HTCPNs–Based Tool for Web–Server Clusters Development

Slawomir Samolej, Tomasz Szmuc

HTCPNs–based Tool for Web–Server Clusters Development

Slawomir Samolej* and Tomasz Szmuc**

* Department of Computer and Control Engineering, Rzeszow University of Technology, Ul. W. Pola 2, 35-959 Rzeszw, Poland,
** Institute of Automatics, AGH University of Science and Technology, Al. Mickiewicza 30, 30-059 Krakw, Poland
*ssamolej@prz.edu.pl, **tsz@agh.edu.pl

Abstract. A new software tool for web–server clusters 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–cluster 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 the guidelines for applying it in cluster–based web–server development.

Key words: Hierarchical Timed Coloured Petri Nets, Web–Server Clusters, Performance 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–a server cluster [3, 8, 13, 19]. 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.

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 [7, 11, 12]. Second, some distribution strategies of cluster nodes are investigated [3] in conjunction with searching for load balancing policies for the nodes [5, 18, 21]. In several research projects reported in [8, 17, 19] 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 [2, 8, 17–19, 21]. In [2, 18, 19, 21] Queueing Nets whereas in [17] Stochastic Petri Nets are applied for 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)[8]. 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 [9]. The research results reported in [8] 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) [6],
- 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 has an expression power comparable to QPNs. Second, the available software toolkits for HTCPNs composition and validation seem to be more popular than"SimQPN" [9]. Third, there exist a reach knowledge base of successful HTCPNs applications to modelling and validation of wide range software/hardware systems [6] including web–servers [13, 14, 20]. 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 application HTCPNs to real–time systems development [15, 16].

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. Section 3 touches the simulation based HTCPNs models validation methods. 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 [6]. All the Coloured Petri Nets in the paper have been edited and analysed using Design/CPN tool.

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/∞ [13, 14]. *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 [13, 14]). 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. Finally, top–level HTCPNs structure of an example cluster–server model will be shown.

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 fist group consist of so–called local tokens,

that"live" in individual design patters.They provide local functions and data structures for the patterns. The second group tokens represent Internet requests that are serviced in the system. They are transported throughout several cluster components. Their internal state carries the data that may be used for timing and performance evaluation of the system modelled. As the tokens representing 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 [7]. $START\_TIME$ 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 (eg. with processor sharing service).

2.2 Queueing system models

The basic components of the software tool for web–server clusters development that is being 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 make it possible to monitor the internal state of the queue during it's execution.

The example HTCPNs based queueing system model is shown in fig. 1. The model is a HTCPNs subpage that can communicate with the parent page via $INPUT\_PACKS$, $OUTPUT\_PACKS$ and QL port places. The 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 make it possible to model the duration of packet execution. When $REMOVE\_FIFO$ transition fires, 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 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 [6]). The packets are treated as serviced when they

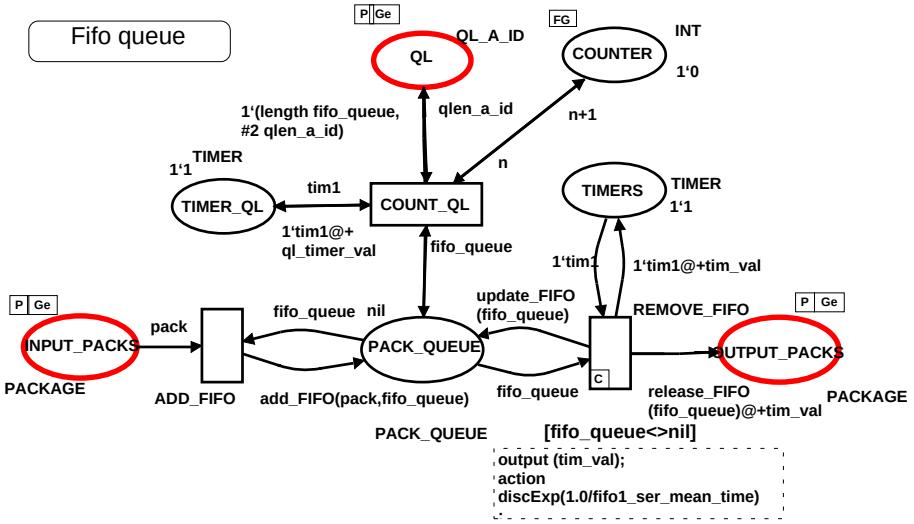


Fig. 1: HTCPNs based –/1/FIFO/∞ queueing system model.

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.

The main parameters that define the queueing system model dynamics are the queue mean service time, the service time probability distribution function and the number of servicing units. The 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 Cluster load–balancing model

Having a set of *queueing systems design patterns* some *packet distribution HTCPNs structures* may be proposed. In [13] a typical homogenous multi–tier web–server structure pattern was examined, whereas in [14] a preliminary version of server structure with feedback like admission control of Internet requests was introduced. The packet distribution pattern presented in this paper touches the load balancing in web–server cluster problem.

Fig. 2 includes an example cluster load–balancing HTCPNs model.The cluster consist of 3 computers represented as $FIFO1...FIFO3$ substitution transitions, where each transition is attached to a $FIFO$ queueing pattern. The Internet requests serviced by the cluster arrive through $PACKS2$ port place. A *load balancer* decides where the currently acquired request should be send to achieve an uniform load for all, even heterogenic nodes of the cluster.

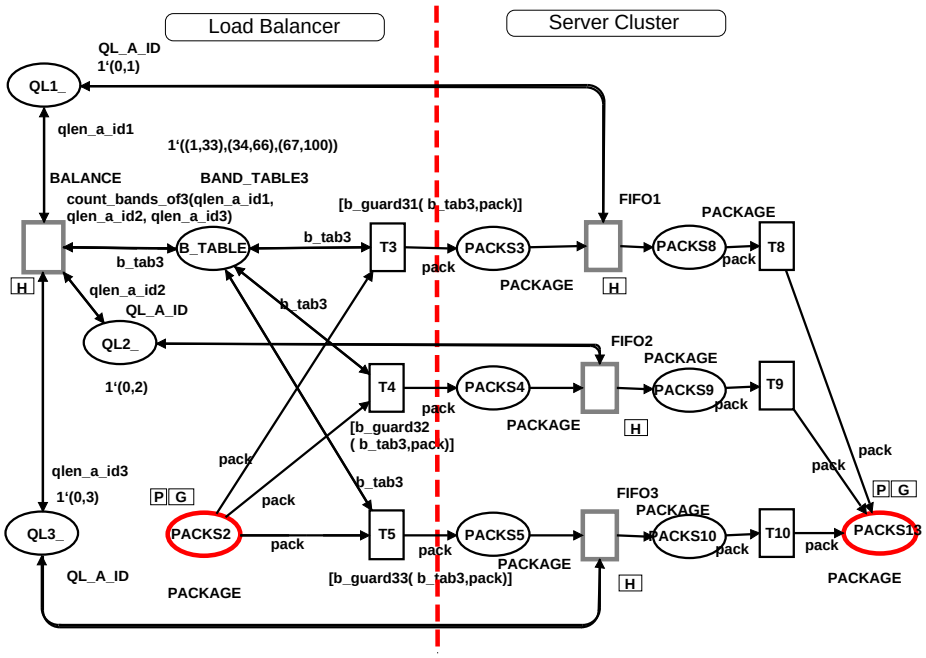


Fig. 2: Server cluster with load balancing model.

Generally, the load balancing procedure can follow the Fewest Server Processes First [17] or the Adaptive Load Sharing [5] algorithms. In both algorithms some feedback information about the state of cluster nodes under balance is needed. $QL1\_$, $QL2\_$ and $QL3\_$ places (connected to corresponding QL port places of $FIFO$ queueing system models–compare section 2.2) provide the queue's lengths of each cluster node to the load balance procedure. The less loaded server nodes have the highest probability to get a new request to serve.

The HTCPNs implementation of the algorithms involves periodical firing $BALANCE$ transition. During the firing, a set of threshold values is generated and stored in B_TABLE place. Finally, the thresholds values are mapped to guard functions associated to $T3$, $T4$ and $T5$ transitions and can be understand as some kind of bandwidths for the requests streams. At current state of the

design pattern composition, the only load balancer parameter that influences its dynamics is the frequency at witch the load of cluster nodes is measured.

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.

Fig. 3 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 timestamps for tokens stored in $TIMER0$ place. The values have the probability distribution function defined. 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 follows the distribution function mentioned. $PACKS1$ place has a port place status and thereafter tokens appearing in it can be consumed by other model components (e.g. server cluster model).

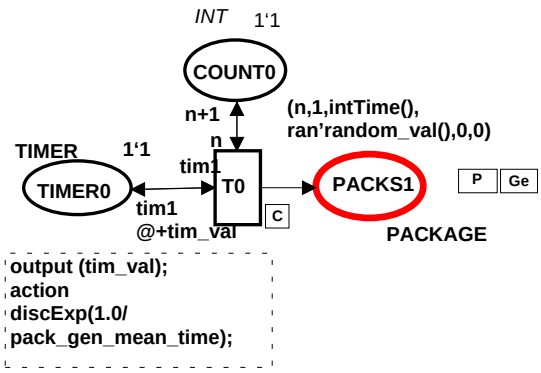


Fig. 3: Web–server arrival process model.

The Internet request frequency can have any standard probability distribution function or can be individually constructed as it was proposed in [20].

2.5 Example top–level cluster 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.

Fig. 4 includes an example top–level HTCPN model of server cluster that follows the abovementioned modelling development rules. The HTCPN in fig. 4 consist of 2 substitution transitions. $INPUT\_PROCS$ transition represents the arrival process for the server cluster whereas $SERVER\_CLUSTER$ transition represents example one–layer web–server cluster. The modelling process can be easily continued by attaching the request generator model as in section 2.4 under the $INPUT\_PROCS$ transition and by attaching the cluster model with load balancing module as in section 2.3 under $SERVER\_CLUSTER$ 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).

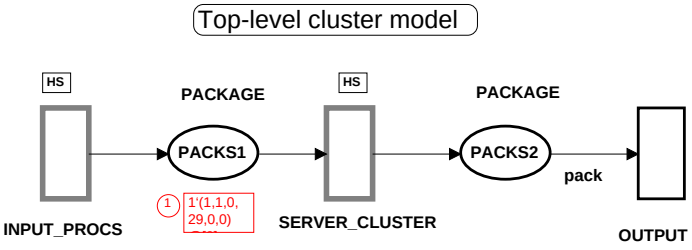


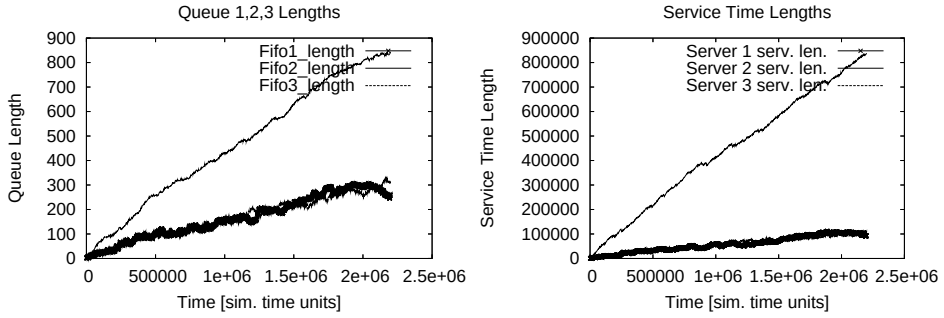
Fig. 4: Example top–level cluster server model.

3 Model validation capabilities

Typical elements of HTCPNs modelling software tools are performance evaluation routines, e.g.: [10] . The routines make it possible to capture the state of dedicated tokens or places 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 seem to be the most natural load

measure available at typical software systems. The service time lengths are measurable in the modelling method proposed because of a special kind $PACKAGE$ type tokens construction (compare section 2.1). The tokens "remember" the simulation time at witch their appear at the cluster and thereafter the time at each state of their service may be captured. In real systems the service time is one of predominant quality of service parameters for performance evaluation.

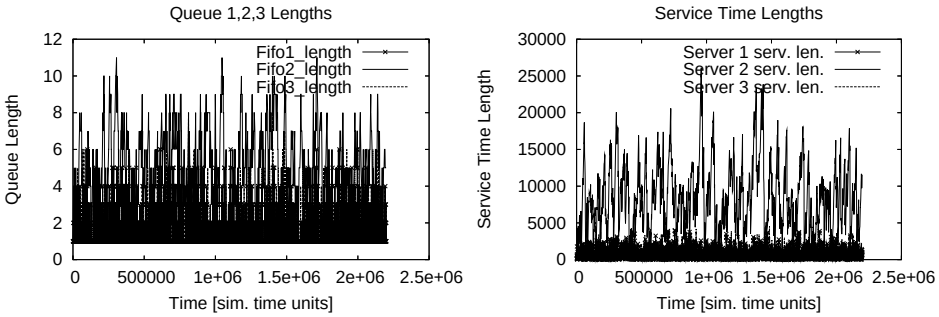


(a) (b)

Fig. 5: Queue lengths (a) and service times (b) under overload condition.

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 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 because there are not limitations for queue lengths in the modelling method proposed. Fig. 5 shows the queue lengths (fig. 5a) and service time lengths (fig. 5b) when the example web server cluster model presented in the paper experiences the permanent overload.

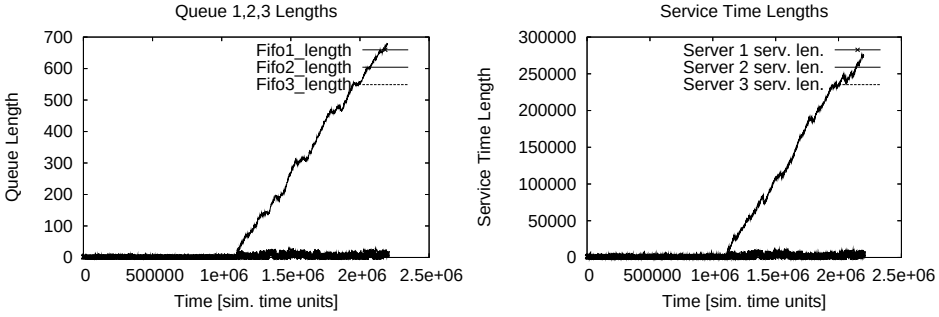
Second, the average values of queueing system systems parameters such as average queue lengths and average servicing time for the balanced model can be estimated. Provided that the arrival process model and the server nodes models parameters are acquired from the real devices as in [11, 17, 19, 20], the software model can be used for derivation the system properties under different load conditions. In the fig. 6 queue lengths (fig. 6a) and service times (fig. 6b) under stable system execution are shown. The cluster had a heterogenic structure, where server 2 (fifo 2 model) had 4 times lower performance. The load balance procedure was trying to reduce amount of Internet requests for the second server executing the Adaptive Load Sharing [5] algorithm. 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



(a) (b)

Fig. 6: Queue lengths (a) and service times (b) under stable system execution.

was 7471 time units. Third, some individual properties of cluster node structures or load balancing strategies may be observed. For example, in some load balancing strategies mentioned in [5], the load of cluster node is estimated without any feedback information from the node. Such load balancing strategy may easily fail when node performance becomes reduced due to some external reasons (e.g. hardware fail or some extra node load). Fig. 7 shows a possible web–server cluster reaction to the not reported performance reduction of one server node. It can be easily noticed that under some disadvantageous conditions the loss or unavailability of feedback information about the current state of cluster server can lead to its unstable behavior. The Internet requests scheduled to server 2 node after the node performaance reduction (at approximately 1000000 time units) may not be serviced due to the server overload.



(a) (b)

Fig. 7: Example queue lengths (a) and service times (b) of unbalanced web–server cluster.

4 Conclusions and future research

The paper introduces the HTCPNs–based software tool making it possible to construct and validate some 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 validaton. Queueing systems templates was checked whether they meet theoretically derived performance functions.

The analysis of HTCPNs simulation reports makes it possible 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.

Currently, the software tool announced 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. Bause, F.: Queueing Petri Nets – a formalism for the combined qualititative and quantitative analysis of systems. In: PNPM'93, pp. 14–23, IEEE Press, (1993).
2. Cao, J., Andersson, M., Nyberg, C., Khil, M.: Web Server Performance Modeling Using an M/G/1/K*PS Queue. In: ICT 2003, 10th International Conference on Telecommunications, pp. 1501–1506, vol. 2, (2003)
3. Cardellini, V., Casalicchio, E., Colajanni M.: The State of the Art in Locally Distributed Web-Server Systems. ACM Computing Surveys, Vol. 34, No. 2,263–311 (2002)
4. Filipowicz, B.: Stochastic models in operations research, analysis and synthesis of service systems and queueing networks. (in Polish), WNT, Warszawa (1997)
5. Guo, J., Bhuyan, L.N.: Load Balancing in a Cluster-Based Web Server for Multimedia Applications. IEEE Transactions on Parallel and Distributed Systems, Vol. 17, No. 11, 1321–1334, (2006)
6. Jensen, K.: Coloured Petri Nets, Basic Concepts, Analysis Methods and Practical Use. Springer (1996)
7. Kim, D., Lee, S., Han, S, Abraham, A.: Improving Web Services Performance Using Priority Allocation Method. In: Proc. Of International Conference on Next Generation Web Services Practices, pp. 201–206, IEEE, (2005)

8. Konunev, S.: Performance Modelling and Evaluation of Distributed Component–Based Systems Using Queuing Petri Nets. IEEE Transactions on Software Engineering, Vol 32. No. 7, 486–502, (2006)
9. Kounev, S., Buchmann, A.: SimQPN–A tool and methodology for analyzing queueing Petri net models by means of simulation. Performance Evaluation, Vol. 63, Issues 4–5, 364–394, (2006)
10. Linstrom B., Wells L.: Design/CPN Perf. Tool Manual. CPN Group, Univ. of Aarhus, Denmark, (1999)
11. Liu, X., Sha, L., Diao, Y., Froehlich, S., Hellerstein, J. L., Parekh, S.: Online Response Time Optimization of Apache Web Server, In: IWQoS 2003: 11th International Workshop, LNCS Vol. 2707/2003, pp. 461–478, Springer, (2003)
12. Liu, X., Zheng, R., Heo, J., Wang, Q, Sha, L.: 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, p.75, IEEE, (2005)
13. Samolej, S., Rak, T.: Timing Properties of Internet Systems Modelling Using Coloured Petri Nets. (in Polish), Systemy czasu rzeczywistego–Kierunki badan i rozwoju, Wydawnictwa Komunikacji i Lacznosci, 91–100, (2005)
14. Samolej, S., Szmuc, T.: Dedicated Internet Systems Design Using Timed Coloured Petri Nets. (in Polish), Systemy czasu rzeczywistego–Metody i zastosowania, Wydawnictwa Komunikacji i Lacznosci, 87–96, (2007)
15. Samolej, S., Szmuc, T.: TCPN–Based Tool for Timing Constraints Modelling and Validation. Software Engineering: Evolution and Emerging Technologies, Volume 130 Frontiers in Artificial Intelligence and Applications, pp. 194–205, IOS Press, (2005)
16. Samolej, S., Szmuc, T.: Time Constraints Modeling And Verification Using Timed Colored Petri Nets. Real–Time Programming 2004, pp. 127–132, Elsevier, (2005)
17. Shan, Z., Lin, C., Marinecu, D.C., Yang, Y.: Modelling and performance analysis of QoS–aware load balancing of Web–server clusters. Computer Networks, Vol. 40, 235–256, (2002)
18. Spies, F.: Modeling of Optimal load balancing strategy using queueing theory. Microprocessing and Microprogramming, Vol. 41, Elsevier, 555–570, (1996)
19. Urgaonkar, B., et all: An Analytical Model for Multi–tier Internet Services and Its Applications. ACM SIGMETRICS Performance Evaluation Review, Vol. 33, 291–302, (2005)
20. Wells L., et all: Simulation Based Performance Analysis of Web Servers. In: Proc. of the 9th Internat. Workshop on Petri Nets and Perf. Models, p. 59, IEEE, (2001)
21. Zhang, Z., Fan, W.: Web server load balancing: A queuing analysis. European Journal of Operation Research, Vo. 186, 681–693, (2008)