

Computer simulation of the stochastic RED algorithm

Anna Maria Yu. Apreutesey¹, Anna V. Korolkova¹ and Dmitry S. Kulyabov^{1,2}

¹Peoples' Friendship University of Russia (RUDN University), 6 Miklukho-Maklaya St, Moscow, 117198, Russian Federation

²Laboratory of Information Technologies, Joint Institute for Nuclear Research, 6 Joliot-Curie St., Dubna, Moscow region, 141980, Russian Federation

Abstract

The purpose of this work is to study the capabilities of the Julia language for numerical modeling of stochastic systems. As a stochastic system we consider the model of interaction between the process of data transmission via the Transmission Control Protocol (TCP) and the process of regulating the flow state using the Random Early Detection (RED) algorithm. The mathematical model of this interaction is a system of stochastic differential equations, where there are both continuous and discrete elements of the model. When simulating such systems, it is important to consider the properties of continuous parameters, such as queue length of a router and the TCP window size, as well as discrete transitions between TCP states and the probabilistic packet drop function. Such hybrid systems can be quite easily implemented in specialized dynamic systems modeling languages, for example in Modelica. However, this software package does not have built-in universal tools for modeling stochastic systems, where it is important to consider the random nature of the behavior. The aim of this work is to find optimal tools for modeling such stochastic systems using the Julia language which is used for scientific calculations. For modeling the RED algorithm, the DifferentialEquations.jl library is used. This tool of the Julia language allows solving various kinds of differential equations, including stochastic differential equations and delay differential equations. As a result of the simulation graphs were obtained that demonstrate the dynamics of changes of the TCP window size and the queue length, depending on the initial model parameters and queue threshold values, the correct selection of which ensures the stable operation of the system.

Keywords

Stochastic differential equations, Active Queue Management, mathematical modeling, Julia, Random Early Detection

1. Introduction

Active Queue Management techniques have been proposed to both alleviate some congestion control problems for IP networks as well as provide the quality of service. RED controlled systems [1, 2] are widely used in modern data networks, which allow to study the traffic transmission system with AQM policy as the RED type algorithm [3]. The modeling and


Workshop on information technology and scientific computing in the framework of the XI International Conference Information and Telecommunication Technologies and Mathematical Modeling of High-Tech Systems (ITTMM-2021), Moscow, Russian, April 19–23, 2021

✉ 1032193049@pfur.ru (A. M. Yu. Apreutesey); korolkova-av@rudn.ru (A. V. Korolkova); kulyabov-ds@rudn.ru (D. S. Kulyabov)

🌐 <https://yamadharma.github.io/> (D. S. Kulyabov)

🆔 0000-0001-7141-7610 (A. V. Korolkova); 0000-0002-0877-7063 (D. S. Kulyabov)

© 2021 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

analysis of such algorithms are important for understanding systems dynamics which depends on the initial settings of the routers. The studies of various models of the traffic transmission process are urgent tasks, since they allow to evaluate the network weaknesses and traffic losses. The mathematical model for the interaction of an incoming TCP stream and a router that processes traffic using a RED control algorithm can be represented as a system of stochastic differential equations. Such hybrid system [4] combines the work of both continuous and discrete elements of the system, such as transitions between TCP states and the probabilistic function of dropping packets. The purpose of this work is to search in the Julia scientific computing language for universal tools for the numerical modeling of stochastic systems, where it is necessary to take into account the random nature of the behavior of the main parameters.

2. Stochastic model of the RED algorithm

Consider the process of transmitting TCP-like traffic controlled by the RED algorithm. A continuous state vector is $(W, Q, \hat{Q})^T$, where $W := W(t)$ – average TCP window size (in packets), $Q := Q(t)$ – average queue length (in packets), $\hat{Q} := \hat{Q}(t)$ – exponentially weighted moving average of the queue length.

The stochastic model is a system of three Ito stochastic differential equations [5]:

$$\begin{cases} dW(t) = \left(\frac{1}{T(t)} - \frac{W^2(t)P(\hat{Q})}{2T(t)} \right) dt + \sqrt{\frac{1}{T(t)} + \frac{W^2(t)P(\hat{Q})}{2T(t)}} dV^1, \\ dQ(t) = \left(\frac{W(t)}{T(t)}N(t) - C(t) \right) dt + \sqrt{\frac{W(t)}{T(t)}N(t) - C(t)} dV^2, \\ d\hat{Q}(t) = w_q C(t)(Q(t) - \hat{Q}(t)). \end{cases} \quad (1)$$

- $T(t)$ is the Round Trip Time (RTT). It is the time taken for a packet to be sent from a source to a destination and for the corresponding acknowledgement to be received by the source, assuming no packet loss; (t) – speed of processing packets in the queue;
- $N(t)$ is a number of TCP sessions;
- dV^1 is the Wiener process corresponding to a random $W(t)$ process;
- dV^2 is the Wiener process corresponding to a random $Q(t)$ process.

As the the average queue length increases, the packets drop probability also increases. The classical example of an AQM policy is RED for which $P(\hat{Q})$ takes the next form:

$$P(\hat{Q}) = \begin{cases} 0, & 0 \leq \hat{Q}(t) < Q_{\min}, \\ \frac{\hat{Q}(t) - Q_{\min}}{Q_{\max} - Q_{\min}} p_{\max}, & Q_{\min} \leq \hat{Q}(t) \leq Q_{\max}, \\ 1, & \hat{Q}(t) > Q_{\max}. \end{cases} \quad (2)$$

Here $P(\hat{Q})$ is the package dropping function, $\hat{Q}(t)$ is the queue length weighted average, Q_{\min} and Q_{\max} are thresholds of queue length weighted average, p_{\max} is the maximum level of packages reset.

3. Computer simulation of the RED module

Julia is a high-level language for scientific and engineering calculations [4, 6, 7, 8]. To model the described system, the DifferentialEquations.jl [9] library was used, which makes it possible to solve various types of differential equations, including stochastic differential equations of the next form

$$dx(t) = f(t, x(t)) dt + g(t, x(t)) dW, \quad (3)$$

where $x(t) \in \mathbb{R}$ is some random process, $W := W(t) \in \mathbb{R}$ is the Wiener process.

To install the package we will use the following command in Julia REPL:

```
using Pkg
Pkg.add("DifferentialEquations")
```

We will enable the package using the command:

```
using DifferentialEquations
```

We set the vector of the initial system parameters $p = (T, N, C, wq, q_{\min}, q_{\max}, R, p_{\max}, w_{\max})$.

```
T = 0.5
N = 60.0
q_min = 0.25
q_max = 0.50
R = 300.0
p_max = 0.1
w_max = 32.0
p = (T, N, C, wq, q_min, q_max, R, p_max, w_max)
```

After declaring the variables and the vector of the system parameters, in the RED function we define the deterministic part of the (1) system of equations:

```
function RED(du, u, param, t)
    if w < w_max
        du[1] = 1.0 / T(q) - ( w / 2 ) * w * p(q_avg) / T(q)
    else
        du[1] = - ( w / 2 ) * w * p(q_avg) / T(q)
    end
    du[2] = N * w / T(q) - C(q)
    du[3] = wq * C(q) * (q - q_avg)
end
```

The stochastic part of the (1) system of equations is specified in the REDst function:

```

function REDst(du,u,param,t)
    w, q, q_avg = u
    du[1] = sqrt(1.0 / T(q) + ((w / 2) * w * p(q_avg) / T(q)))
    if ((N * w / T(q) - C(q)) < 0)
        du[2] = 0.0
    else
        du[2] = sqrt(N * w / T(q) - C(q))
    end
    du[3] = 0.0
end

```

Let us define the probabilistic function of dropping packets according to the (2) equation:

```

function p(q_avg)
    global q_min, q_max
    if (q_avg < q_min * R)
        p = 0.0
    elseif (q_avg > q_max * R)
        p = 1.0
    else
        p = p_max * (q_avg / R - q_min) / (q_max - q_min)
    end
end

```

The DifferentialEquations.jl package allows you to use callbacks to inject custom code into the solver algorithms to model complex functions with conditions and discontinuities. To work with callbacks, two functions are defined: the condition function checks if an event has occurred, the affect function is executed if the event has occurred.

Let us define the condition function and the affecting function, thereby limiting the growth of the TCP window size to the maximum value:

```

function condition_w_max(u,t,integrator)
    global w_max
    u[1] >= w_max
end

function affect_w_max!(integrator)
    global w_max
    for c in full_cache(integrator)
        c.u[1] = w_max
    end
end

```

We also add callbacks to control the growth of the $Q(t)$ variable:

```

function condition_Rq(u,t,integrator)
    global R
    u[2] >= R
end

function affect_Rq!(integrator)
    global R
    for c in full_cache(integrator)
        c.u[2] = R
    end
end

```

The callback option for the $\hat{Q}(t)$ variable is set in the same way:

```

function condition_Rq_avg(u,t,integrator)
    global R
    u[3] >= R
end

function affect_Rq_avg!(integrator)
    global R
    for c in full_cache(integrator)
        c.u[3] = R
    end
end

```

In this case, we used several callback functions of the discrete type `DiscreteCallback`. The condition function implements event detection at each step of solving `dt`, and each affecting function will be executed if the condition function returns `true`:

```

save_positions = (true,true)
cb_w_max = DiscreteCallback(condition_w_max, affect_w_max!,
    save_positions = save_positions)
cb_Rq = DiscreteCallback(condition_Rq, affect_Rq!,
    save_positions = save_positions)
cb_Rq_avg = DiscreteCallback(condition_Rq_avg, affect_Rq_avg!,
    save_positions = save_positions)

```

With the `CallbackSet()` tool, multiple callbacks can be combined into one group:

```

Clbsset = CallbackSet(PositiveDomain(), cb_w_max, cb_Rq, cb_Rq_avg)

```

Let's call the `SDEProblem` solver of the `DifferentialEquations.jl` package, the arguments of which are passed the `RED` function that specifies the deterministic part of the equations and the `REDst` function that specifies the stochastic part of the system. Also, in the arguments to the solver, the vector of the initial states of the system and the simulation time are indicated:

```
prob_sde_RED = SDEProblem(RED, REDst, u0, tspan)
```

Let's call the `solve` method of the `DifferentialEquations.jl` library to solve the above system of equations. The Euler - Maruyama method is used as a numerical method, which has a strong order $(p_d, p_s) = (1.0, 0.5)$. The value p_d denotes the deterministic accuracy order, the value p_s denotes the stochastic part approximation order.

```
sol = solve(prob_sde_RED, EM(), dt=dt, callback = Clbset)
```

Using the parallel computing capabilities of the Julia language, let us simulate a large number of trajectories:

```
ensembleprob = EnsembleProblem(prob_sde_RED)
sim = solve(ensembleprob, EM(), dt=dt, callback = Clbset, trajectories =
→ 200)
```

Use the `EnsembleSummary()` method to combine the modeled set of trajectories.

```
summ = EnsembleSummary(sim, 0:dt:tf)
```

4. Simulation results

As a result of the simulation, graphs of changes in the TCP window size and the average queue size in a router with a queue control module according to the RED algorithm were obtained for various initial values of the model. With some initial parameters, the system quickly finds suitable values of variables and changes within the average values (fig. 1–3). In the case of modeling only the deterministic part of the (1) system of equations, the model enters a stationary mode of operation (fig. 2).

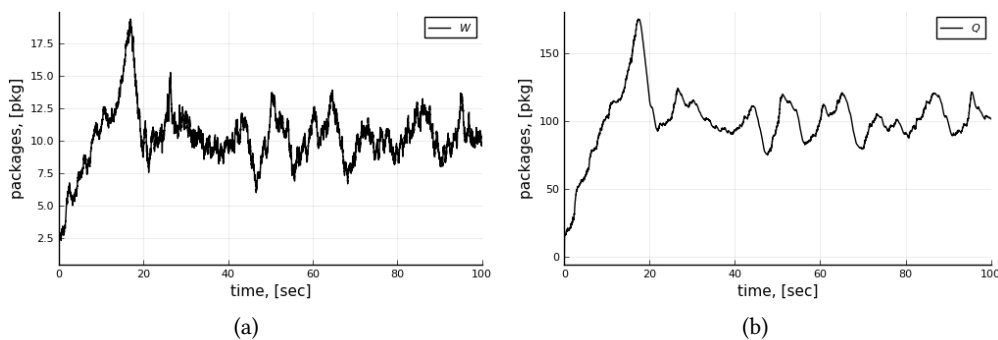


Figure 1: The $W(t)$ (a) and $Q(t)$ (b) functions in the RED algorithm under $Q_{min} = 0.2$ and $Q_{max} = 0.8$ thresholds

Using active queue management algorithms such as RED for traffic control reduces the chances of global synchronization occurring, but does not completely eliminate it [10]. At some values of the initial parameters in the settings of the routers, the system has auto-oscillations of the main parameters (fig. 4–6).

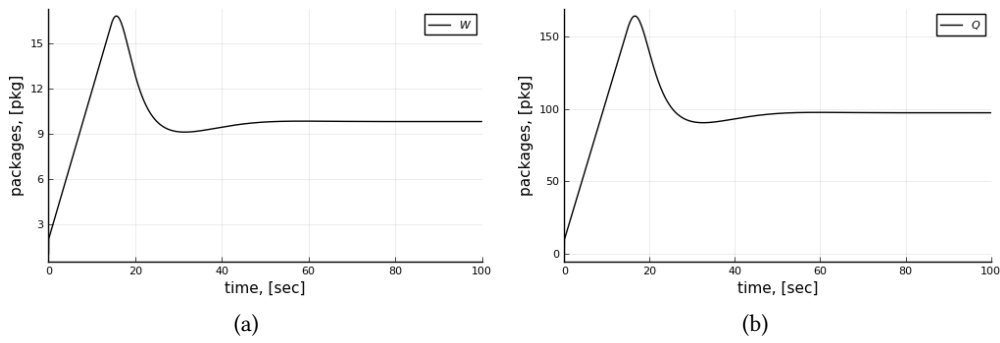


Figure 2: The $W(t)$ (a) and $Q(t)$ (b) functions in the deterministic part of the RED algorithm under $Q_{min} = 0.2, Q_{max} = 0.8$ thresholds

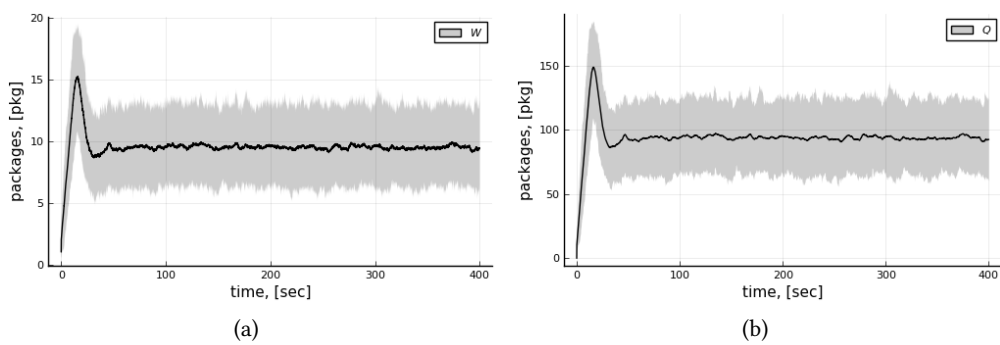


Figure 3: Ensemble average (200 trajectories) of the $W(t)$ TCP window size (a) and $Q(t)$ average queue length (b) under $Q_{min} = 0.2, Q_{max} = 0.8$ thresholds

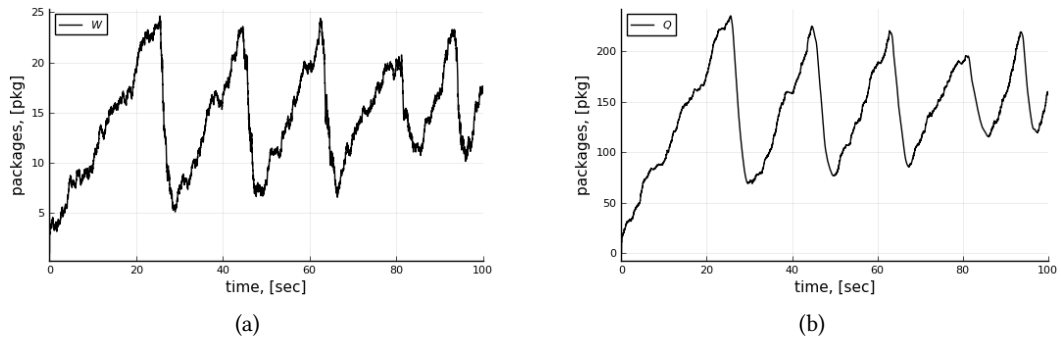


Figure 4: The $W(t)$ (a) and $Q(t)$ (b) functions in the RED algorithm under $Q_{min} = 0.55$ and $Q_{max} = 0.6$

5. Conclusion

The universal and effective tools of the DifferentialEquations.jl library of the Julia scientific computing language were demonstrated for the numerical simulation of a nonlinear system with control, the mathematical model of which is a system of stochastic differential equations.

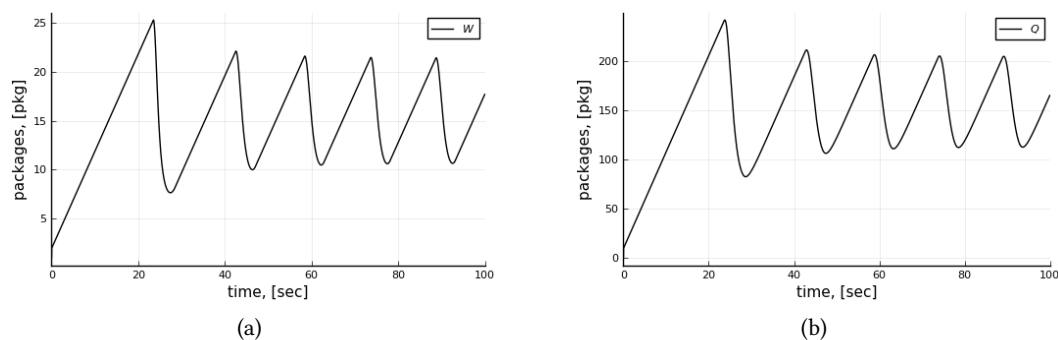


Figure 5: The $W(t)$ (a) and $Q(t)$ functions (b) in the deterministic part of the RED algorithm under $Q_{min} = 0.55$, $Q_{max} = 0.6$ thresholds

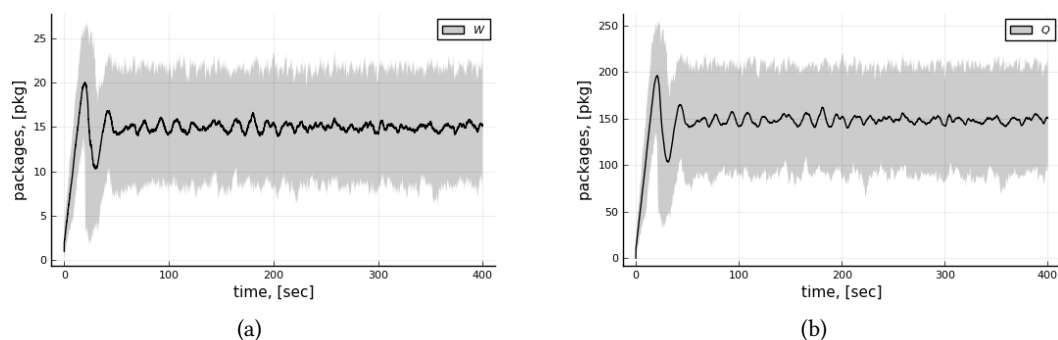


Figure 6: Ensemble average (200 trajectories) of the $W(t)$ TCP window size (a) and $Q(t)$ average queue length (b) under $Q_{min} = 0.55$, $Q_{max} = 0.6$ thresholds

The interaction of the process of data transmission via the TCP protocol and the process of the flow state regulation by the RED algorithm in the event of overloads was considered. The tools of the Julia language used in this work are applicable to modeling such stochastic systems with control, containing elements of both continuous and discrete nature of functioning. As a result of modeling, graphs were obtained that demonstrate changes in the main parameters of the system depending on the threshold values of the queue.

Acknowledgments

This paper has been supported by the RUDN University Strategic Academic Leadership Program and by Russian Foundation for Basic Research (RFBR) according to the research project No 19-01-00645.

References

- [1] S. Floyd, V. Jacobson, Random Early Detection Gateways for Congestion Avoidance, *IEEE/ACM Transactions on Networking* 1 (1993) 397–413. doi:10.1109/90.251892.
- [2] V. Misra, W.-B. Gong, D. Towsley, Fluid-Based Analysis of a Network of AQM Routers

- Supporting TCP Flows with an Application to RED, *ACM SIGCOMM Computer Communication Review* 30 (2000) 151–160. doi:10.1145/347057.347421.
- [3] A. V. Korolkova, T. R. Velieva, P. A. Abaev, L. A. Sevastianov, D. S. Kulyabov, Hybrid Simulation Of Active Traffic Management, *Proceedings 30th European Conference on Modelling and Simulation* (2016) 685–691. doi:10.7148/2016-0685.
- [4] D. Färnqvist, K. Strandemar, K. H. Johansson, J. P. Hespanha, Hybrid Modeling of Communication Networks Using Modelica, in: *The 2nd International Modelica Conference, 2002*, pp. 209–213.
- [5] A. V. Korolkova, D. S. Kulyabov, T. R. Velieva, I. S. Zaryadov, Essay on the study of the self-oscillating regime in the control system, in: M. Iacono, F. Palmieri, M. Gribaudo, M. Ficco (Eds.), *33 European Conference on Modelling and Simulation, ECMS 2019*, volume 33 of *Communications of the ECMS*, European Council for Modelling and Simulation, Caserta, 2019, pp. 473–480. doi:10.7148/2019-0473.
- [6] J. Bezanson, S. Karpinski, V. B. Shah, A. Edelman, *Julia: A Fast Dynamic Language for Technical Computing* (2012) 1–27. arXiv:1209.5145.
- [7] J. Bezanson, A. Edelman, S. Karpinski, V. B. Shah, *Julia: A fresh approach to numerical computing*, *SIAM Review* 59 (2017) 65–98. doi:10.1137/141000671. arXiv:1411.1607.
- [8] A. Joshi, R. Lakhanpal, *Learning Julia*, Packt Publishing, 2017.
- [9] C. Rackauckas, Q. Nie, *DifferentialEquations.jl – A Performant and Feature-Rich Ecosystem for Solving Differential Equations in Julia*, *Journal of Open Research Software* 5 (2017). doi:10.5334/jors.151.
- [10] D. S. Kulyabov, A. V. Korolkova, T. R. Velieva, E. G. Eferina, L. A. Sevastianov, The Methodology of Studying of Active Traffic Management Module Self-oscillation Regime, in: W. Zamojski, J. Mazurkiewicz, J. Sugier, T. Walkowiak, J. Kacprzyk (Eds.), *DepCoS-RELCOMEX 2017. Advances in Intelligent Systems and Computing*, volume 582 of *Advances in Intelligent Systems and Computing*, Springer International Publishing, Cham, 2018, pp. 215–224. doi:10.1007/978-3-319-59415-6_21.