Annales UMCS Informatica AI 3 (2005) 27-34

Annales UMCS
Informatica
Lublin-Polonia
Sectio AI

http://www.annales.umcs.lublin.pl/

Using Markov chains for modelling networks

Beata Bylina*, Jarosław Bylina

Department of Computer Science, Institute of Mathematics, Marie Curie-Skłodowska University, pl. M.Curie-Skłodowskiej 1, 20-031, Lublin, Poland

Abstract

The paper contains the review and the discussion on modelling communication networks with the use of queuing models and Markov chains. It shows how to take into account various characteristics of real systems – like some control mechanisms and the traffic self-similarity. There are presented two mechanisms modelled with Markov chains: the RED algorithm in TCP/IP and a self-similar traffic shaping.

1. Introduction

Communication networks are getting more and more complicated nowadays. They allow storing, processing and transferring an enormous amount of data. Internet (as a global information medium) and its growth is an example of the network capabilities. Communication network activities depend on many factors, related to each other. They are being improved to fulfil more and more elevated and various users' needs.

To anticipate a network activity, one needs suitable tools that can evaluate the network and its work. Communication networks' modelling is such a tool. It allows estimating behaviour of the network at the project stage, localizing congestion points and bottlenecks, finding packet loss probabilities and peak loads of network elements. With modelling, one can compare topologies, architectures and configurations and optimise the use of the resources.

Queuing models of the communication networks are systems of service stations (with queues) and customers travelling among them. The service stations represent lines, nodes, switches, routers and other network elements; the customers represent packets; the queues represent queues of packets waiting in network devices for being sent.

*Corresponding authors: *e-mail addresses*: beatas@golem.umcs.lublin.pl, jmbylina@hektor.umcs.lublin.pl

Queuing models render real properties of the networks well and enable relatively easy efficiency analysis. These models can be solved by various mathematical manners such as mean value analysis [1], Markovian models [2], diffusion approximation [3], network calculus [4], fluid-flow approximation [5,6]. In this paper Markovian models are presented.

2. Markov chains

A *stochastic process* $X(t)$ is a set of random variables, defined in the same probability space and indexed by t. For every $t = t_i$ there is defined a distribution function:

$$F_X(x;t_i) = P[X(t_i) \le x].$$

Values of the variables form the state space of the process, which can be *continuous* or *discrete* – in the latter case the process is called a *chain*. Statistical dependencies between $X(t_i)$ for different t_i can be described by an n-dimensional distribution function

$$F_{X_1,...,X_n}(x_1,...,x_n;t_1,...,t_n) = P[X(t_1) \le x_1,..., X(t_n) \le x_n].$$

Markov processes are a special class of stochastic processes – they have the *Markovian property* ("they lack the memory" as it is colloquially referred), namely, they fulfil

$$P[X(t) \le x \mid X(t_n) \le x_n,..., X(t_0) \le x_0] = P[X(t) \le x \mid X(t_n) \le x_n]$$

for every $t > t_n > ... > t_0$.

The state space of a Markov chain is usually mapped to a subset of the set of natural numbers. The parameter t may belong to a continuous set or a discrete one. In the former case we have a *continuous time Markov chain* (CTMC) and in the latter case we have a *discrete time Markov chain* (DTMC). The modelled system – and the Markov chain representing it – adopts exactly one state at any time moment t. The evolution of the modelled system is rendered by transitions between states in Markov the chain. The most frequently needed piece of information is the probability of a given state (or states) appearance at a given time moment t – or after a very long ("infinite") time (if the initial state has less and less influence we search *stationary probabilities*).

We are interested in such stationary probabilities of a homogeneous, irreducible CTMC. Such chains appear in Markovian modelling of communication networks. Such a chain may be described with one matrix **Q**, called an *infinitesimal generator matrix*. It can be defined as following:

$$\mathbf{Q} = (q_{ij})_{1 \le i,j \le m},$$

$$q_{ij} = \lim_{\Delta t \to 0} \frac{p_{ij}(\Delta t)}{\Delta t} \qquad \text{for } i \ne j,$$

$$q_{ii} = -\sum_{j \neq i} q_{ij} ,$$

where $p_{ij}(\Delta t) = P[X(t+\Delta t) = j \mid X(t) = i]$.

The stationary probabilities mentioned above can be obtained from a linear system:

$$\boldsymbol{\pi}\mathbf{Q} = \mathbf{0} , \tag{1}$$

where $\boldsymbol{\pi} = (\pi_1, ..., \pi_m)$, $\boldsymbol{\pi} \geq \mathbf{0}$, $\boldsymbol{\pi}\mathbf{e} = 1$ is a vector of desired probabilities. There are many ways to obtain the solution from (1), some of them were considered in [7,8].

3. Modelling the RED mechanism

The *Random Early Detection* (RED) mechanism [9] is an algorithm helping to avoid connection congestion. It is an *active queue management* (AQM) algorithm being introduced in IP routers in the network layer. This algorithm does wait with dropping packets for the moment when the queue is full but starts dropping them randomly somewhat earlier (hence its name), when the queue length starts to grow too fast. This algorithm is supposed to enhance the quality of network services.

In this algorithm, when a packet arrives, a weighted average queue length is calculated as following:

$$avg = (1 - x) \cdot avg + w \cdot n ,$$

where w is fixed (and small) and n is the current size of the queue. Moreover, there are two fixed thresholds: *min* and *max* (*min*<*max*). If the average queue length just computed is greater than *max*, the packet just received is dropped; if it is less than *min*, the packet is saved. However, between the thresholds the packet can be dropped or not (see figure 1; λ and μ are arrival and departure intensities, respectively). The exact formula for the drop probability is given below:

$$p_d(avg) = \begin{cases} 0, & \text{for } avg < min, \\ \dfrac{avg - min}{max - min} \cdot p_{max}, & \text{for } avg \in \langle min, max \rangle, \\ 1, & \text{for } avg > \max . \end{cases}$$

In other words: the longer the queue is, the more likely the packet is dropped (see figure 2).

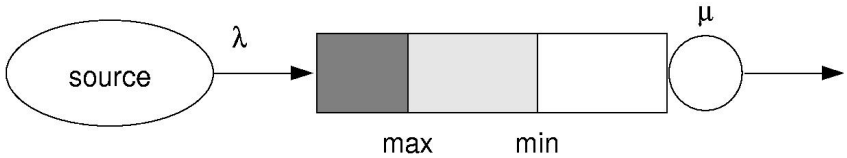

Fig. 1. The RED mechanism

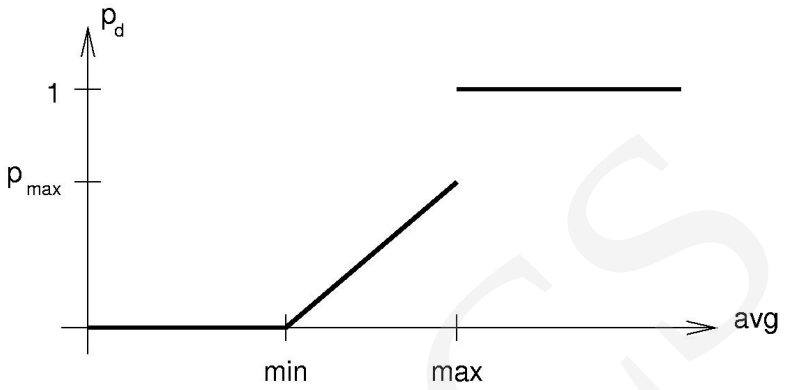

Fig. 2. The drop probability $p_d(avg)$ in the RED mechanism

In [10] a model was proposed which can be used to analyse the activity of a RED-enabled switch. The behaviour of the switch for various drop probabilities was analysed. A CTMC was used for modelling the switch behaviour, with states defined by vector $V=(n_1,n_2,n_3,n_4,n_5,n_6)$, where n_1 is the current size of the queue, n_2 is equal to $\lfloor avg \rfloor$, and n_i (for i=3,4,5,6) make an approximation of $avg-\lfloor avg \rfloor$ as following:

$$n_i=\begin{cases}1, & \text{for } avg-\lfloor avg \rfloor \in \left\langle \frac{i-3}{4}, \frac{i-2}{4} \right), \\ 0, & \text{else.}\end{cases}$$

A new packet arrival is represented by a transition of the Markov chain state from $(n_1,n_2,n_3,n_4,n_5,n_6)$ to $(n_1+1,n_2*,n_3*,n_4*,n_5*,n_6*)$ with intensity $\lambda\cdot\left[1-p_d(avg)\right]$, where $n_2*,...,n_6*$ represent *avg* calculated after the new packet arrival. A packet departure is represented by a transition from state $(n_1,n_2,n_3,n_4,n_5,n_6)$ to $(n_1-1,n_2,n_3,n_4,n_5,n_6)$ with intensity μ. This model allows researching the influence of particular parameters of the RED algorithm on the queue size, loss probability and so on.

4. Modelling self-similarity

The traffic in networks is a self-similar process whose Hurst parameter grows with the traffic intensity. The Hurst parameter mentioned above describes a degree of the self-similarity of the stochastic process defined as the amount of information received in a time unit. The process self-similarity is connected to the traffic long-range dependence (its "burstyness"), and this means that such traffic is difficult to model with Markov chains ("memoryless", in a sense). Moreover, when the traffic is self-similar, the queues are getting longer (on

average), and the packet loss probabilities are getting greater – it makes the quality of network services worse.

The self-similarity of a continuous-time process can be defined [11] as following: the stochastic process $X(t)$ is statistically self-similar with the Hurst parameter H, $H \in \left(\frac{1}{2},1\right)$, if for every $g>0$ a process $\frac{X(gt)}{g^H}$ has the same statistical properties (mean, variance, autocorrelation) as $X(t)$, that is:

$$E\left[X(t)\right] = \frac{E\left[X(gt)\right]}{g^H},$$

$$Var\left[X(t)\right] = \frac{Var\left[X(gt)\right]}{g^{2H}},$$

$$R_X(t,s) = \frac{R_X(gt,gs)}{g^{2H}}.$$

The definition for a discrete-time process is slightly different: the stochastic process X_k is statistically self-similar with the Hurst parameter $H = 1 - \frac{\beta}{2}$ if equations $Var\left[X_k^{(m)}\right] = \frac{Var\left[X_k\right]}{m^\beta}$ and $R_{X_k^{(m)}}(k) = R_{X_k}(k)$ hold for an aggregated process $X_k^{(m)} = \frac{X_{mk-m+1} + ... + X_{mk}}{m}$ for every m.

Here is presented (from [12]) a manner for investigating the self-similar traffic influence on the network buffers congestion and a way for adaptive shaping such a traffic.

A source of the self-similar traffic is modelled with the use of compound Markov chains. The traffic is generated as a superposition of d two-state MMPPs (*Markov Modulated Poisson Process*) [13]. Each of these d MMPPs may be in one of two states, but one of these states means no activity – such an MMPP is also called SPP (*Switched Poisson Process*). The ith $(i = 1,...,d)$ source can be described with the following parameters:

- λ_{1i} – the source intensity in state 1,
- λ_{2i} – the source intensity in state 2,
- c_{1i} – the intensity of the transition from state 2 to state 1,
- c_{2i} – the intensity of the transition from state 1 to state 2.

Hence, the probabilities of being in the state s, s=1,2, for the source i are $\pi_{si} = \frac{c_{si}}{c_{1i} + c_{2i}}$, and the average source intensity is $\lambda_i = \pi_{1i}\lambda_{1i} + \pi_{2i}\lambda_{2i}$.

Figure 3 shows a model of the whole transmission route. In this model there are d SPP sources (representing together one source of a self-similar traffic received from the user; the bigger d is, the better self-similarity approximation can be achieved), a traffic shaper (which smoothes the traffic to maintain a high level of the network performance) and a finite queue service station (which can be interpreted as a network edge node).

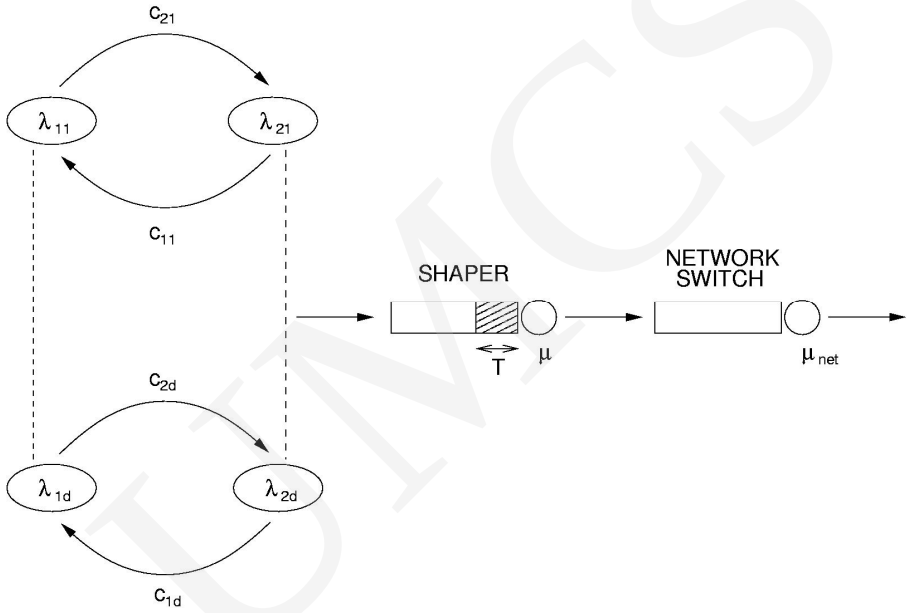

Fig. 3. The investigated model

In [12] a dynamic shaping mechanism is considered. It is based on a jumping window T, which means that the number of arrivals n_{arr} is observed in a fixed time interval T and it is used to control the behaviour of the shaper during the next time interval T. The service rate μ describes how many packets are to be let in in the current interval T and depends on n_{ave} (see below).

The behaviour of the whole system shown in figure 1 is modelled by a CTMC $X(t)$ with d=4. As a state representation a vector $X(t)=(s_1,s_2,s_3,s_4,r,n_{shap},n_{arr},n_{ave},n_{net})$ is chosen, where:

- s_i (i = 1,…,4) is the state of the ith source (s_i = 1 – the source is *on* and generates traffic, s_i = 0 – the source is *off*),
- r (r = 0,…,R) is the current phase of Erlang distribution approximating the constant time T (the bigger R, the better approximation),
- n_{shap} (n_{shap} = 0,…, N_{shap}) is the number of packets in the shaper,

- n_{arr} ($n_{arr} = 0,\ldots, N_{arr}$) is the number of packets received during the current time interval,
- n_{ave} ($n_{ave} = 0,\ldots, N_{arr}$) n_{ave} is an average number of packets received computed at the end of each interval as: $n_{ave} = \lfloor \alpha \cdot n_{ave} + (1-\alpha) \cdot n_{arr} \rfloor$; the service rate μ depends on it and is computed from the equation: $\mu = \frac{n_{ave}}{T}$,
- n_{net} ($n_{net} = 0,\ldots, N_{net}$) n_{net} is the number of packets in the network edge node.

For $n_{shap} < N_{shap}$ every arrival causes growth of n_{shap} and n_{arr} by one; however, for $n_{shap} = N_{shap}$ the received packet is lost and nothing changes. When an intermediate Erlang phase ($r<R$) ends, r grows by one; and when the last Erlang phase ($r=R$) ends, the new n_{ave} is computed, and both r and n_{arr} are set to zero. Every departure of a packet (from the shaper to the network) causes the decrement of n_{shap} by one, and the increment of n_{net}. When the packet is served, n_{net} decreases by one.

The stationary probabilities of $X(t)$ can be defined as following:

$$\pi_{n_1,n_2,n_3,n_4,n_5,n_6,n_7,n_8,n_9} = \lim_{n\to\infty} P\left[X(t) = (n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_8, n_9) \right]$$

and the stationary probabilities vector – as $\boldsymbol{\pi} = \left(\pi_{n_1,n_2,n_3,n_4,n_5,n_6,n_7,n_8,n_9} \right)_{\text{for all possible } n_i}$. It can be computed from (1). Now, the influence of T and α on the performance of the shaper can be investigated; moreover the performance of the network may be analysed for various values of the other parameters as well.

Addition of SPP sources (for better approximation of the self-similarity) causes exponential growth of the number of states (namely, it grows twice for every SPP source added).

5. Conclusions

Some examples of the use of Markov chains for analysis mechanisms and phenomena of contemporary networks are presented above. A model of the RED mechanism [9] and a manner for investigating shapers of the network traffic [12] were quoted. In these areas Markov chains confirm their usefulness. Their advantages are: possibilities of unrestricted configuration of investigated networks and arbitrary approximation of the traffic nature. The flaws of the method can also be seen: the great number of the states and the big cost of computation connected to that number.

References

[1] Reiser M., Kobayashi H., *Queueing Network Models with Multiple Closed Chains: Theory and Computational Algorithms*, IBM J. Res. Develop., 19 (1975).

[2] Bolch G., Greiner S., de Meer H., TrivediK. S., *Queueing Networks and Markov Chains, Modeling and Performance Evaluation with Computer Science Applications*, John Wiley, New York, (1998).
[3] Fourneau J.-M., Pekergin N., *Brief Introduction to an Algorithmic Approach for Strong Stochastic Bounds*, Proceedings from EuroNGI Workshop: New Trends in Modeling, Quantitive Methods and Measurements, Jacek Skalmierski Computer Studio, Gliwice, (2004) 57.
[4] Le Boudec J.-Y., Thiran P., *Network Calculus. A Theory of Deterministic Queuing Systems for the Internet*, LNCS 2050, Springer, (2001).
[5] Kleinrock L., *Queueing Systems Volume II: Computer Applications*, John Wiley & Sons, New York, (1976).
[6] Sharma S., Tipper D., *Approximate Models for the Study of Nonstationary Queues and Their Applications to Communication Networks*, Proceedings of IEEE ICC'93, Geneva, (1993) 352.
[7] Bylina J., *Distributed solving of Markov chains for computer network models*, Annales UMCS Informatica, 1 (2003) 15.
[8] Bylina B., *The inverse iteration with the WZ factorization used to the Markovian models*, Annales UMCS Informatica, 2 (2004) 15.
[9] Floyd S., Jacobson V., *Random Early Detection Gateways for Congestion avoidance*, IEEE/ACM Transaction on Networking, 1(4) (1997) 397.
[10] Laalaoua R., Czachórski T., Atmaca T., *Diffusion approximation and Markovian models of RED control mechanism*, Proc. of the 8th Polish Teletraffic Symposium, Zakopane, (2001) 289.
[11] Stllings W., *High-Speed Networks: TCP/IP and ATM Design Principles*, Prentice-Hall, Inc, (1998).
[12] Elbiaze H., Atmaca T., Czachórski T., *Blind shaping strategy of self-similar traffic: Markovian model*, Archiwum informatyki teoretycznej i stosowanej, 14(1) (2002) 61.
[13] Andersen A. T., Nielsen G. F., A *Markovian Approach for Modeling Packet Traffic with Long-Range Dependence*, IEEE Journal on Selected Areas in Communications, 16(5) (1998) 719.