ELECTRONICS AND ELECTRICAL ENGINEERING

ISSN 1392 – 1215 *2006. Nr. 6(70)*

ELEKTRONIKA IR ELEKTROTECHNIKA

TELECOMMUNICATION ENGINEERING

T 180

TELEKOMUNIKACIJŲ INŽINERIJA

Numerical Model of Transmission Control Protocol

H. Pranevičius, T. Kirvėlaitis

Business Informatics Department, Kaunas University of Technology
Studentu str. 56, LT-51424, Kaunas, Lithuania, phone: +370 37 300376; e-mail: hepran@if.ktu.lt

I. Pranevičienė

Department of Applied Mathematics, Kaunas University of Technology,
Studentu str. 50, LT-51368, Kaunas, Lithuania, phone: +370 37 300318; e-mail: irena.praneviciene@ktu.lt

Introduction

Transmission Control Protocol [1, 2] (TCP) is oriented to connections. When a transmitter forms a message it sends a request of logical connection making and it sends data only after the confirmation that the receiver is ready to accept it. TCP ensures data transfer, one waits for the confirmation after sending data, and if there is no confirmation the data are sent once more. This protocol ensures the package delivery in the correct order as well, i.e. in the same way as they were sent. The transmitter can be stopped temporally in order to avoid data loss due to the overflow in the transmitter's buffer. The User Datagram Protocol (UDP) protocol is not oriented to connections; it does not ensure data transfer, one does not require the confirmation and data is not sent once more, so if the transmitter's buffer is overloaded then packages are lost. The UDP protocol uses less control information. The size of the header of the TCP protocol is 20 bytes, and the one of the UDP protocol is 8 bytes.

The simulation model of the TCP protocol is presented in the paper [3], the delay of packages is analyzed here as well. This model seeks to estimate the duration of the TCP transfer start, i.e. the logical connection making, as well to estimate the duration of transferring a certain amount of information. Delayed confirmations influence the duration of information transfer negatively. If the receiver accepts data without errors and the confirmation to the transmitter arrives too late, data is already sent once more. The paper presents the results of the duration of transferring a certain amount of information, the duration of connection making in this transfer, the number of packages that are repeated after receiving a delayed confirmation.

The paper [4] analyzes the losses of data transferring by UDP protocol. More than 80% of information in WAN networks is transferred using the TCP protocol. The UDP protocol is used for transferring information in real time and for some control data. The volume of packages' losses using the UDP protocol depends on the size of the UDP package, the transfer speed and the transfer distance. Simulation model development software *Real 4.0* was used for obtaining the simulation results. The simulation results demonstrated that it is more efficient to transfer data using the UDP protocol in smaller packages, in such case less data is lost.

The TCP protocol digital model is presented in the paper [5]; losses of packages are analyzed here with the condition that packages can be lost only in the direct direction, i.e. in the sender – receiver direction, and the confirmations are not lost.

The efficiency of the TCP protocol depending on the network load conditions are analyzed in the paper [6]. A simulation model was created in order to perform analysis and to obtain results. As the network load increases the protocol will transfer data more efficiently if it sends shorter data packages, in the opposite case the packages spend a lot of time in service queues.

For the development of the TCP protocol model an automated model development system MSM for systems that are described by Markovian processes with a discrete set of states and a continuous time [7] is used. The theoretical approach of developing models of this class is presented in [8, 9]. MSM software enables to automate the main stages of numeric model development: a textual specification of a system, creation of equations that describe the system functioning; solving equations that describe the system functioning; calculations of system characteristics.

Computing Algorithm

Computation of stationary probabilities of a Markovian process by numerical methods is executed solving a system of linear equations. The methods applied in solving the system of linear equations can be classed into two groups: direct and iteration ones. Applying a direct numerical methods for solving a system of equations during the reduction phase, the elimination of one non-zero element of the matrix often results in the creation of several non-zero elements in positions which previously contained a zero. This is called fill-in and not only does it make the organization of a compact storage scheme more

difficult, since the provision must be made for the deletion and the inclusion of elements, but in addition the amount of the fill-in can often be so extensive that available memory is quickly exhausted.

Iterative methods have traditionally been preferred to direct methods. However, iterative methods have a major disadvantage in that they often require a very long time to converge to the desired solution. For certain classes of problems, direct methods often result in a much more accurate answer being obtained in less time. It was observed that although the direct method required more memory for storing arrays than the iterative method, it obtains more accurate results in a considerably shorted time.

A successful direct method must incorporate a means of overcoming the difficulties mentioned above as well as other ones. A new method for automatic creation of numerical models of systems represented by Markov processes is proposed below.

The method of sequential embedding of Markov chains is a promising one for calculating stationary probabilities. The essence of the method is the probability interpretation of the elements of the matrix of a set of linear equations. The normalized elements of the matrix of a set of linear equations denote the probabilities of Markov chain transitions, describing the system functioning. Reduction of the size of the set of equations is done by a sequential elimination of the Markov chain states and a corresponding recalculation of transition probabilities.

Let $P_N = \left(p_{ij}^{(N)}\right)_{NxN}$ be the probability matrix of a transition of the Markov chain $\left\{x_m^{(N)}, m \geq 0\right\}$, with a set of states $X = \{x_1, x_2, \ldots, x_N\}$. The stationary probabilities of on embedded Markov chain are determined by the system of linear equations:

$$\overline{p_i} = \sum_{j=1}^{N} \overline{p_j} p_{ji}, \; i = \overline{1, N}, \tag{1}$$

where

$$p_{ji} = \lambda_{ji} / \sum_{j=1}^{N} \lambda_{ji}, \; i, j = \overline{1, N}.$$

Computation of the stationary non-normalized probabilities $\overline{p_i} = \overline{p}_i^{(N)}$, $i = \overline{1, N}$, involves two stages: the stage of reducing the number of equations of the set (that of embedding Markov chains) and the one of computing the stationary probabilities.

The algorithm has the following form:

1. The stage of embedding Markov chains:

$$p_{ij}^{(k)} = p_{ji}^{(k+1)} + p_{i,k+1}^{(k+1)} \cdot \frac{p_{k+1,j}^{(k+1)}}{1 - p_{k+1,k+1}^{(k+1)}}, \tag{2}$$

$i, j = \overline{1, k}, \; k = \overline{N-1, 1}$.

2. The stage of computing the stationary probabilities:

$$\overline{p}_1^{(1)} = 1; \tag{3}$$

$$\overline{p}_i^{(k+1)} = \begin{cases} \overline{p}_i^{(k)}, & i = \overline{1, k}, \\ \dfrac{\sum_{j=1}^{k} \overline{p}_j^{(k)} p_{ji}^{(k+1)}}{1 - p_{ii}^{(k+1)}}, & k = \overline{1, N-1}; \; i = k+1; \end{cases} \tag{4}$$

$$p_j = p_j^{(N)} = \frac{\overline{p}_j^{(N)}}{\sum_{j=1}^{N} \overline{p}_j^{(N)}}, \; j = \overline{1, N}, \tag{5}$$

where p_j is the normalized probability.

The stationary probabilities of Markov process are found from the formula:

$$q_i = \frac{p_i \Big/ \sum_{j=1}^{N} \lambda_{ij}}{\sum_{j=1}^{N} \left(p_j \Big/ \sum_{i=1}^{N} \lambda_{ji} \right)}, \; i = \overline{1, N}. \tag{6}$$

Formal Description of TCP Protocol

Conceptual Model

TCP works in 4-5 OSI levels. It is responsible for decomposing messages into packages, for transferring and an appropriate rebuild of messages from packages. The protocol is oriented to connections; it works in a duplex mode.

The protocol model consists of five aggregates: the sender, the receiver and three intermediate devices – *routers*. When the sender forms a message it sends a request of logical connection making. After the confirmation about the connection is received, i.e. the confirmation that the receiver is ready to accept data, only then the sender sends data message.

The packages of the same message can reach the receiver through different paths, i.e. through different routers. The TCP protocol is based on the routing protocol information that estimates the occupancy of routers by sending a package through a certain path.

The TCP protocol ensures the package delivery in the correct order. Every TCP package has an assigned sequence number in a concrete message. If the packages reached the receiver in the order that is different from the one of sending, the receiver recovers the message in the correct order according to the numbers of the packages.

After sending the message the sender saves its copy and the timer of confirmation. If there switches-on is no confirmation, the message is repeated. The receiver sends a confirmation only if the message was accepted without errors.

Every device checks the control sum of packages; if there are some errors, the package is eliminated in order not to load the network wastefully. If there is no confirmation the sender sends the packages once more

In order to avoid the overload of the sender's buffer, the confirmation package has a section where one defines if it is possible to send packages to the receiver. If the receiver's buffer approaches the limit of overload, the

sender is stopped temporally in order to avoid the loss of packages. When the sender receives the information that the receiver's buffer is free, it sends messages again.

Aggregate Model

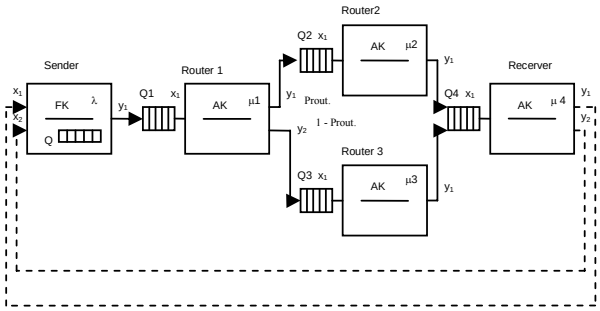

Fig. 1. Aggregate Scheme of TCP Protocol

Parameters:

λ – the intensity of message generation;

$\mu_i, i = \overline{1,4}$ – the intensity of package service;

max_Q – a maximum capacity of the buffer;

PS – the overload sender (1 – buffer is overloaded, 0 – the buffer is not overloaded);

conf_time – confirmation waiting time;

messtype – message type (0 – connection executed, 1 – data is sending);

P_{rout} – the probability for estimating the load of routers on the basis of routing protocol information;

P_{CRC} – the probability of an error in the package.

Formal Specification of *Sender*

1. The set of input signals: $X = \{x_1, x_2\}$, (x_1 – the confirmation from the receiver about the connection making, x_2 – the confirmation from the receiver about receiving the message without errors).

2. The set of output signals: $Y = \{y_1\}$, (y_1 – the message is sent).

3. The set of external events: $E' = \{e'_1, e'_2\}, (e'_1$ – the receiving of the signal x_1, e'_2 – the receiving of the signal x_2).

4. The set of external events: $E'' = \{e''_1, e''_2\}$, $(e''_1$ – the end of message generation, e''_{2j} – the end of confirmation waiting time for the j – th message, $j = \overline{1,\infty}$).

5. Controlling sequences: $\{e''_1\} \to \{\xi_j\}, j = \overline{1,\infty}$, ($\xi_j$ a random value distributed according to the exponential law with the parameter λ).

6.The discrete state: $v(t) = \{Q(t), FK(t), conf\_timer(t)\}$.

$Q^{(t)}$ – the buffer of sender where messages wait for connection making. When the connection is made and the message is sent, the copies of the messages wait for the confirmation. When the confirmation is received the message is eliminated from the buffer $\left({}^{\#}Q \le \max\_Q\right)$;

FK– the state of message generation channel (1– occupied, 0 – free); *conf _ timer* – the timer's state (1– on, 0 – off).

7. The continuous state: $z_v(t) = \{w(e''_1, t), w(e''_{2j}, t)\}$,

$w(e''_1, t)$ - the end of message generation;

$w(e''_{2j}, t)$ - the end moment of confirmation waiting.

8. The state of the system: $z(t) = \{Q(t), FK(t),$

$conf\_timer(t), w(e''_1, t), w(e''_{2j}, t)\}$.

The initial state: $z(t_0) = \{Q(t_0), FK(t_0) = 0,$

$conf\_timer(t_0) = 0, w(e''_1, t_0) = t_0 + \xi_j,$

$w(e''_{2j}, t) = \infty\}$.

9. Transition and output operators:

$H(e'_1)$:

$$FK(t+0) = \begin{cases} 1, \text{ if } Q(t) < \max\_Q \text{ ir } FK(t) = 0, \\ FK(t), \text{ otherwise.} \end{cases}$$

If after the confirmation the generation of messages is stopped, the buffer is not overloaded, then the message generation is enabled again, new messages are generated.

Messtype = 1,

${}^{\#}Q(t+0) = {}^{\#}Q(t)$.

If the receiver's buffer is not overloaded, the message from the buffer is sent, the copy of the message is saved in it; if the buffer is overloaded the message is not sent for a while.

$$conf\_timer = \begin{cases} 1, \text{if } \ PS = 0, \\ 0, \text{if } \ PS = 1. \end{cases}$$

When the message is sent the confirmation waiting timer is switched-of.

The moment of the message generation end

$$w(e''_1, t+0) = \begin{cases} t + \xi_j, \text{ if } {}^{\#}Q(t) < \max\_Q \text{ ir } FK(t) = 0, \\ w(e''_1, t), \text{ otherwise.} \end{cases}$$

The moment of confirmation waiting time end

$$w(e''_{2j}, t+0) = \begin{cases} t + conf\_time, \text{ if } PS = 0, \\ \infty, \qquad\qquad \text{if } \ PS = 1. \end{cases}$$

When the confirmation about the connection making is received and if the receiver's buffer is not overloaded, the message is sent.

$H(e'_2)$:

$$FK(t+0) = \begin{cases} 1, \text{if } PS = 0 \text{ and } {}^{\#}Q(t) < \max\_Q, \\ \text{and } \ FK(t) = 0, \\ FK(t), \text{otherwise.} \end{cases}$$

When the confirmation is received and if the message generation is stopped, the inner buffer is not overloaded, the receiver's buffer is not overloaded, then the message generation is available again, new messages are generated.

${}^{\#}Q(t+0) = {}^{\#}Q(t) - 1$.

The copy of the message is eliminated from the buffer.

$conf\_timer(t+0) = 0$.

The confirmation timer is stopped.

The moment of the message generation end

$$w(e_1'', t+0) = \begin{cases} t+\xi_j, \text{if } {}^{\#}Q(t) < \max\_Q \text{ and } FK(t) = 0, \\ w(e_1'', t), \text{ otherwise.} \end{cases}$$

The end of confirmation waiting time

$$w(e_{2j}'', t+0) = \infty.$$

$G(e_2'): \varnothing$.

$H(e_1'')$:

A new message is generated when the number of messages in the buffer does not exceed $\max\_Q - 1$, thus the overload of the buffer is avoided. When the buffer of the sender reaches the limit of the overload, the generation of new messages is stopped for a while.

$$FK(t+0) = \begin{cases} 1, \text{if } Q(t) < \max\_Q - 1, \\ 0, \text{if } Q(t) = \max\_Q - 1. \end{cases}$$

$messtype = 0$,

${}^{\#}Q(t+0) = {}^{\#}Q(t) + 1$.

The number of the messages in the buffer increases.

The moment of the message generation end

$$w(e_1'', t+0) = \begin{cases} t+\xi_j, \text{if } Q(t) < \max\_Q - 1, \\ \infty, \quad \text{if } Q(t) = \max\_Q - 1. \end{cases}$$

When a new message is generated the connection making package is sent. The message will be sent only when the confirmation about the connection making is received.

$H(e_{2j}'')$:

messtype = 1,

${}^{\#}Q(t+0) = {}^{\#}Q(t)$.

$$conf\_timer(t+0) = \begin{cases} 1, \text{ if } PS = 0, \\ 0, \text{ if PS} = 1. \end{cases}$$

If there is no confirmation, the copy of the message remains in the buffer and the sending is repeated, the timer is switched-on again if the receiver's buffer is not overloaded.

The moment of confirmation waiting time end

$$w(e_{2j}'', t+0) = \begin{cases} t + conf\_time, \text{ if } PS = 0, \\ \infty, \quad \text{if } PS = 1. \end{cases}$$

$G(e_{2j}'')$: y_1, if $PS = 0$.

Simulation Results of TCP Protocol

The TCP protocol model was created using digital model development software MSM.

Such characteristics of the TCP protocol are analyzed in this paper:

- The loss size $P_{n\ sist.}$;
- The average length of package queue in the receiver $\overline{N_{q\ 4}}$;
- The average duration of the stay of a package in the system $\overline{T_s}$;
- The average duration of the package's waiting in the service equipment queues $\overline{W}$.

On the grounds of the developed model of the TCP protocol the dependencies of the loss size, the average queue length in the receiver, the average duration of the stay of a package in the system and the average duration of the package's waiting in the queues on the intensity of the flow of packages generated by the sender λ, the intensity of packages' service in the receiver μ_4, the size of the equipment buffers ${}^{\#}Q_{1-4}$ will be analyzed. The simulation results are presented in Fig. 2-10.

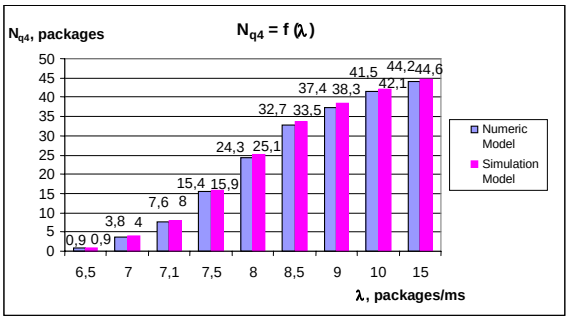

Fig. 2. The Dependency of the Average Length of the Package's Queue on the Intensity of the Flow of Packages Generated by the Sender

Analyzing the dependency $\overline{N_{q\ 4}}$= f(λ) such system parameters were chosen: μ_1 = 10 packages/ms, μ_2 = 10 packages /ms,

μ_3 = 10 packages /ms, μ_4 = 7 packages /ms, Q_4 = 50 packages.

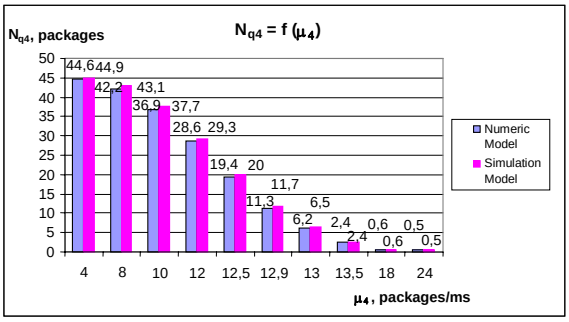

Fig. 3. The Dependency of the Average Length of the Package's Queue on the Intensity of the Packages Serviced in the Receiver

Analyzing the dependency $\overline{N_{q\ 4}}$= f(μ_4) such system parameters were chosen: λ = 13 packages/ms, Q_4 = 50 packages, μ_1 = 12 packages/ms, μ_2 = 12 packages/ms, μ_3 = 12 packages/ms.

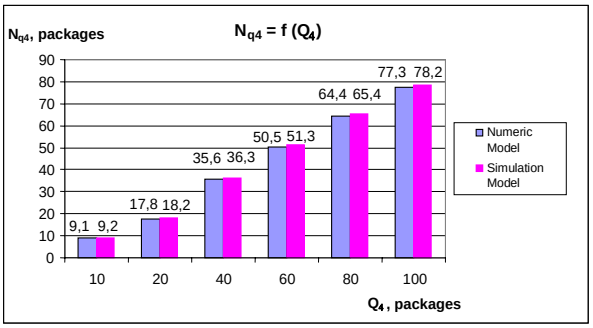

Fig. 4. The Dependency of the Average Length of the Package's Queue on the Size of the Receiver's Buffer

Analyzing the dependency $\overline{N_q}_4$= f(Q_4) such system parameters were chosen: λ = 7 packages/ms, μ_1 = 10 packages/ms, μ_2 = 10 packages/ms, μ_3 = 10 packages/ms, μ_4 = 2,5 packages/ms.

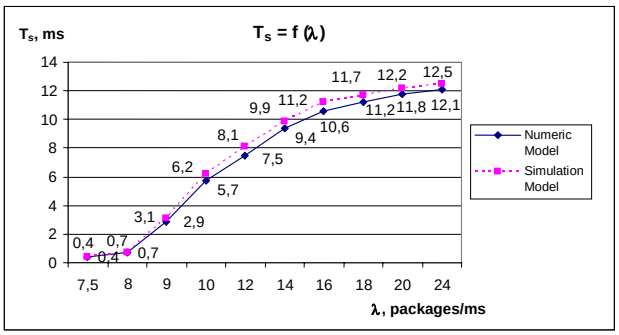

Fig. 5. The Dependency of the Average Time of the Package's Stay in the System on the Intensity of the Flow of Packages Generated by the Sender

Analyzing the dependency $\overline{T_s}$ = f(λ) such system parameters were chosen: $Q_{1\text{-}4}$ = 50 packages, μ_1 = 16 packages/ms, μ_2 = 13 packages/ms, μ_3 = 12 packages/ms, μ_4 = 8 packages/ms.

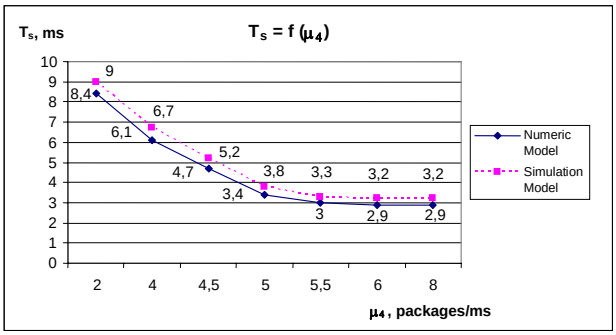

Fig. 6. The Dependency of the Average Time of the Package's Stay in the System on the Intensity of the Flow of Packages Serviced by the Receiver

Analyzing the dependency $\overline{T_s}$ = f(μ_4) such system parameters were chosen: λ = 5 packages/ms, $Q_{1\text{-}4}$ = 50 packages, μ_1 = 4 packages/ms, μ_2 = 4,5 packages/ms, μ_3 = 4 packages/ms.

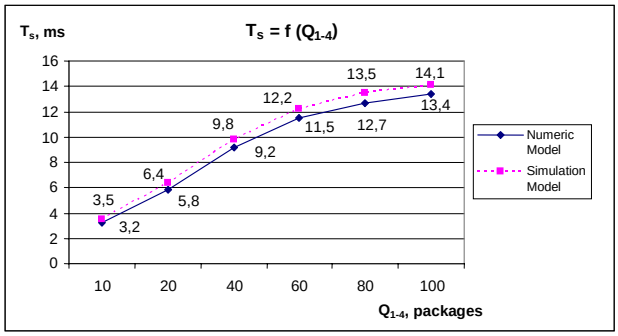

Fig. 7. The Dependency of the Average Time of the Package's Stay in the System on the Size of the Equipment Buffers

Analyzing the dependency $\overline{T_s}$ = f($Q_{1\text{-}4}$) such system parameters were chosen: λ = 16 packages/ms, μ_1 = 16 packages/ms, μ_2 = 13 packages/ms, μ_3 = 12 packages/ms, μ_4 = 8 packages/ms.

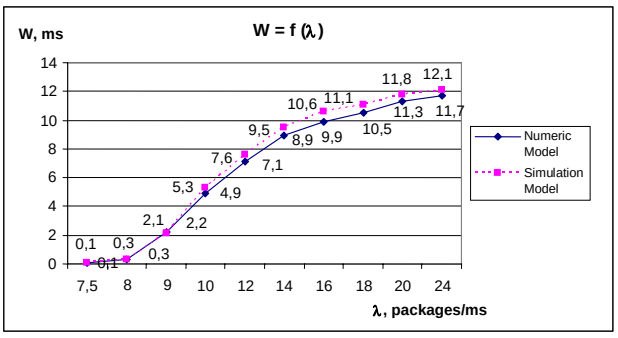

Fig. 8. The Dependency of the Average Time of the Package's Waiting in Queues on the Intensity of the Packages Generated by the Sender

Analyzing the dependency $\overline{W}$ = f(λ) such system parameters were chosen: $Q_{1\text{-}4}$ = 50 packages, μ_1 = 16 packages/ms, μ_2 = 13 packages/ms, μ_3 = 12 packages/ms, μ_4 = 8 packages/ms.

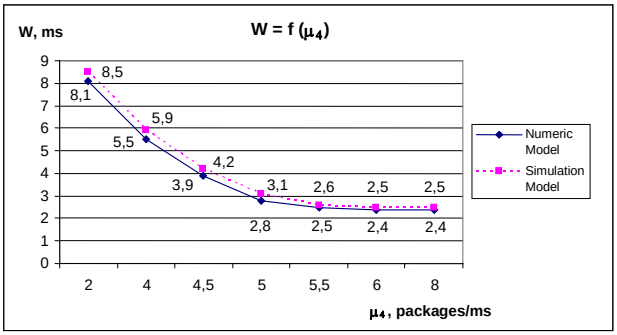

Fig. 9. The Dependency of the Average Time of the Package's Waiting in Queues on the Intensity of the Flow of Packages Serviced by the Receiver

Analyzing the dependency $\overline{W}$ = f(μ_4) such system parameters were chosen: λ = 5 packages/ms, $Q_{1\text{-}4}$ = 50 packages, μ_1 = 4 packages/ms, μ_2 = 4,5 packages/ms, μ_3 = 4 packages/ms.

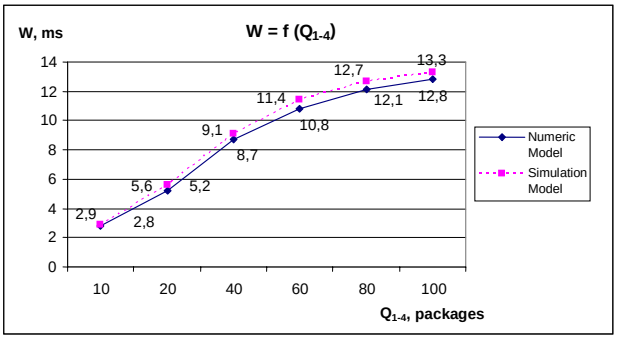

Fig. 10. The Dependency of the Average Time of the Package's Waiting in Queues on the Size of the Equipment Buffers

Analyzing the dependency $\overline{W} = f(Q_{1\text{-}4})$ such system parameters were chosen: λ = 16 packages/ms, μ_1 = 16 packages/ms, μ_2 = 13 packages/ms, μ_3 = 12 packages/ms, μ_4 = 8 packages/ms.

Conclusions

The research on the dependencies of the average duration of the package's waiting in queues on the intensity of the flow of packages generated by the sender, the intensity of the flow of the serviced package in the receiver, the size of the equipment buffers enables to design such data transfer system that the package would spend less time in queues.

References

1. **Transmission Control Protocol** (REC 793). – http://fags.org/rfcs/rfc793.html
2. **Cisco**–TCP/IP. http://www.cisco.com/warp/public/535/4.html
3. **Cardwell N., Savame S., Anderson T.** Modeling TCP Latency. – Department of Computer Science and Engineering University of Washington, USA, 2000. – http://downloads.securityforcus.com/library/infocom2000tcp.pdf
4. **Sawashima H., Hori Y., Sunahara H.** Characteristics of UDP Packet Loss: Effect of TCP Traffic – http://www/isoc.org/whatis/conferences/inet/97/proceedings/F3/F3_1.html
5. **Fortan S., Sericola B.** A Model of TCP in Wide Area Networks. – 2003. – http://downloads.securityforcus.com/library/infocom2000tcp.pdf
6. **Olsen, J.** Stochasting modeling and simulation of the TCP protocol. – Uppsala: Upsala University, 2003. – http://publications.uu.se/uu/fulltext/nbn_se_uu_diva-3534.pdf
7. **Pranevičius H., Tumelis G., Germanavicius V.** Automatic creation of numerical models of systems specified by PLA method // 12th International Conference: Proceedings of ASMTA. – Riga, Latvia, 2005.– P. 118–124.
8. **Pranevičius H., Pranevičienė I., Valakevičius E.** Aptarnavimo sistemų analiziniai ir skaitmeniniai modeliai. – Kaunas: Technologija, 1995. – P. 92–116.
9. **Pranevičius H.** Kompiuterinių tinklų protokolų formalusis specifikavimas ir analizė: agregatinis metodas. – Kaunas: Technologija, 2004. – P.17–75.

Submitted 2006 03 10

H. Pranevičius, T. Kirvėlaitis, I. Pranevičienė. Numerical Model Transmission Control Protocol // Electronics and Electrical Engineering. – Kaunas: Technologija, 2006. – No. 6(70). – P. 49–54.

A numerical model of data transfer control protocol is presented. An automated system for creation of numerical models described by Markov processes is used to create the model. The following stages are marked out while creating the system model: description of system functioning by piece-linear aggregates, composition and solution of a system of linear equations for calculation of stationary probabilities, calculation of system characteristics. An aggregate specification of TCP protocol is presented. It is used to create the protocol model. Simulation results that permit evaluation of permissible load of the protocol are presented. Ill. 10, bibl. 9 (in English; summaries in English, Russian and Lithuanian).

Г. Пранявичюс, Т. Кирвелайтис, И. Пранявичене. Численная модель протокола управления передачей данных // Электроника и электротехника. – Каунас: Технология, 2006. – № 6(70). – С. 49–54.

Приводится численная модель протокола управления передачей данных. Модель создана используя систему автоматизированного построения моделей для систем, описываемых Марковскими процессами. При построении модели выделены этапы: описание системы кусочно-линейными агрегатами, составление системы линейных управлений для расчета стационарных вероятностей, расчет характеристик системы. Представляется агрегативное описание TCP протокола, которое использовано при создании модели протокола. Представляемые результаты моделирования позволяют оценить допустимую нагрузку протокола. Ил. 10, библ. 9 (на английском языке; рефераты на английском, русском и литовском яз.).

H. Pranevicius, T. Kirvėlaitis, I. Praneviciene. Duomenų perdavimo valdymo skaitmeninis modelis // Elektronika ir elektrotechnika. – Kaunas: Technologija, 2006. – Nr. 6(70). – P. 49–54.

Pateikiamas duomenų perdavimo valdymo protokolo skaitmeninis modelis. Modeliui sudaryti panaudota sistemų, aprašomų Markovo procesais, skaitmeninių modelių automatizuotojo sudarymo sistema. Automatizuotai sudarant sistemos modelį yra išskirti tokie etapai: sistemos funkcionavimo aprašymas atkarpomis tiesiniais agregatais, tiesinių lygčių sistemos stacionarioms tikimybėms skaičiuoti sudarymas ir sprendimas, sistemos charakteristikų skaičiavimas. Pateikiama TCP protokolo agregatinė specifikacija, kuri panaudota sudarant protokolo modelį. Pateikiami modeliavimo rezultatai, leidžiantys įvertinti leistiną protokolo apkrovą. Il. 10, bibl. 9 (anglų kalba; santraukos anglų, rusų ir lietuvių k.).

DOI: 10.5755/j02.eie.10691