Hybrid modelling of the RED algorithm in the Julia language

A M Yu Apreutesey¹, A V Korolkova¹, D S Kulyabov¹, ²
¹Peoples’ Friendship University of Russia (RUDN University), 6 Miklukho-Maklaya St., Moscow, 117198, Russian Federation
²Joint Institute for Nuclear Research, 6 Joliot-Curie, Dubna, Moscow region, 141980, Russian Federation
E-mail: 1032193049@pfur.ru, korolkova-av@rudn.ru, kulyabov-ds@rudn.ru

Abstract. This work is devoted to studying the possibilities of hybrid modeling of communication network algorithms in the Julia programming language. A simulation of the system, which consists of an incoming stream processed according to the Transmission Control Protocol (TCP) and a router processing traffic according to the Random Early Detection (RED) algorithm, is carried out. This model contains both continuous and discrete elements, for example, a discrete probability function of random packet dropping in the case of queue congestion.

1. Introduction
The hybrid approach can be used to simulate systems containing elements with both continuous and discrete functioning [1, 2]. Let us consider a model of the interaction between the process of data transmission via the Transmission Control Protocol (TCP) and the process of regulating the flow state when congestion occurs using the Random Early Detection (RED) algorithm [3]. The system contains continuous control objects and a discrete control device. In addition, it is also necessary to consider the transitions between TCP states.

In this paper, it is planned to describe a TCP traffic control system with the RED active queue management algorithm and demonstrate a hybrid approach to modeling communication network algorithms in the Julia scientific computing language.

2. Description of the RED module model
Active queue management algorithms, in particular the RED, are used to control and prevent congestion in router queues [4]. Traffic state control algorithms can be implemented as a control module in network equipment. The advantage of the RED algorithm is its efficiency and relatively simple implementation on network equipment.

The model for transmitting TCP-like traffic with dynamic flow rate, which is regulated by an RED, consists of two elements. The first element is the TCP source that generates the packets, the second element is the receiver, which is the router’s queue, where the incoming packets are processed in accordance with the RED control algorithm and a notification is sent to the source about the packets delivery. The source and the receiver interact through an intermediate
link in accordance with the control algorithm, i.e. the average queue length affects the source parameters, in particular the TCP congestion window size.

The continuous state vector is \( \left( W, Q, \hat{Q} \right)^T \), where \( W := W(t) \) is the TCP window size, \( Q := Q(t) \) is the router queue length, \( \hat{Q} := \hat{Q}(t) \) is exponentially weighted moving average of the queue length. A mathematical model of the interaction of an incoming TCP stream and a router that processes traffic using a RED-type control algorithm is the autonomous system of three differential equations [5, 6, 7].

\[
\dot{W}(t) = \frac{1}{T(Q,t)} \vartheta(W_{\text{max}} - W(t)) - \frac{W(t)W(t-T(Q,t))}{2T(t-T(Q,t))} p(t-T(Q,t)),
\]

\[ \dot{Q}(t) = \begin{cases} 
(1 - p(t)) \frac{W(t)}{T(Q,t)} N(t) - C, & Q(t) > 0 \\
\max \left[ (1 - p(t)) \frac{W(t)}{T(Q,t)} N(t) - C, 0 \right], & Q(t) = 0 
\end{cases} \]

\[
\dot{\hat{Q}}(t) = -\omega Q \hat{Q} + \omega Q \hat{Q}(t)
\]

Equation (1) describes the dynamic change in the average TCP window size. The \( \vartheta(W_{\text{max}} - W(t)) \) Heaviside function limits the growth of the TCP window. The first term in the (2) equation reflects the increase in the queue length when packets arrive which corresponds to the average packet arrival rate, the second term is set to reduce the number of packets in the queue. The (3) equation reflects the change of the exponentially weighted moving average of the queue length function, which is introduced for some smoothing outliers of the queue length.

The RED algorithm is controlled by the \( p(\hat{Q}(t)) \) — piecewise probabilistic packet drop function which can be written as a system of nonlinear equations.

\[
p(\hat{Q}(t)) = \begin{cases} 
0, & 0 \leq \hat{Q} < Q_{\min}, \\
\frac{Q - Q_{\min}}{Q_{\max} - Q_{\min}} p_{\max}, & Q_{\min} \leq \hat{Q} \leq Q_{\max}, \\
1, & \hat{Q} > Q_{\max}.
\end{cases}
\]

This function depends on the threshold values of the queue size \( Q_{\min} \) and \( Q_{\max} \), as well as the \( p_{\max} \) parameter that sets the part of packets that will be discarded if \( \hat{Q}(t) \) reaches the maximum value.

3. Hybrid implementation of the RED module

Julia is a high-level, high-performance programming language designed for scientific computing [8, 9, 10]. Let’s implement a simulation of the RED algorithm in Julia. The DifferentialEquations library is used to effectively solve differential equations of different types [11]. To install and then connect this library, use the following commands in the Julia REPL:

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

Let us set the vector of system parameters \( p = (T, N, C, wq, q_{\min}, q_{\max}, R, p_{\max}, w_{\max}) \). The variable \( pr \), which is the probability of dropping packets, is set as a global variable.
pr = 0.0 # initial value of the drop function
T = 0.5 # round-trip time, RTT
N = 60.0 # number of TCP sessions
c = 10.0 # service intensity, Mbps
packet_size = 500.0 # package size, bit
C = 125000.0*c / packet_size # service intensity, packets
wq = 0.0004 # EWMS parameter
q_min = 0.25 # minimum queue threshold
q_max = 0.50 # maximum queue threshold
R = 300.0 # queue size
p_max = 0.1 # maximum drop probability
w_max = 32.0 #maximum window size
p = (T, N, C, wq, q_min, q_max, R, p_max, w_max) # vector of system parameters

Let us set the vector of the initial state of the system and the simulation parameters.

u0 = [1.0, 0.0, 0.0]
tf = 30.0
tspan = (0.0, tf)

In the original system of differential equations there are delayed arguments, therefore we define the history function h(p, t), which depends on the parameters vector p and time t. Next, we define the delay of the $W(t - T(t))$ variable in (1) equation.

\[ h(p, t) = \text{zeros}(1) \]
\[ \tau = T \]
\[ \text{lags} = [\tau] \]

Let’s set the RED function, in which continuous parameters $W$, $Q(t)$, $\dot{Q}(t)$ are calculated, and some restrictions are set for these parameters.

function Red(du, u, h, p, t)
w, q, q_avg = u
hist1 = h(p, t - T)[1]
\[ du[1] = \frac{1}{T} - \left( w * \text{hist1} * \text{pr} / (2.0 * T) \right) \]
\[ du[2] = \text{qAdd}(q, w, T, C, N, R) \]
\[ du[3] = -wq * C * q_{\text{avg}} + wq * C * q \]
end

Let’s set the function for changing the current queue size.

function qAdd(q,w,T,C,N,R)
q1 = N * w / T - C
q2 = q + q1
if q2 > R
return R - q
else if q2 > 0
return q1
else return -q
end
end
To implement a discrete reset function into a continuous system, we use one of the tools of the DifferentialEquations package called callbacks. Two functions are defined to work with this tool. The condition function is needed to check if some event has occurred. The affect function is executed if the event occurs. Next, we define the action function, which is the controller. The control_loop! function at each step calculates the new value of the packet drop probability function $p$ depending on the current values of the $W$, $Q(t)$, $\dot{Q}(t)$ parameters in accordance with the formula (4). As soon as the $\dot{Q}(t)$ value exceeds some predetermined threshold value, the router finds out that the system has started overloading and reports this to the source.

```julia
function condition_control_loop(u,t,integrator)
    true
end

function control_loop!(integrator)
    global pr
    w = integrator.u[1]
    q = integrator.u[2]
    q_avg = integrator.u[3]
    if (q_avg < q_min * R)
        pr = 0.0
    elseif (q_avg > q_max * R)
        pr = 1.0
    else
        pr = p_max * (q_avg/R - q_min) / (q_max - q_min)
    end
end
```

Then we define a discrete type callback and call the DDEProblem package solver. The Red function, vector of initial states, simulation time, and variables delay parameters are passed to its arguments.

```julia
cb = DiscreteCallback(condition_control_loop, control_loop!)
prob = DDEProblem(Red, u0, h, tspan, p, constant_lags=lags)
alg = MethodOfSteps(Tsit5())
sol = solve(prob, alg, callback = cb)
```

4. Modeling Results
As a result of the simulation, we obtain a graph demonstrating the behaviour of the TCP Reno window size and the average queue length, i.e. reflecting the dynamics of a queue in a router (or gateway) with a queue management module using the RED algorithm (Fig. 1). As we can see, in the system there are some auto-oscillations of the main parameters under presented initial values.

5. Conclusion
The application of a hybrid approach to modeling nonlinear systems with control has been demonstrated. The simulated system was an active traffic control model using an algorithm like RED. It has been demonstrated that the Julia programming language contains tools for implementing the continuous-discrete paradigm. The DifferentialEquations package has been used, which allows to solve differential equations systems of various types, including differential equations with delay, the continuous probabilistic function has been implemented using the callback option, the delayed continuous argument to a continuous function has been implemented using the history function.
The capabilities of the Julia programming languages in modeling of hybrid systems containing both continuous and discrete aspects of behavior has been studied. The numerical simulation of the transmitting data process via TCP and the process of regulating the flow state using the RED algorithm in the event of congestion has been performed, graphs demonstrating changes in the main parameters of the system have been obtained.

**Acknowledgment**

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

**References**

[1] Korolkova A V, Velieva T R, Abaev P A, Sevastianov L A and Kulyabov D S 2016 *Proceedings 30th European Conference on Modelling and Simulation* 685–691

[2] Färnqvist D, Strandemar K, Johansson K H and Hespanha J P 2002 *The 2nd International Modelica Conference* pp 209–213

[3] Korolkova A V, Kulyabov D S, Velieva T R and Zaryadov I S 2019 *33 European Conference on Modelling and Simulation, ECMS 2019* (*Communications of the ECMS* vol 33) ed Iacono M, Palmieri F, Gribaudo M and Fico M (Caserta: European Council for Modelling and Simulation) pp 473–480

[4] Floyd S and Jacobson V 1993 *IEEE/ACM Transactions on Networking* 1 397–413

[5] Misra V, Gong W B and Towsley D 2000 *ACM SIGCOMM Computer Communication Review* 30 151–160

[6] Misra V, Gong W B and Towsley D 1999 *Proceedings of PERFORMANCE* 99

[7] Korolkova A V, Kulyabov D S and Sevastianov L A 2015 *Mathematical Modelling and Geometry* 3 1–18

[8] Bezanson J, Karpinski S, Shah V B and Edelman A 2012 1–27 (*Preprint* 1209.5145)

[9] Bezanson J, Edelman A, Karpinski S and Shah V B 2017 *SIAM Review* 59 65–98 (*Preprint* 1411.1607)

[10] Joshi A and Lakhanpal R 2017 *Learning Julia* (Packt Publishing)

[11] Rackauckas C and Nie Q 2017 *Journal of Open Research Software* 5