4] V. Cardellini, E. Casalicchio, and M. Colajanni. The state of the art in locally distributed web-server systems. *ACM Computing Surveys*, Volume 34(2):263–311, June 2002.

[5] V. Cardellini, M. Colajanni, and P. Yu. Dynamic load balancing on web-server systems. *IEEE Internet Computing*, Volume 3(3):28–39, May/June 1999.

[6] J. Guo and L. Bhuyan. Load balancing in a cluster-based web server for multimedia applications. *IEEE Transactions on Parallel and Distributed Systems*, Volume 17(11):1321–1334, 2006.

[7] K. Jensen. *Coloured Petri Nets, Basic Concepts, Analysis Methods and Practical Use*, volume I-III. Springer, 1996.

[8] K. Jensen, L. Kristensen, and L. Wells. Coloured Petri Nets and CPN tools for modelling and validation of concurrent systems. *International Journal on Software Tools for Technology Transfer (STTT)*, Volume 9(3-4):213–254, 2007.

[9] Y. Ji and I. S. Ko. A design of the simulator for web-based load balancing. *Springer LNCS*, Volume 4496/2007:884–891, July 2007.

[10] L. Kencl and J.-Y. L. Boudec. Adaptive load sharing for network processors. In *Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings*, pages 545–554. IEEE, 2002.

[11] D. Kim, S. Lee, S. Han, and A. Abraham. Improving web services performance using priority allocation method. In *Proc. Of International Conference on Next Generation Web Services Practices*, pages 201–206. IEEE, 2005.

[12] S. Konunev. Performance modelling and evaluation of distributed component–based systems using Queuing Petri Nets. *IEEE Transactions on Software Engineering*, Volume 32(7):486–502, 2006.

[13] S. Kounev and A. Buchmann. SimQPN–a tool and methodology for analyzing Queueing Petri Net models by means of simulation. *Performance Evaluation*, Volume 36(4–5):364–394, 2006.

[14] M. Kristensen, S. Christensen, and K. Jensen. The practitioner's guide to coloured Petri Nets. *International Journal on Software Tools for Technology Transfer (STTT)*, Volume 2:98–132, 1998.

[15] T. T. Kwan, R. E. McGrath, and D. A. Reed. Ncsa's world wide web server: Design and performance. *Computer*, Volume 28(11):68–74, Nov. 1995.

[16] B. Linstrom and L. Wells. *Design/CPN Performance Tool Manual*. CPN Group, Univ. of Aarhus, Denmark, 1999.

[17] Linux virtual server project. http://www.linuxvirtualserver.org/.

[18] X. Liu, L. Sha, Y. Diao, S. Froehlich, J. L. Hellerstein, and S. Parekh. Online response time optimization of Apache web server. In *IWQoS 2003: 11th International Workshop*, pages 461–478. Springer, 2003. LNCS.

[19] X. Liu, R. Zheng, J. Heo, Q. Wang, and L. Sha. Timing performance control in web server systems utilizing server internal state information. In *Proc. of the Joint Internat. Conf. on Autonomic and Autonomous Systems and International Conference on Networking and Services*, page 75. IEEE, 2005.

[20] loadbalancers.org. http://loadbalancer.org/.

[21] Meta Software Corporation. *Design/CPN Reference Manual for X-Windows*, 1993.

[22] G. Park, B. Gu, J. Heo, S. Yi, J. Han, J. Park, H. Min, X. Piao, Y. Cho, C. W. Park, H. J. Chung, B. Lee, and S. Lee. Adaptive load balancing mechanism for server cluster. *Springer LNCS*, Volume 3983/2006:549–557, May 2006.

[23] T. Rak and S. Samolej. Distributed internet systems modeling using tcpns. In *Proc. of International Multiconference on Computer Science and Information Technology*, pages 559–566. IEEE, 2008.

[24] S. Samolej and T. Rak. Timing properties of internet systems modelling using Coloured Petri Nets. In *Systemy czasu rzeczywistego – Kierunki badań i rozwoju*, pages 91–100. Wydawnictwa Komunikacji i Łączności, 2005. In Polish.

[25] S. Samolej and T. Szmuc. TCPN–based tool for timing constraints modelling and validation. In *Software Engineering: Evolution and Emerging Technologies, Volume 130 Frontiers in Artificial Intelligence and Applications*, pages 194–205. IOS Press, 2005.

[26] S. Samolej and T. Szmuc. Time constraints modeling and verification using Timed Colored Petri Nets. In *Real–Time Programming 2004*, pages 127–132. Elsevier, 2005.

[27] S. Samolej and T. Szmuc. Dedicated internet systems design using Timed Coloured Petri Nets. In *Systemy czasu rzeczywistego – Metody i zastosowania*, pages 87–96. Wydawnictwa Komunikacji i Łączności, 2007. In Polish.

[28] M. Satyanarayanan. Scalable, secure, and highly available distributed file access. *Computer*, Volume 23(5):9–18, 20–21, May 1990.

[29] T. Schroeder, S. Goddard, and B. Ramamurthy. Scalable web server clustering technologies. *IEEE Network*, Volume 14(4):38–45, May/June 2000.

[30] Z. Shan, C. Lin, D. Marinecu, and Y. Yang. Modelling and performance analysis of QoS–aware load balancing of web–server clusters. *Computer Networks*, Volume 40:235–256, 2002.

[31] B. A. Shirazi, A. R. Hurson, and K. M. Kavi. *Scheduling and Load Balancing in Parallel and Distributed Systems*. Wiley-IEEE Computer Society Press, April 1995.

[32] F. Spies. Modeling of optimal load balancing strategy using queueing theory. *Microprocessing and Microprogramming*, Volume 41:555–570, 1996.

[33] Thomas-kern load balancers. http://www.thomas-krenn.com.

[34] B. Urgaonkar, G. Pacifici, P. Shenoy, M. Spreitzer, and A. Tantawi. Analytic modeling of multitier Internet applications. *ACM Transactions on the Web*, Volume 1(2), 2007.

[35] L. Wells. Performance analysis using CPN tools. In *Proc. of the 1st Inter. Conf. on Performance*

Evaluation Methodolgies and Tools, 2006. Article No. 59.

[36] L. Wells, S. Christensen, L. Kristensen, and K. Mortensen. Simulation based performance analysis of web servers. In *Proc. of the 9th Internat. Workshop on Petri Nets and Perf. Models*, page 59. IEEE, 2001.

[37] Q. Zhang, A. Riska, W. Sun, E. Smirni, and G. Ciardo. Workload-aware load balancing for clustered web servers. *IEEE Transactions on Parallel and Distributed Systems*, Volume 16(3):219–233, March 2005.

[38] W. Zhang. Linux virtual server for scalable network services. In *Ottava Linux Symposioum. Proceedings*, 2000.

[39] Z. Zhang and W. Fan. Web server load balancing: A queueing analysis. *European Journal of Operational Research*, Volume 186(2):681–693, April 2008.