Performance Analysis of Interactive Internet Systems for a Class of Systems with Dynamically Changing Offers

Tomasz Rak[1] and Jan Werewka[2]

[1] Rzeszow University of Technology,
Department of Computer and Control Engineering,
Rzeszow, Poland
[2] AGH University of Science and Technology,
Department of Automatics, Computer Science Laboratory,
Krakow, Poland

Abstract. This paper presents the performance analysis of interactive Internet systems, for which the time rate of system offer change is comparable to the users interaction time with the system. In this paper such systems class is called Interactive Internet Systems with Dynamically Changing Offers (IISDCO). In IISDCO systems an offer change may cause cancellation of the started but incomplete transactions. The necessity of cancellation can result from the sale of a set of resources (goods sold) which is getting exhausted or whose price has changed.
This paper includes a performance analysis of an on-line stock exchange system (sample of IISDCO). For the analysis we used models in the form of Time Coloured Petri Nets (TCPNs). We also reviewed the proposed models on the example of the Internet on-line stock exchange system.

1 Introduction

In this paper we consider a certain class of interactive Internet systems. It is assumed that these systems perform maintenance bids quickly. Execution of transactions in these systems may be cancelled due to offer expiration. We will consider only the cases in which the number of transactions started within one second amounts to hundreds or thousands. We assume further, that a large part of the transaction will involve the same offer or the same range of products. In addition, the transaction process must take into account the results of a previous transaction connected with the same offer.

The presented system class is interesting from the practical point of view. A stock exchange system, with transactions carried out on-line, could be its representative. These transactions are performed in real time. Within each second, all over the world there are generated thousands of events related to the stock exchange. They must be sent in real time to participants, who can be located in various sites.

Today, many of the e-commerce systems are based on multi-layered and distributed structures [2]. The complexity of this architecture is difficult for the systems to provide the desired level of service (Quality of Service) [6]. The major problems are as follows:

1) temporary heavy load, a large number of requests supported per second,
2) average response time increase,
3) resources overloading (system performance decrease).

On the other hand, developers are often faced with questions such as the following [3]:

1. How could performance change if load is increased?
2. What would the response time be?
3. Which components hale the largest effect on the overall system performance?

The remaining work is organized as follows. In Section 2, we introduce two-layered Internet distributed system. Section 3 presents performance analysis of TCPNs models. The last section sums up the paper and includes our future research plans.

2 Two-Layered Internet Distributed System

A typical architecture of the Internet is usually made up of several layers. The presented architecture (Fig. 1) is very popular on the Internet and is based on two layers: the front-end and back-end layer [5]. The double-layer architecture was chosen because of the possibility of a full performance analysis of an Internet-based system.

The front-end layer (presentation-application) is based on the one hand on web servers and on application servers on the other. The presentation server is supposed to provide the customer with a system offer. The task of the application server is to manage the transactions. Modern application servers provide the cluster functionality, which allows the Load Balancing (LB). In case of any server failure, the request can be automatically forwarded to another node. The system architecture includes many servers to ensure greater efficiency of this layer. In case of complex systems, network services as LB [1] or Domain Name Server are often used to distribute traffic. Software including these two functions in one layer (Tier 1 and 2) is for instance: Active Server Pages, Java Server Pages or PHP hypertext pre-processor.

The back-end layer (data) contains one or more databases (replication). This layer stores the systems data, and access to them is realized through transactions. Replication is applied:

1) when there are several different databases and data from them are replicated to a central database.
2) in order to speed up processing. It is profitable only if the database record processing is longer than the time needed to copy the processed records [7].

3 Performance Analysis

In this chapter we analyse performance metrics for the selected models of IISDCO architectures, such as the following layered models: **basic** (Model I), **front-end cluster** (Model II), **front-end and back-end cluster** (Model III) and **front-end cluster and**

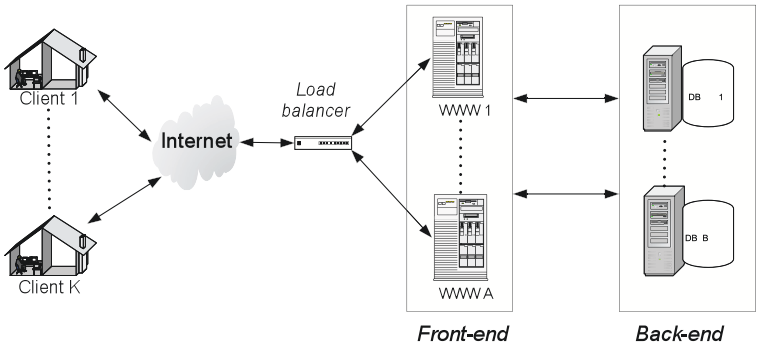

Fig. 1. Internet system layers.

back-end cluster with replication (Model IV). The purpose of the models analysis is to demonstrate their suitability for modelling Web systems architectures. The analysis is based on TCPNs simulation studies.

To present the system behaviour in various configurations, we present the selected simulation results, and adopt the following course of study. The characteristics of the arriving requests intensity with the same parameters are used for the simulation models [9] [10]. The process of requests arrival to the system is modelled by exponential distribution with the λ parameter, and throughput of service units by exponential distribution with μ parameter. The simulation is limited to three values for the exponential distribution. The main configuration of modeled system is determined by A – the number of nodes in the front-end layer and B – the number of nodes in the back-end layer:

1) case 1 model I ($A = 1$, $B = 1$),
2) case 2 model II ($A = 2$, $B = 1$); case 3 model II ($A = 4$, $B = 1$),
3) case 4 model III ($A = 2$, $B = 2$); case 5 model III ($A = 4$, $B = 2$); case 6 model III ($A = 4$, $B = 4$),
4) case 7 model IV ($A = 2$, $B = 2$); case 8 model IV ($A = 4$, $B = 2$); case 9 model IV ($A = 4$, $B = 4$).

All systems have the same value on the exponential service parameter $\mu = 100\,[1/s]$. The study will encompass two performance parameters: queues length and response times in different layers. The final element of the performance analysis is a summary of the simulations results at the same simulation time, $100000\,[s]$.

3.1 TCPN Simulation Models

In our elaboration we created system models using TCPNs [8], which allow the performance analysis. We verified later the constructed models with the real experimental environment as a benchmark. During the simulation all queues were monitored, and in the first part of the research we examined their length.

Servers of the front-end layer are modelled by the use of the Processor Sharing (PS) queuing systems. `PS1_A` present a processing unit (`PS1_A` stands for an *A*-umpteenth queue system PS modelling a processing unit of the front-end layer). The back-end server presents the following queues: `PS2_B` modelling a processing unit, and `FIFO2`, cooperating with it and modelling a server disk (`FIFO2_B` stands for a *B*-umpteenth queue system FIFO modelling a storage data element in back-end layer).

Model I In case 1 the growing number of requests causes an increase of response waiting time. In the front-end and back-end layers we observe the increasing length of queues. Increasing the number of requests in the web-based system causes an increase in the response time as well. The client does not accept this time. For model I, in any case, unbalance is observed.

Case 1 (Fig. 2) with $\lambda = 100\,[1/s]$ illustrates a growth of `PS1_1` queue. It leads to system unbalance, although the length of the back-end layers queues does not grow. Case 1 and $\lambda = 300\,[1/s]$ illustrates further increase of the `PS1_1` queue. The effect is the same as in the previous case, i.e. the system unbalance. The lengths of the back-end layer queues are on average about 1.5 for the PS2 and about 1.3 for `FIFO2`. There is some growth, but it is not critical for the balance of the system. Case 1 with $\lambda = 500\,[1/s]$ illustrates the front-end layer overload (by continuing to increase the length of `PS1_1` queue to infinity). The effect is the same as in previous cases the whole system unbalance. This time, the lengths of the back-end layer queues are on average about 6.9 for `PS2` and about 6 for `FIFO2`, and they keep growing.

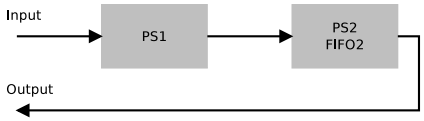

Fig. 2. Base model Model I for $A = 1$ and $B = 1$.

Figure 3 presents the response times of individual layers of model I and three different types of load. Figure 3 illustrates this unbalance in the front-end layer.

Cases for model II Analysis of the queues length (for a 2- and 4-node cluster installed in the front-end layer) in case of $100\,[requests/s]$ would suggest that the problem is solved. However, in case of $300\,[requests/s]$ the front-end queues are extended. The same happens with `PS2` queue; its length increases almost ten times. For $500\,[requests/s]$ the system balance is completely disturbed. Number of requests in the `FIFO` queue, especially in the case of $500\,[requests/s]$, begins to grow alarmingly. A 4-node cluster in the front-end layer copes somewhat better with the growing number of requests. In the case of $500\,[requests/s]$ the response waiting time is extending. In the case of a front-end cluster layer model the system unbalance occurs for $500\,[requests/s]$ both for the cluster $A = 2$ (Fig. 4a) and $A = 4$ (Fig. 4b). It results in an extension of all PS queues in both layers.

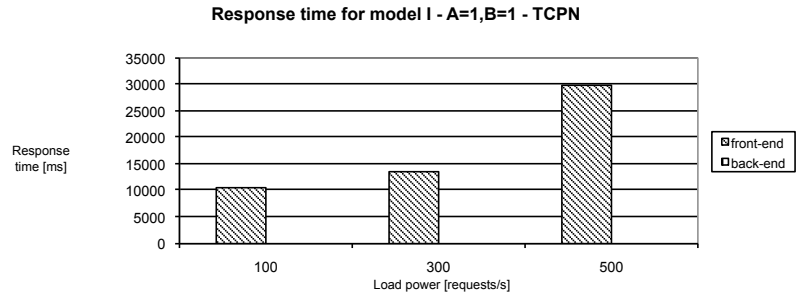

Fig. 3. Response time for model I case 1 - (front-end and back-end layer) with three load types.

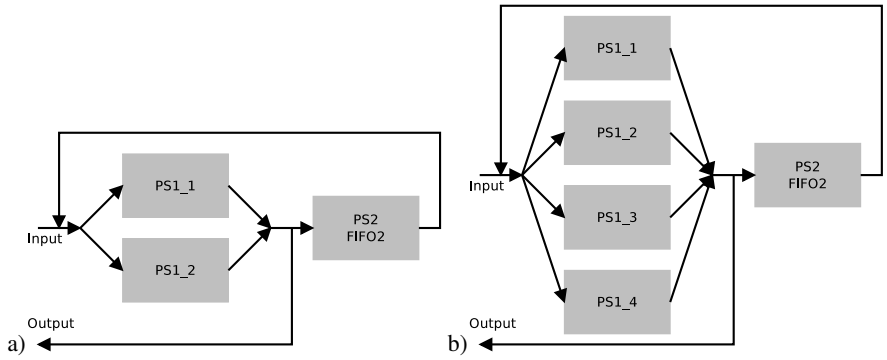

Fig. 4. Cases for model II for (examples): a) $A = 2$, $B = 1$, b) $A = 4$, $B = 1$.

Characteristics for case 2 with $\lambda = 100\,[1/s]$ indicate the system balance. The average queue length does not exceed the value of several pending requests. Figure 5 presents case 2 characteristics for $\lambda = 300\,[1/s]$, showing extension of the queue for this model. This is particularly evident in the case of `PS2` queue length (Fig. 5c). However, it does not indicate the system unbalance. Increasing the load did not involve a loss of balance. The front-end layer characteristics (Fig. 5a,b) present a balance in the average length of almost several requests in the queue. The `FIFO2` queue behaves similarly (Fig. 5d). Characteristics of case 2 with $\lambda = 500\,[1/s]$ illustrate loss of balance. All queues grow up to infinity. Only the value of the back-end layer `FIFO` queue is smaller. Another load increase results in a loss of balance.

Characteristics for case 3 with $\lambda = 100\,[1/s]$ illustrate the balanced system. Number of requests waiting in queues does not exceed 10 in this case. Characteristics for case 3 with $\lambda = 300\,[1/s]$, present unbalance in spite of increasing the number of servers in the front-end layer. The situation is similar to case 2 for $A = 2$. Increasing the number of cluster elements led to `PS2` queue length shortening. Characteristics for case 3 with $\lambda = 500\,[1/s]$ present system unbalance. Reducing the queues length results in an improvement with relation to the same model for the case $A = 2$, but the unbalance remains.

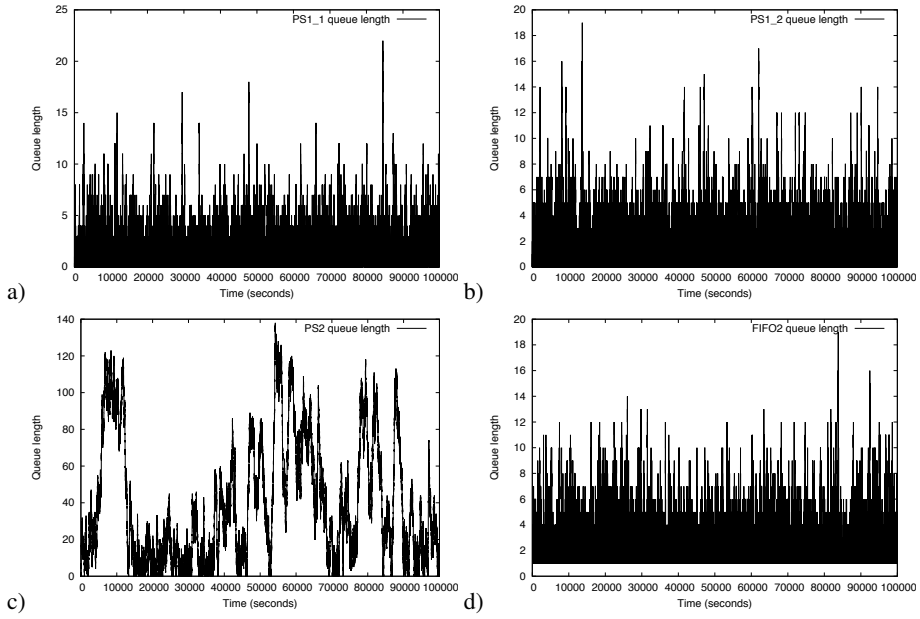

Fig. 5. Sample queue length history for case 2 ($300\,[requests/s]$): a) PS1_1, b) PS1_2, c) PS2, d) FIFO2.

The graphs in (Fig. 6) present the response times of individual layers for model II with three different load values $\lambda = 100\,[1/s], 300\,[1/s], 500\,[1/s]$. The graph in Fig. 6a presents a case when $A = 2$, while the graph in Fig. 6b presents a case when $A = 4$. We can observe the response time shortening with the largest load applied.

Cases for model III Another model to be analysed is model III (Fig. 7). We can see the performance improvement for the front-end cluster, but only in case when $\lambda = 500\,[1/s]$. For case 4 the lengths of `PS1_A` queues grow along with the requests number increase. In case 5 `PS1_A` queues lengths are at a similar level for 100 and $300\,[requests/s]$ and grow almost 100 times for $500\,[requests/s]$. In case 6 `PS1_A` queue lengths grow steadily several times for each of the three levels of load. This increase for $500\,[requests/s]$ causes unbalance. Requests load of `PS2_B` queues for the case when $B = 2$ and for the case when $B = 4$ is maintained at the same level. They indicate the system unbalance. For the case when $B = 4$, the number of pending requests in the queues was about half as much. In `FIFO2` queue the number of requests does not exceed 8 in any case. As a result of waiting for the requests realization in queues `PS2_B` the response time increases. Applying a similar number of nodes in the two layers makes the length of all queues shorten (in case of 4 nodes). In this instance the system unbalance is not as clear as for $100\,[requests/s]$. The system unbalance occurs in all other cases.

Characteristics of case 4 (Fig. 7a) with $\lambda = 100\,[1/s]$ show the system unbalance. Especially PS queues in the back-end layer present it. The same happens for case 4 with

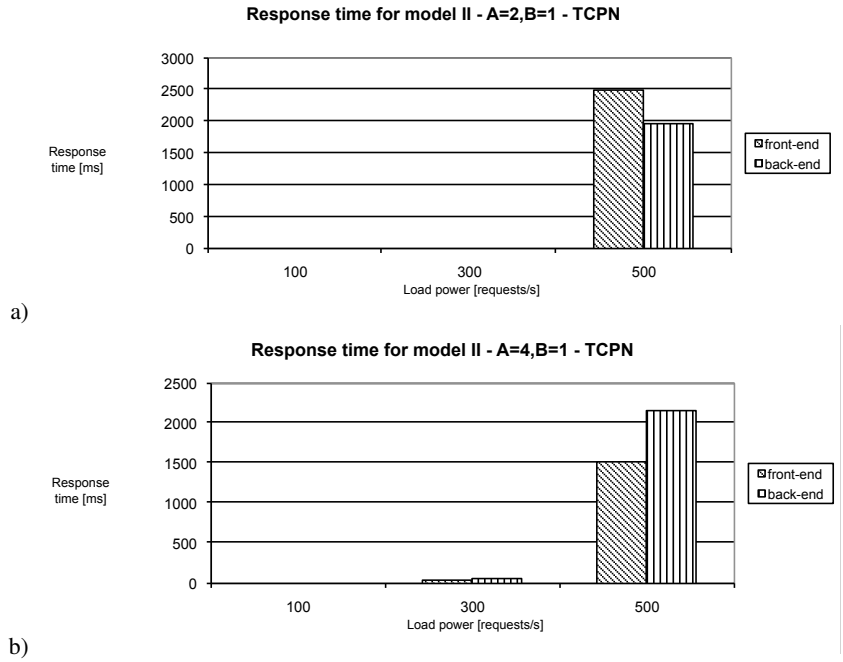

Fig. 6. Response time for model II (front-end and back-end layer) with three load types: a) case 2, b) case 3.

$\lambda = 300\,[1/s], 500\,[1/s]$. This time, PS queue of both system layers presents unbalance. A similar situation occurs in all model III cases (Fig. 7b).

Characteristics of case 6 (Fig. 7c) with $\lambda = 100\,[1/s]$ present the system unbalance. PS queue in the back-end layer presents unbalance, yet reduced almost ten times as compared to an identical model III with $B = 2$. Characteristics of case 6 with $\lambda = 300\,[1/s]$ present the system unbalance. PS queues in the back-end layer present unbalance. There can be observed the queues length reduction in comparison to an identical model III for $B = 2$. Characteristics of case 6 with $\lambda = 500\,[1/s]$ present the system unbalance, illustrated by PS queues in both layers, in another way as it was with the same model for the same parameters, except for $B = 2$. Concluding the analysis it seems advisable to use a similar number of nodes in both layers. Graphs in Fig. 8 present the response times for model III in particular layers with three different values of load. The graph in Fig. 8a presents case 4, the graph in Fig. 8b presents case 5, while the graph in Fig. 8c presents case 6. We can see extention of the response time in both layers, using the largest load. The response time of the front-end layer for the cases in (Fig. 8a,b) with $\lambda = 500\,[1/s]$ causes the system unbalance. With the largest load for the case in Fig. 8c the queues show system unbalance. But the queues are shorter.

Cases for model IV In model IV, only cases with 4 front-end and 4 back-end nodes are balanced. In case 7 (Fig. 7a) `PS1_A` queues lengthen as the number of requests

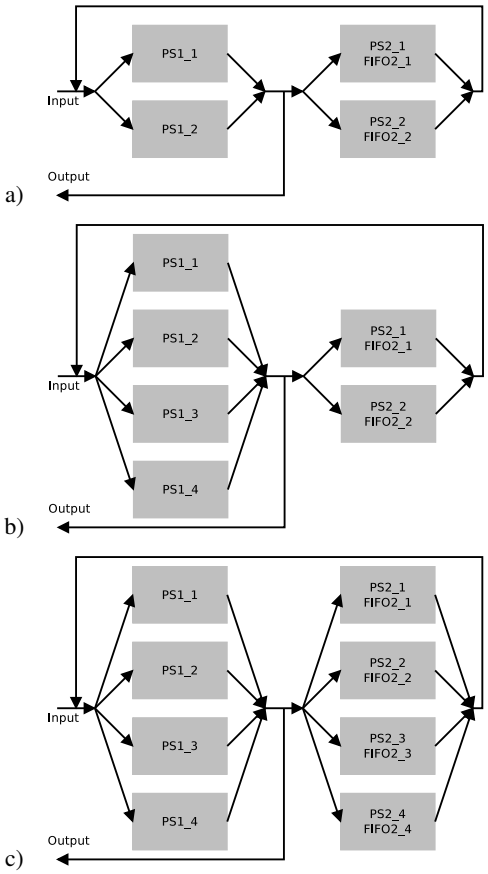

Fig. 7. Model III and IV for (examples): a) cases 4 and 7, b) cases 5 and 8, c) cases 6 and 9.

increases. It presents a system unbalance. In cases 8 (Fig. 7b) and 9 (Fig. 7c) `PS1_A` queues lengths grow as well. However, in the case when $A = 4$ and $B = 4$, the number of requests waiting in the queue is not increasing that much. The number of requests in `PS2_B` queues remains on the level of both layers cluster model. For `FIFO2_B` queues the number of requests does not exceed 5 in any load case. The analysis of the requests number in the queues indicates the still inadequate level of requests service. In case 9 for $500\,[requests/s]$ the growth is not that large, when compared to $300\,[requests/s]$.

In model IV we can observe an opposite situation to that in previous cases. `PS2_B` queues are here the major reason for the system unbalance. Yet, despite the increasing load, the Internet system remains balanced as the number of nodes in the back-end layer grows. This behaviour is similar to model III. In this case we can observe reduction of number and equalization of the requests load in queues.

Characteristics of case 7 with $\lambda = 100\,[1/s]$ present the system unbalance (`PS2_B` queues). Characteristics of case 7 with $\lambda = 300\,[1/s], 500\,[1/s]$ provide further system unbalancing (PS queues in two layers). A similar situation occurs in case 8 with $\lambda =$

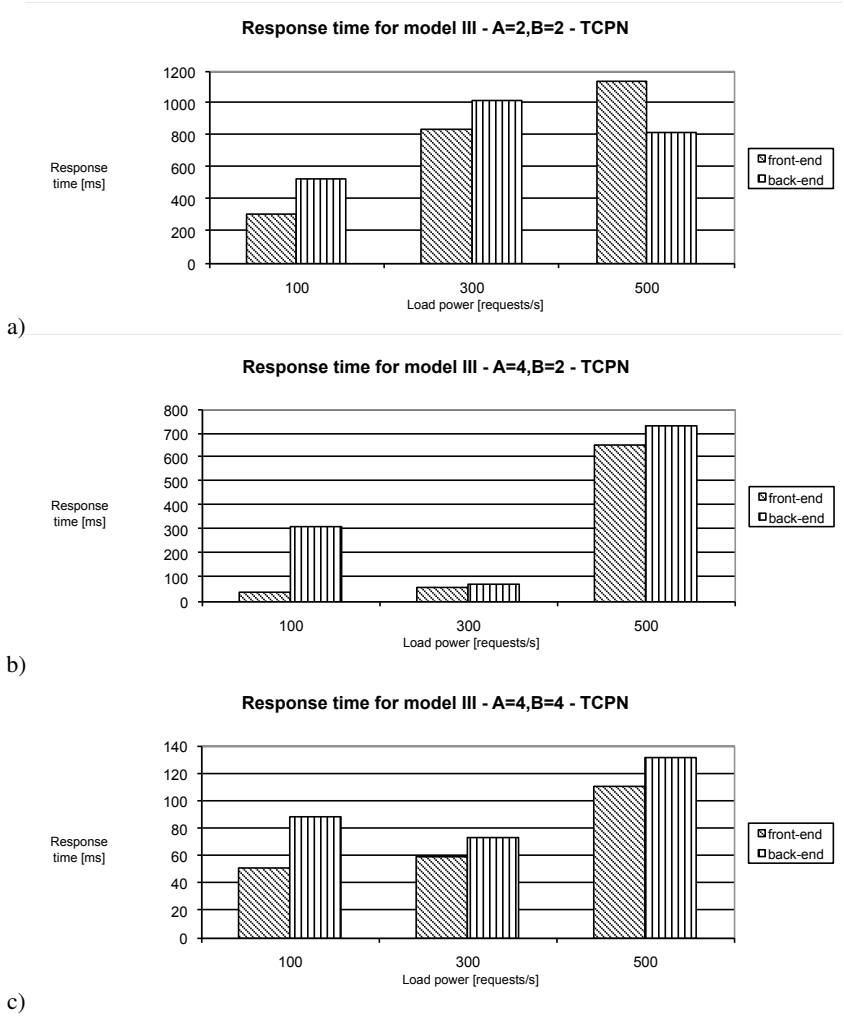

Fig. 8. Response time for model III (front-end and back-end layer) with three load types: a) case 4, b) case 5, c) case 6.

$100\,[1/s], 300\,[1/s], 500\,[1/s]$. Characteristics of case 9 with $\lambda = 100\,[1/s]$ provide the system unbalance, but it is not so clear as in other model IV cases. Characteristics of case 9 with $\lambda = 300\,[1/s]$ provide the system unbalance, which is illustrated by `PS2_B` queues. In this case, the back-end layer queues are shorter in comparison to a corresponding case of model III. Figures 9 and 10 present model IV characteristics for $A = 4$, $B = 4$ and $\lambda = 500\,[1/s]$. Characteristics of case 9 with $\lambda = 500\,[1/s]$ represent the aspirations for balancing the system. The length of `PS2_B` queues (Fig. 9e, 10g,i,k) is two times shorter than for case 6 and $\lambda = 500\,[1/s]$ in model III. It is true both for `PS2_B` (Fig. 9a,b,c,d) and `FIFO2_B` queues (Fig. 9f, 10h,j,l).

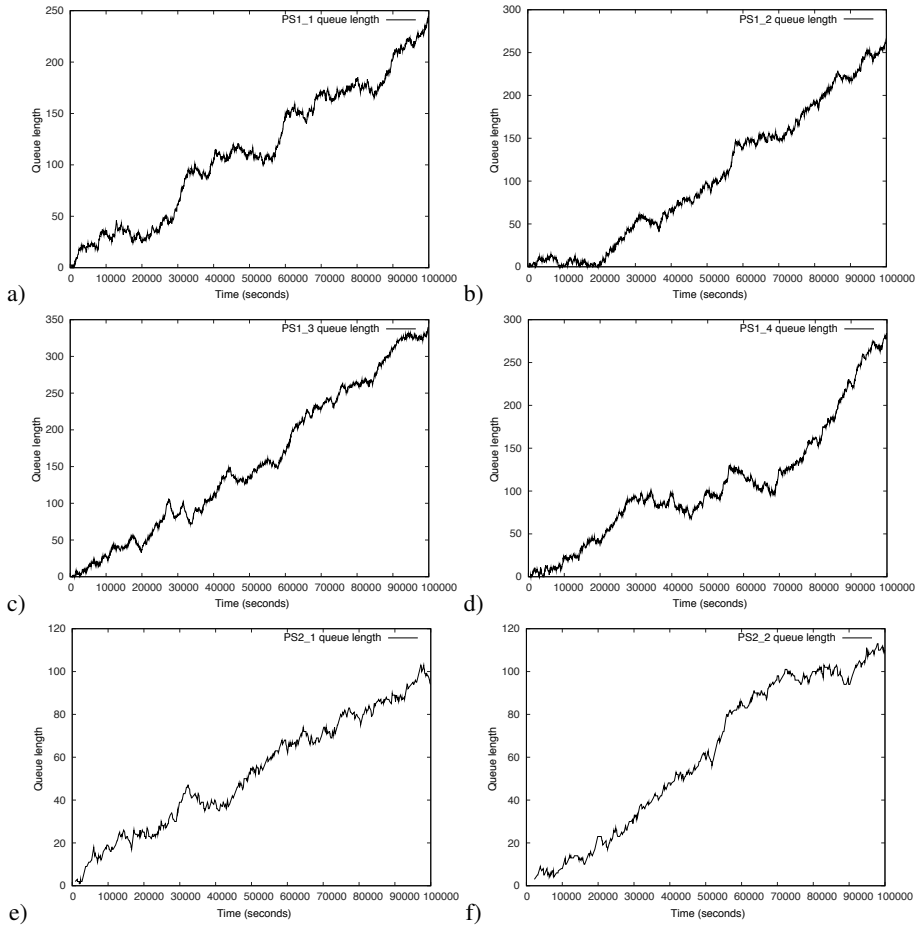

Fig. 9. Sample queue length history for case IV with $500[requests/s]$: a) PS1_1, b) PS1_2, c) PS1_3, d) PS1_4, e) PS2_1, f) PS2_2.

Graphs in (Fig. 11) present the response times of model IV particular layers with three different load values: $\lambda = 100\,[1/s], 300\,[1/s], 500\,[1/s]$. The graph in Fig. 11a presents case 7, and the graph in Fig. 11b presents case 8, while the diagram in Fig. 11c presents case 9. There is a gradual shortening of the response time in layers for the largest peak load applied.

The presented detailed analysis of TCPN models illustrates trends that occur in cases when the system load is increasing. We should notice improvement in the system balance when using parallel processing (cluster). Distributing the requests among more nodes results in the improvement of system performance. On the other hand, using replication in the back-end layer does not cause such clear improvement, although it is a good solution, as can be observed particularly in case of increasing load. Graphs in figure 12 present a review of all behaviour patterns for particular loads. Model I has

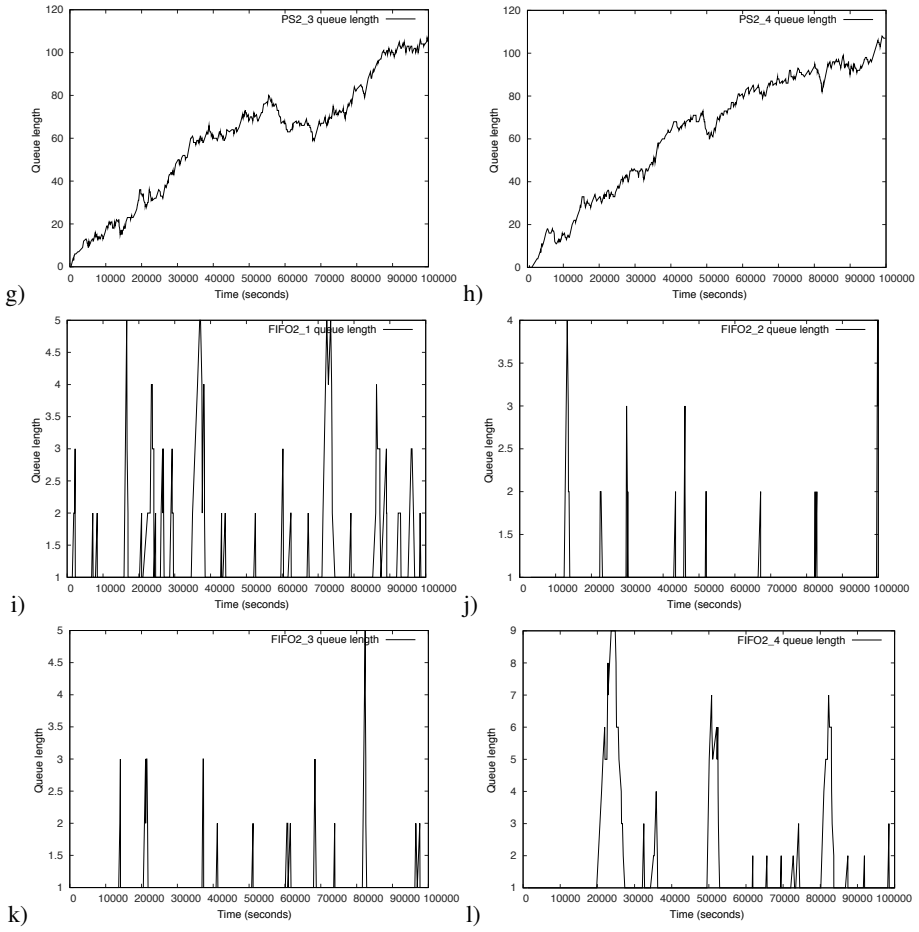

Fig. 10. Sample queue length history for case IV with $500[requests/s]$: g) PS2_3, h) PS2_4, i) FIFO2_1, j) FIFO2_2, k) FIFO2_3, l) FIFO2_4.

the longest response time that increases further on along with the load increase. The response times shape in the same way with declining tendency for models with clusters and replication (Fig. 12a,b,c).

3.2 Analysis Summary

Detailed conclusions from the IISDCO models analysis determine particular reactions to the load. Increasing the number of nodes in the front-end layer (using the cluster) results in accelerating the requests realization. For smaller loads such solution gives positive results. But the increase of requests number prolongs the response time. Using a cluster in the front-end and back-end layers results in shortening the response time in the front-end layer and lengthening it in the back-end layer. This disadvantageous state

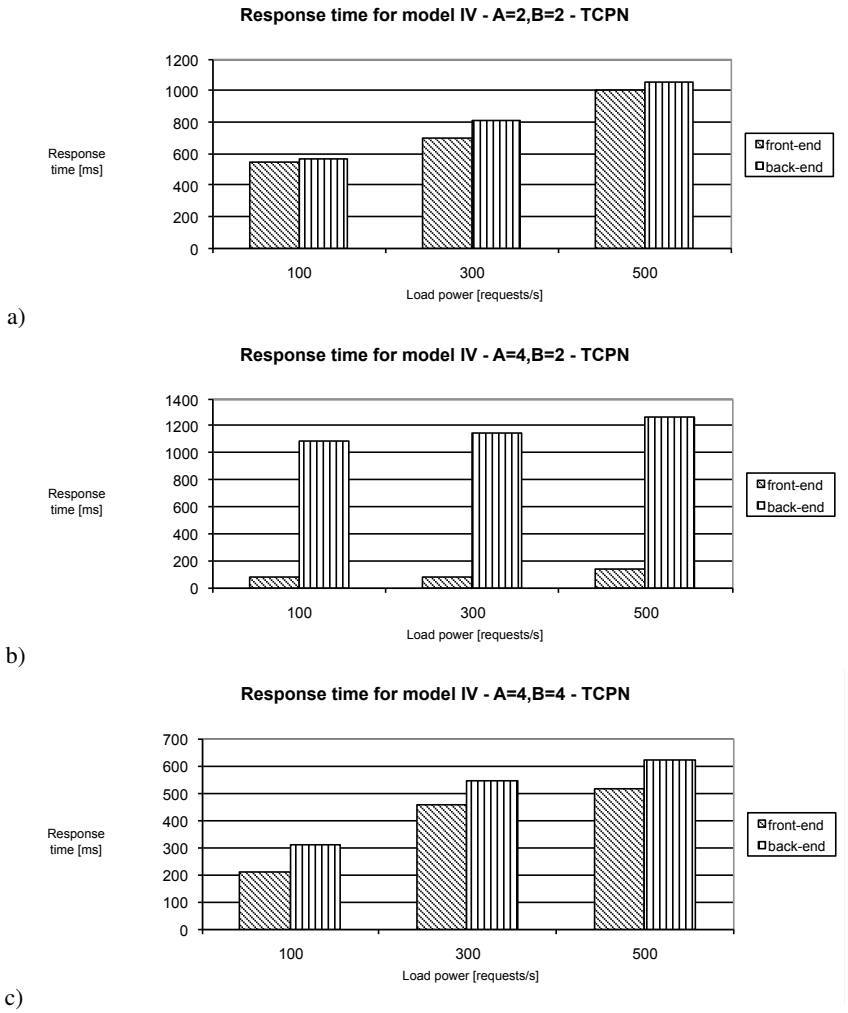

Fig. 11. Response time for model IV (front-end and back-end layer) with three load types: a) case 7, b) case 8, c) case 9.

can be somewhat improved by enlarging the back-end layer cluster, yet the changes are not very significant. Only using a similar number of nodes in the back-end cluster/replication layer as in the front-end layer does provide a more satisfactory result in all cases. The use of replication implies the need for synchronization. The cost of synchronization is smaller than the profit obtained owing to additional service nodes. The load of each node is increased slightly, but on the whole the system reduces the response time by distributing the requests evenly. In cases where the unbalance is observed, there occurs e.g. a very short queue length of waiting requests in one layer and a very long one

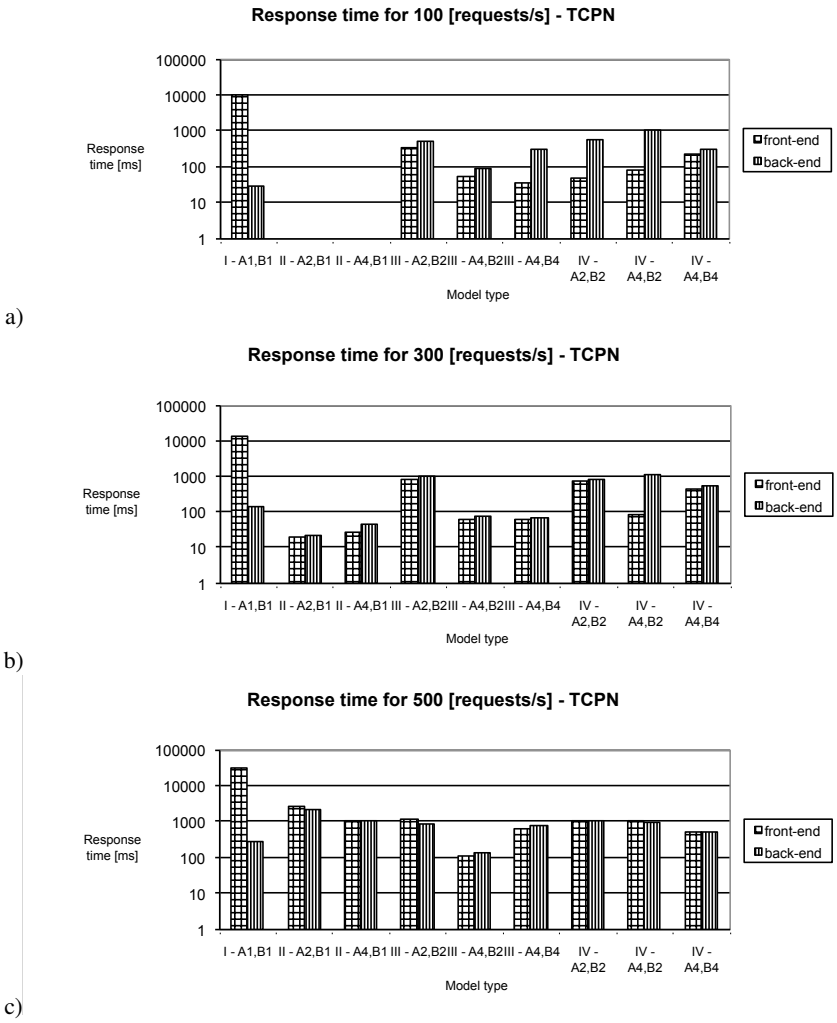

Fig. 12. Response time for all tested model cases (front-end and back-end layer) with three load types: a) $100[requests/s]$, b) $500[requests/s]$, c) $500[requests/s]$.

in another layer at the same time. The lack of balancing is associated with irregular distribution of loads in layers and between them.

The systems balance in the case of increasing load, is disturbed. Further increasing of the external load only worsens the performance. We could of course modify the hardware elements, yet this solution would mean greater costs and a necessity to exchange whole servers due to the devices incompatibility, for instance. We should rather concentrate on decentralised systems that are more natural for the Internet systems. It is assumed that to construct the systems identical computers were used. The effort was put into determining how they are connected. Basing on this analysis, it seems ap-

propriate to apply the systems modification without interfering either in the computer construction or in software. In order to achieve systems balancing, it is necessary to:

1) use layers in the construction of systems, in order to balance the processed requests,
2) select the number of nodes in the cluster structured front-end layer, according to the actual load,
3) use replication with the number of nodes more than half the number of the front-end nodes.

4 Conclusion

The results obtained in models simulation, using TCPN and CSIM, were compared with experiments in Real On-line Internet Stock Exchange System (ROISES). ROISES is a simplified stock exchange Internet system constructed as a part of the IISDCO systems research. This software with appropriately selected parameters presents an example of IISDCO. In order to allow comparison of the developed models with the experiment results, a specific clients load stream was necessary to be used. The arrival stream was the same as the load used in the models. In addition, it was necessary to adjust the service time.

Experiments of the Internet stock reference model were limited to 4 nodes in the front-end layer and 2 nodes in the back-end layer due to the limited amount of equipment. The real system tests up to this moment are identical with the performance simulation models experiments in most cases. It was proved that the IISDCO performance models are correct. Despite the simplifications made in the models and on the assumptions given, an analysis reflecting the modelled reality can be provided. For layers response times the error remains within 15%. This is not a perfect model, but the error does not disqualify it in any way [4].

The advantages of this approach are:

1) verification of simulation models,
2) application of TCPN quantitative analysis,
3) using performance simulators,
4) the possibility of models expansion,
5) ability to monitor system elements during the simulation and experiments.

The obtained results are the basis for further study, concerning more complex processing models. Owing to the possibility of models further development, it will be possible to verify the system behaviour in case of a higher number of nodes and with greater input load. The preliminary results show that adequate modification of queuing systems parameters can produce acceptable level of compatibility between models and real systems.

References

1. Aversa, L., Bestavros, A.: Load Balancing a Cluster of Web Servers Using Distributed Packet Rewriting. In: Proc. of the 2000 IEEE International Performance, Computing, and Communications Conference, pp. 24–29. Phoenix, AZ (2000)

2. Kounev, S.: Performance Engineering of Distributed Component-Base Systems, Banchmarking, Modeling and Perrformance Prediction. Shaker Verlag (2006)
3. Kounev, S., Buchmann, A.: Performance Modelling of Distributed E-Business Applications using Queuing Petri Nets. In: Int. Symposium on Performance Analysis of Systems and Software, pp. 143–155. IEEE (2003)
4. Menasce, D.A., Almeida, V.A.F., Dowdy, L.: Performance by Design. Prentice Hall (2004)
5. Pai, V.S., Aront, M., Banga, G., Svendsen, M., Druschel, P., Zwaenpoel, W., Nahum, E.: Locality-aware Request Distribution in Cluster-based Network Servers. In: Proc. of 8th ACM Conf. on Arch. Support for Progr. Languages, vol. 32, pp. 205–216. San Jose (1998)
6. Rak, T.: The Modeling an Analysis of Interactive Internet Systems Realizing the Service of High-Frequency Offers (in Polish). PhD dissertation supervised by J. Werewka, Krakow, AGH-UST (2007)
7. Rak, T., Swider, K.: Database Replication in Practice (in Polish). In: Data-base Systems. Structures, Algorithms and Methods, pp. 153–162. WKŁ, Warsaw (2006)
8. Rak, T., Samolej, S.: Distributed Internet Systems Modeling Using TCPNs. In: Proc. of the Int. Multiconf. on Computer Science and Information Technology, pp. 559–566 (2008)
9. Design/CPN homepage, http://www.daimi.au.dk/designCPN
10. CPN Tools homepage, http://cpntools.org