HTCPNs–Based Modelling and Evaluation of Dynamic Computer Cluster Reconfiguration

Sławomir Samolej[1] and Tomasz Szmuc[2]

[1] Rzeszow University of Technology, Department of Computer and Control Engineering,
Ul. W. Pola 2, 35-959 Rzeszów, Poland
ssamolej@prz.edu.pl
[2] AGH University of Science and Technology,
Department of Automatics,
Al. Mickiewicza 30, 30-059 Kraków, Poland
tsz@agh.edu.pl

Abstract. An extension of HTCPNs-based software tool for Internet systems modelling and evaluation has been proposed in the paper. After the extension the tool provides new possibilities for modelling and analysis of dynamic cluster reconfiguration algorithm. Some implementation details of the algorithm has been presented. The algorithm has been preliminary assessed by simulation.

Keywords: Hierarchical Timed Coloured Petri Nets, Web-Server Systems, Performance Evaluation, Dynamic Reconfiguration.

1 Introduction

Current Internet (or web) servers usually have distributed multi-tier structure [1,2,3,4,5,6,7,8]. Each tier consist of one or more computers and constitutes a hardware platform for *layers* of a distributed software system running on it. The tier including a set of computers is often called a *computer cluster*. Typically, a user of the software system connects to its *presentation layer* seen as a web site and forces actions to acquire some data. The user request is then send to system's *application layer* that may offer the requested data or may pass the request to *database layer*. The above-mentioned software system's layers may be attached to separate hardware layers or may be only partly distributed. Figure 1 shows a hardware structure of an exemplary Internet system. *Presentation* and *application* software layers are deployed in "WWW cluster" and cooperate with "Database cluster" that stores a distributed data base. This distributed multi-tier development approach enables natural scalability and redundancy of both hardware and software modules of the system.

Current Internet servers are developed to efficiently serve an average frequency of requests stream, and to "survive" transient overloads [1,2,3,4,8]. The typical development procedure covers: estimation of hardware/software performance parameters of system components and performance analysis of prototype or model of the system under different load. The systematic performance analysis makes it possible to detect bottlenecks in the system and to predict overall system efficiency. As the system is of distributed structure, it is possible to iteratively analyse it, and then reconfigure it in order to remove defects.

T. Szmuc, M. Szpyrka, and J. Zendulka (Eds.): CEE-SET 2009, LNCS 7054, pp. 97–108, 2012.

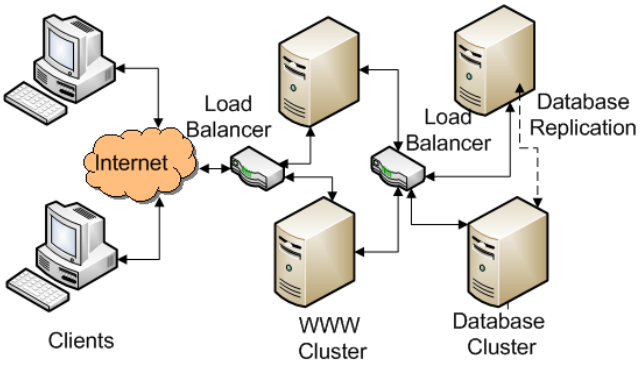


Fig. 1. Hardware configuration of an example multi-tier Internet system

It seems that the main thread of research concerning development of future Internet servers concentrate on the following subjects. Firstly, load balancing procedures for clusters are systematically proposed and evaluated [9,10,11,12,13]. Secondly, the software of individual Internet server nodes is modified to satisfy average response time for dedicated classes of consumers [14,15,16]. Thirdly, a possibility of dynamic reconfiguration of the clusters to reduce power consumption [17,18,19,20] or to compensate failure of some nodes [21,22] are investigated. In this paper the third of the above-mentioned research paths will be developed in detail.

The research reported in this paper constitute a part of long-term programme that concentrates on applications of Hierarchical Timed Coloured Petri Nets (HTCPNs) [23,24] for distributed Internet systems modelling, development, and performance evaluation [3,4,25,5,6,7]. The main aim of the programme is to acquire the systematic HTCPNs-based method for Internet systems development. It is planned to develop a set of HTCPNs models for current and future hardware/software subsystems of distributed Internet systems. The models will constitute a set of *design patterns* or *building blocks*. The blocks or patterns might be used to compose a virtual prototype of a distributed Internet system. It seems that a main advantage of using HTCPNs as a modelling language is a possibility to acquire both structural and executable model of system. Consequently future developer would have the possibility to model, simulate and evaluate developed system using one formalism. So far preliminary HTCPNs based Internet systems modelling and analysis was introduced [4,25], and also more mature methodology for systematic modeling and analysis was proposed [3,5,6]. The modelling framework was used for current Internet systems modelling [3,4], analyses of admission control [25] and scheduling [7] of Internet requests algorithms, and selected cluster load balancing policies modelling and evaluation [5,6]. In this paper the method will be extended by a dynamic cluster reconfiguration algorithm model and its performance analysis.

The next subsections of the paper are organised as follows. Firstly, a method of applying the TCPNs for distributed Internet system modelling will be briefly reminded. Secondly, the set of HTCPNs-based patterns will be extended by a new cluster dynamic reconfiguration algorithm. Some algorithm parameters setting rules will also be

proposed. Then, simulation results of the algorithm application for an example cluster will be presented and discussed. The final part of the paper will include related works discussion, conclusions and future research plans.

The HTCPNs-based Internet system modelling and performance analysis method applied in this paper combines features from the two formalisms: Queueing Networks [26,27] and Hierarchical Timed Coloured Petri Nets [23,24] . The models presented in the paper were created using DesignCPN software toolkit [28] and analysed by its performance evaluation subsystem [29]. The equivalent results may be easily obtained with CPN Tools toolkit [30,31].

2 Web-Server Modelling Methodology Overview

Internet systems development is one of the rapidly extended branch of computer engineering. The predominant amount of research concerning this area concentrates on developing or applying new practical algorithms or procedures that improve selected features of the system. In most solutions at hoc experimental systems are constructed and analysed to prove the benefits of the improvement proposed [14,15,16,10,19,20,11,12]. Research reported in this paper and our previous works [3,4,25,5,6,7] aim at systematisation of modern Internet system development techniques into one consistent methodology. Inspired by research described in [32,2,33] we decided to propose both methodology and software tool based on application of Hierarchical Timed Coloured Petri nets and Queueing Systems, and providing systematic modelling and performance analysis of Internet systems.

The main features of the methodology at its current state of development are:

- Hierarchical Timed Coloured Petri Nets (HTCPNs) [23,24] are used as modelling language,
- A set of the so called HTCPNs design patterns (predefined net structures) have been prepared and validated to model typical Internet or web systems components;
- The basic patterns are executable models of ”classic” and ”modified” queueing systems;
- Internet or web systems are modelled using the three levels: *queueing systems level*, *packet distribution level*, and *top system model level*;
- A set of design rules of composing system models using the patterns has been formulated;
- A way of applying the performance analysis subsystem [29] of DesignCPN software toolkit [28] for the analysis of the models have been proposed.

The main concept of the methodology lies in a definition of reusable timed coloured Petri nets structures (patterns) giving possibilities for composition of web-server models in a systematic way. The patterns have been divided into three levels described below.

The *queueing systems level* includes the basic set of the patterns that are TCPNs implementations of queueing systems [26,27]. So far some ”classic” queueing system models, such as –/M/PS/∞ , –/M/FIFO/∞ were defined [3,5]. Additionally, some extended queueing system models were proposed, i.e. the queueing system with admission control of serviced items [25], and the queueing systems with real-time scheduling policies for serviced items [7].

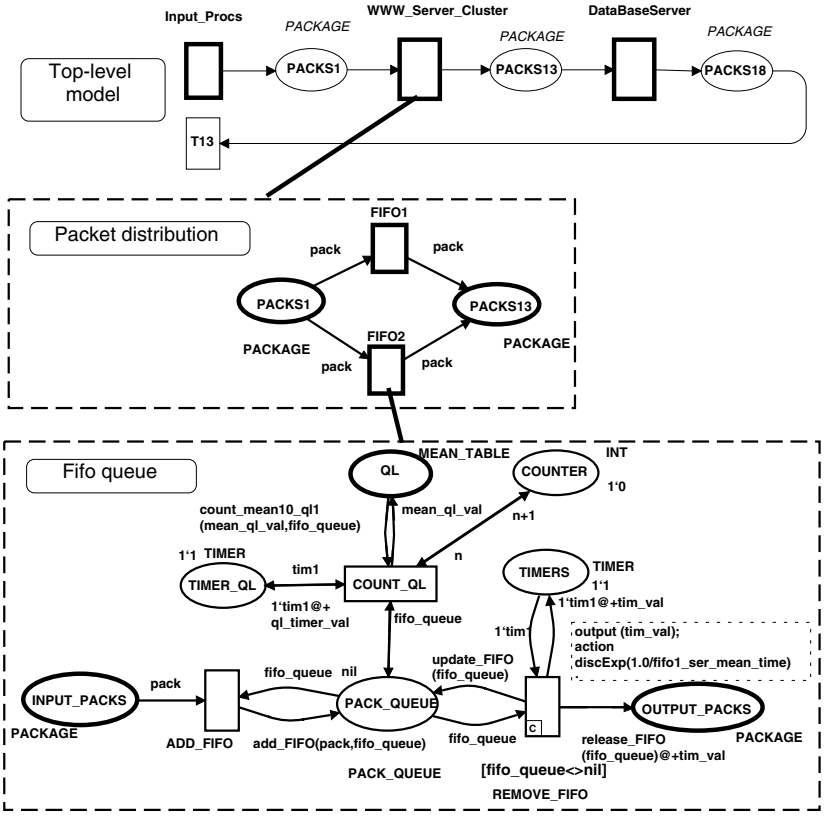


Fig. 2. An overview of HTCPNs-based Internet systems modelling methodology

A role of the *packet distribution level* is to provide some predefined web-server cluster substructures composed from the queueing systems. At this stage of system modelling a queueing system is represented as a separate *substitution transition* [23,24]. So far some models representing typical current Internet systems structures [3,4] and load balancing Internet systems modules [5,6] were proposed.

At the *top system model level* of system description the main components or clusters are highlighted. Each component is usually a substitution transition "hiding" a computer cluster structure.

For the performance evaluation purposes it is possible to define the external input process that generates Internet requests according to an adequate distribution function or according to some historical data. As the complete system model may be treated as an open queueing network, the requests are generated, then sent to the computer system model, and afterward served and removed.

Each Internet request is a HTCPNs token defined as 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 assign unique number to any request. *Request priority* is an integer value that may be used when requests are scheduled according priority driven strategy [14]. $START\_TIME$ 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).

Figure 2 illustrates the system modelling methodology above explained. It includes selected elements of the multi-tier web-system outlined in Fig. 1. The *top system model level* consists of three HTCPNs substitution transitions representing "input process", "presentation and application layer" and "database layer". The *presentation and application layer* is executed in the cluster that consist of two computers depicted as FIFO1 and FIFO2 substitution transitions respectively. The cluster structure belongs to *packet distribution level*. Each of computers in the cluster are modelled in detail as a –/M/FIFO/∞ quequeing system that belongs to the *queueing systems level*. 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 is used for modelling 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 time stamps generated according to the assumed service time calculated using 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 [23,24]). 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.

The complete system model can be executed and evaluated. It has been assumed that performance analysis will be the main way of the model evaluation. The corresponding DesignCPN toolkit subroutines [29] are responsible for capturing 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 off-line. At the currently reported version of the software tool for web-server systems modelling and analysis, queue lengths and service time lengths can be stored during the model execution. The performance analysis of models can be applied in the following domains. Firstly, the system instability and bottlenecks may be easily detected. Secondly, the average values of queueing system parameters such as average queue lengths, and average servicing time for the balanced model can be estimated. Thirdly, some individual properties of cluster node structures or load balancing strategies may be observed.

The modeling and analysis method discussed in the paper has been tested so far with good results on small and medium system models (e.g. 2 tiers and up to 15 quqeueing systems). To us, much more complex system models may be naturally constructed and

examined as DesignCPN [28,29] and CPN Tools [30,31] software toolkits have been already successfully used for detailed modeling and tracking of big hardware/software systems [23,24,34]. The analysis method, based on performance reports collected during simulation, do not suffer from state explosion problem, so it also should give stable results in complex model system assessing. Moreover, the detailed evaluation of the method and the software toolkits capabilities for complex system modeling and analysis will be subject of one of our research paths.

3 Example Dynamic Cluster Reconfiguration Algorithm Evaluation

The distributed structure of current Internet systems entails natural possibility of their reconfiguration. If the system is not efficient enough, some new resources can be added to improve its performance. On the contrary, it is quite natural to switch off some of the nodes when the cluster is evidently underloaded. The states leading to reconfigurations have usually two reasons: an attempt of power consumption reduction, or failures of some system nodes. Currently, the system reconfiguration is made by the administrator, however recent research have shown that it is possible to automate of this procedure [17,18,19,20].

Figure 3 includes a HTCPNs-based model of computer cluster where an example power-aware dynamic cluster reconfiguration algorithm was implemented. The model

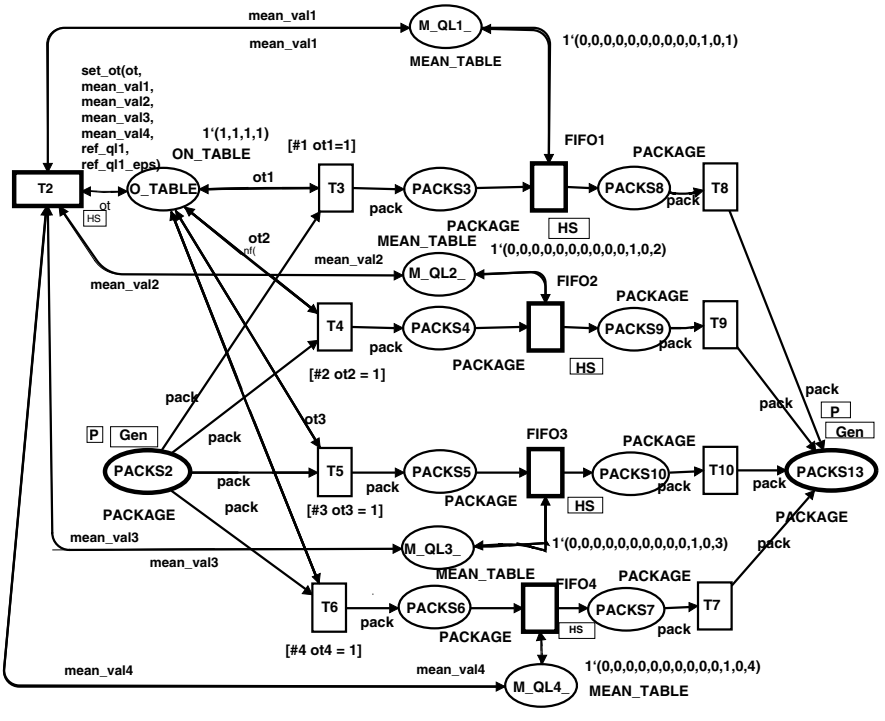


Fig. 3. HTCPNs model of example dynamic cluster reconfiguration algorithm

belongs to set of patterns assigned to *packet distribution level* (comp. sect. 2). The cluster consist of 4 computers represented as $FIFO1..FIFO4$ substitution transitions, where each transition is attached to a $FIFO$ queueing pattern. Internet requests serviced by the cluster arrive through $PACKS2$ port place and lives the cluster through the $PACKS13$ one.

The cluster reconfiguration algorithm works as follows. $M\_QL1\_$, $M\_QL2\_$, $M\_QL3\_$ and $M\_QL4\_$ places provide the average queue's lengths of each cluster node. Periodically fired $T2$ transition collects the average lengths of the queues and adjust the number of "running" cluster nodes to fulfill the following conditions:

- the average queue length must be kept between assumed extreme values;
- the power consumption (the number of running cluster nodes) must be minimised.

The O_TABLE place includes the tuple that decides which cluster node is enabled to run. The corresponding tuple values are used as parameters of guard functions attached to $T3$, $T4$, $T5$, and $T6$ transitions. Only the transitions which guard functions acquire positive value let the requests to go through. The set of "open" transitions define the set of "executing" nodes of the cluster. For example, if the tuple has of (1,1,1,0) value, three cluster nodes are turned on, whereas one (the last) does not work.

The "control" procedure setting the number of the nodes allowed to execution is specified using the following CPN ML code:

```
(*input:  table of running nodes, queue lengths, *)
(*   reference queue length, *)
(*   queue length insensitivity zone;*)
(*output: new table of running nodes; *)
fun set_ot(ot_: ON_TABLE,
   mql1_: MEAN_TABLE, mql2_: MEAN_TABLE,
            mql3_: MEAN_TABLE, mql4_: MEAN_TABLE,
            ref_ql: INT, ref_ql_eps: INT)=
let

(*calculate average queue length: *)
val av_ql=count_av_ql4(ot_,mql1_,mql2_,mql3_,mql4_)
in

(*if average queue length is bigger then or equal *)
(*to the assumed value *)
(*then turn on the next cluster node: *)
if (av_ql >= (ref_ql + ref_ql_eps))
then
case ot_ of
 (1,0,0,0) => (1,1,0,0)
|(1,1,0,0) => (1,1,1,0)
|(1,1,1,0) => (1,1,1,1)
|(1,1,1,1) => (1,1,1,1)
else
```

```
(*if average queue length is smaller than or equal *)
(*to the assumed value *)
(*then turn of the next cluster node: *)
 if(av_ql <= (ref_ql - ref_ql_eps))
 then
 case ot_ of
  (1,1,1,1) => (1,1,1,0)
 |(1,1,1,0) => (1,1,0,0)
 |(1,1,0,0) => (1,0,0,0)
 |(1,0,0,0) => (1,0,0,0)

 (*else do not reconfigure the cluster: *)
 else ot_
end;
```

The main parameters of the procedure are:

- average queue lengths ($mql1\_$, $mql2\_$, $mql3\_$ $mql4\_$);
- assumed average queue length (ref_ql);
- assumed extreme queue length fluctuation value (ref_ql_eps) that enforces the reconfiguration;
- frequency of the transition firing.

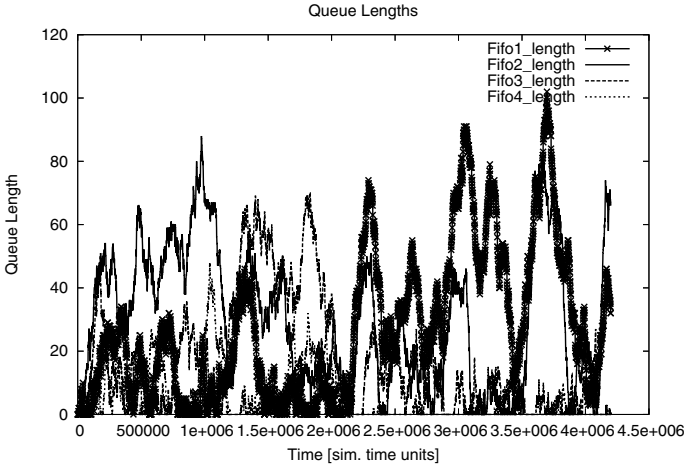
The procedure has been constructed under assumption that all of the nodes are homogeneous. In the case when it is possible, the procedure tries to reduce number of running nodes, under condition of preservation of average queues lengths for cluster nodes. The condition can be naturally converted into cluster's average response time.

Figure 4 shows queue lengths (Fig. 4a) and number of executing cluster nodes (Fig. 4b) in the cluster controlled by the example dynamic reconfiguration procedure during the sample simulation. After 2000000 time units the external system load was reduced. That caused activation of the reconfiguration procedure. The procedure has kept the average queues lengths at the same level, and has modified number of executing cluster nodes. In a typical Internet system all the cluster nodes would have been servicing the requests. In the modelled system some servers have been turned off for some amount of time. The number of executing cluster nodes can be directly interpreted as an amount of power needed for efficient service of the acquired requests stream [19]. It can be easily calculated that the reconfiguration procedure saved approximately 20% of power in the case modelled by this experiment.

4 Related Work

The need of dynamic cluster reconfiguration in future Internet systems was noticed by several authors making independent research.

a)



b)

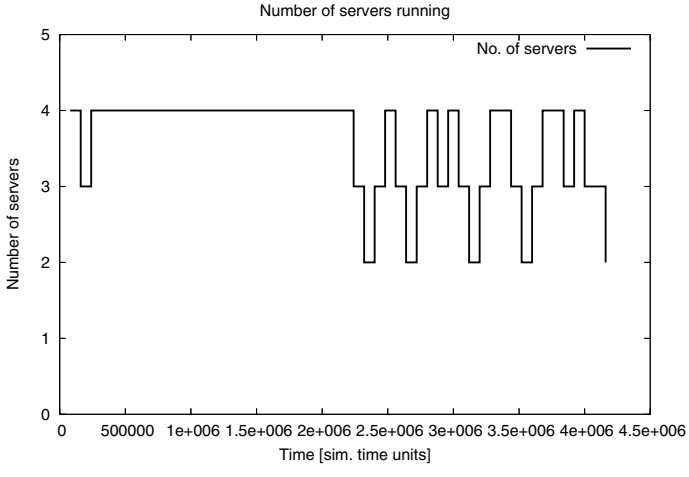


Fig. 4. Queue lengths (a) and number of executing servers (b) – under different load conditions

One of the earliest research results were these published in [19]. The authors proposed an algorithm of power-aware dynamic homogeneous cluster reconfiguration. The input for the algorithm are CPU, disk and network interface utilisations and their dynamics. The dynamics of the input is interpreted by PID control algorithm.

In [20] and [18] the authors independently propose a set of solutions dedicated for heterogeneous computer clusters treated as real-time systems. Firstly, a new on/of policy has been proposed that causes turn on/off machines according to their power efficiency. Secondly, a possibility of Dynamic Voltage Scaling (DVS) application in cluster power management has been analysed.

A set of interesting dynamic reconfiguration policies dedicated to Internet services keeping long-lived connections has been proposed in [17]. The proposed solutions use forecasting techniques, load balancing, and load skewing for reduction of the cluster power consumption.

In [35] the problem of heterogeneous computer cluster power consumption reduction is tackled by attempt to find the best Internet requests distribution among the cluster nodes. An iterative optimisation algorithm that evaluates off-line the space of Internet requests distributions and cluster configurations has been applied to solve the problem.

In [22] and [21] dynamic reconfiguration policies have been applied for the modification of cluster structure, where some nodes were failed.

The dynamic cluster reconfiguration algorithm shown in this paper can be treated as a simplified version of policies proposed in [20,18]. The main algorithm input data (queue lengths) come from typical system performance values. The cluster model is constructed on the basis of our previous research [3,4,25,5,6,7], and can be easily related to the one proposed in [18].

5 Conclusions and Future Research

An extension of HTCPNs-based software tool for Internet systems modelling and evaluation has been proposed in the paper. After the extension the tool provides new possibilities for modeling and analysis of dynamic cluster reconfiguration algorithm using executable model. Some implementation details of the algorithm has been presented and discussed. Advantages of the algorithm has been preliminary assessed by simulation.

The paper may also be treated as an attempt to systematisation and generalisation of current research concerning efficiency of current and future Internet systems. It seems that the results of the research discussed in this paper and the previous ones have proved that it is possible to construct a HTCPNs-based methodology and related software tool for distributed Internet systems modelling, development, and performance evaluation.

Still, the software tool extended in the paper can be applied for modelling and validation of limited structures of Internet systems. Thereafter the main stream of authors' future research will concentrate on developing models for the next structures. In the nearest future, dynamic load-balancing and dynamic cluster reconfiguration policies will be systematically modelled and assessed by the correspondingly extended software tool. Some improvements of the previously proposed policies are also expected.

So far simulation and performance evaluation were the main ways of the modelled system analysis in the proposed method. However, recent research results announced in [24,36] seem to provide the possibility to effective analyse Timed Coloured Petri Nets using the modified reachability graphs. The attempt of application of the mentioned analysys methods to system models presented in this and the previous papers will be another subject of the authors' research.

References

1. Cardellini, V., Casalicchio, E., Colajanni, M.: The State of the Art in Locally Distributed Web-Server Systems. ACM Computing Surveys 34(2), 263–311 (2002)
2. Kounev, S.: Performance Modelling and Evaluation of Distributed Component–Based Systems Using Queuing Petri Nets. IEEE Transactions on Software Engineering 32(7), 486–502 (2006)

3. Rak, T., Samolej, S.: Distributed Internet Systems Modeling Using TCPNs. In: Proc. of International Multiconference on Computer Science and Information Technology, pp. 559–566. IEEE (2008)
4. Samolej, S., Rak, T.: Timing Properties of Internet Systems Modelling Using Coloured Petri Nets (in Polish). In: Proc. of the 12th Real-time Systems Conference, WKŁ, Warsaw, pp. 91–100 (2005)
5. Samolej, S., Szmuc, T.: HTCPNs–Based Tool for Web–Server Clusters Development. In: Huzar, Z., Koci, R., Meyer, B., Walter, B., Zendulka, J. (eds.) CEE-SET 2008. LNCS, vol. 4980, pp. 131–142. Springer, Heidelberg (2011)
6. Samolej, S., Szmuc, T.: Coloured Petri Nets Application in a WWW Clusters Modelling and Development Method (in Polish). In: Proc. of the 10th Software Engineering Conference KKIO 2008, WKŁ, Warsaw, pp. 49–59 (2008)
7. Samolej, S., Szmuc, T.: HTCPNs Application for Selected Internet Requests Scheduling Algorithms Analysis (in Polish). In: Proc. of the 16th Real-time Systems Conference, WKŁ, Warsaw (2009)
8. Urgaonkar, B., Pacifici, G., Shenoy, P., Spreitzer, M., Tantawi, A.: Analytic Modeling of Multitier Internet Applications. ACM Transactions on the Web 1(2) (2007)
9. Cardellini, V., Colajanni, M., Yu, P.S.: Dynamic Load Balancing on Web-Server Systems. IEEE Internet Computing 3, 28–39 (1999)
10. Park, G., Gu, B., Heo, J., Yi, S., Han, J., Park, J., Min, H., Piao, X., Cho, Y., Park, C.W., Chung, H.J., Lee, B., Lee, S.: Adaptive Load Balancing Mechanism for Server Cluster. In: Gavrilova, M.L., Gervasi, O., Kumar, V., Tan, C.J.K., Taniar, D., Laganá, A., Mun, Y., Choo, H. (eds.) ICCSA 2006. LNCS, vol. 3983, pp. 549–557. Springer, Heidelberg (2006)
11. Shan, Z., Lin, C., Marinecu, D., Yang, Y.: Modelling and Performance Analysis of QoS–aware Load Balancing of Web–Server Clusters. Computer Networks 40, 235–256 (2002)
12. Zhang, Q., Riska, A., Sun, W., Smirni, E., Ciardo, G.: Workload-aware Load Balancing for Clustered Web Servers. IEEE Transactions on Parallel and Distributed Systems 16, 219–233 (2005)
13. Zhang, Z.: Web Server Load Balancing: A Queuing Analysis. European Journal of Operation Research 186, 681–693 (2008)
14. 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)
15. Liu, X., Sha, L., Diao, Y., Froehlich, S., Hellerstein, J.L., Parekh, S.: Online Response Time Optimization of Apache Web Server. In: Jeffay, K., Stoica, I., Wehrle, K. (eds.) IWQoS 2003. LNCS, vol. 2707, pp. 461–478. Springer, Heidelberg (2003)
16. 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 International Conference on Autonomic and Autonomous Systems and International Conference on Networking and Services, p. 75. IEEE (2005)
17. Chen, G., He, W., Liu, J., Nath, S., Rigas, L., Xiao, L., Zhao, F.: Energy-aware Server Provisioning and Load Dispatching for Connection-Intensive Internet Services. In: Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation, pp. 337–350 (2008)
18. Guerra, R., Bertini, L., Leite, J.C.B.: Improving Response Time and Energy Efficiency in Server Clusters. In: VIII Workshop Brasileiro de Tempo Real - 2006 - Curitiba (2006)
19. Pinheiro, E., Bianchini, R., Carrera, E.V., Heath, T.: Dynamic Cluster Reconfiguration for Power and Performance. In: Compilers and Operating Systems for Low Power Book Contents, pp. 75–93. Kluwer Academic Publishers (2003)

20. Rusu, C., Ferreira, A., Scordino, C., Watson, A.: Energy-efficient Real-time Heterogeneous Server Clusters. In: Proceedings of the 12th IEEE Real-Time and Embedded Technology and Applications Symposium, pp. 418–428. IEEE (2006)
21. Christodoulopoulou, R., Manassiev, K., Bilas, A., Amza, C.: Fast and Transparent Recovery for Continuous Availability of Cluster-based Servers. In: Proceedings of the 11th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, pp. 221–229 (2006)
22. Kim, M.S., Choi, M.J., Hong, J.W.: A Load Cluster Management System Using SNMP and Web. International Journal of Network Management 12(6), 367–378 (2002)
23. Jensen, K.: Coloured Petri Nets, Basic Concepts, Analysis Methods and Practical Use. Springer, Heidelberg (1996)
24. Jensen, K., Kristensen, L.M.: Coloured Petri Nets Modelling and Validation of Concurrent Systems. Springer, Heidelberg (2009)
25. Samolej, S., Szmuc, T.: Dedicated Internet Systems Design Using Timed Coloured Petri Nets (in Polish). In: Proc. of the 14th Real-time Systems Conference, WKŁ, Warsaw, pp. 87–96 (2007)
26. Bolch, G., Greiner, S., de Meer, H., Trivedi, K.S.: Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications, 2nd edn. Wiley (2006)
27. Dattatreya, G.R., Sahni, S.: Performance Analysis of Queuing and Computer Networks. Chapman and Hall (2008)
28. Meta Software Corporation: Design/CPN Reference Manual for X-Windows (1993)
29. Linstrom, B., Wells, L.: Design/CPN Perf. Tool Manual. CPN Group, Univ. of Aarhus, Denmark (1999)
30. Jensen, K., Kristensen, L.M., Wells, L.: Coloured Petri Nets and CPN Tools for Modelling and Validation of Concurrent Systems. International Journal on Software Tools for Technology Transfer (STTT) 9, 213–254 (2007)
31. Wells, L.: Performance Analysis Using CPN Tools. In: Proc. of the 1st International Conference on Performance Evaluation Methodolgies and Tools, p. 59 (2006)
32. Bause, F.: Queueing Petri Nets – a Formalism for the Combined Qualititative and Quantitative Analysis of Systems. In: PNPM 1993, pp. 14–23. IEEE (1993)
33. Kounev, S., Buchmann, A.: SimQPN–A Tool and Methodology for Analyzing Queueing Petri Net Models by Means of Simulation. Performance Evaluation 63(4–5), 364–394 (2006)
34. Wells, L., Christensen, S., Kristensen, L.M., Mortensen, K.H.: Simulation Based Performance Analysis of Web Servers. In: Proc. of the 9th International Workshop on Petri Nets and Performance Models, pp. 59–68. IEEE (2001)
35. Heath, T., Diniz, B., Carrera, E.V., Meira, W., Bianchini, R.: Energy Conservation in Heterogeneous Server Clusters. In: Proceedings of the 10th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, pp. 186–195 (2005)
36. Szpyrka, M.: Analysis of RTCP-nets with Reachability Graphs. Fundamenta Informaticae 74(2–3), 375–390 (2006)