Modelica-based TCP simulation

2017 J. Phys.: Conf. Ser. 788 012036

(http://iopscience.iop.org/1742-6596/788/1/012036)

Modelica-based TCP simulation

T R Velieva[1], E G Eferina[1], A V Korolkova[1], D S Kulyabov[1,2], L A Sevastianov[1,3]

[1]Department of Applied Probability and Informatics,
RUDN University (Peoples' Friendship University of Russia),
6 Miklukho-Maklaya str., Moscow, 117198, Russia
[2]Laboratory of Information Technologies,
Joint Institute for Nuclear Research,
6 Joliot-Curie, Dubna, Moscow region, 141980, Russia
[3]Bogoliubov Laboratory of Theoretical Physics,
Joint Institute for Nuclear Research,
6 Joliot-Curie, Dubna, Moscow region, 141980, Russia

E-mail: trvelieva@gmail.com, eg.eferina@gmail.com, akorolkova@sci.pfu.edu.ru, yamadharma@gmail.com, leonid.sevast@gmail.com

Abstract. For the study and verification of our mathematical model of telecommunication systems a discrete simulation model and a continuous analytical model were developed. However, for various reasons, these implementations are not entirely satisfactory. It is necessary to develop a more adequate simulation model, possibly using a different modeling paradigm. In order to modeling of the TCP source it is proposed to use a hybrid (continuous-discrete) approach. For computer implementation of the model the physical modeling language Modelica is used. The hybrid approach allows us to take into account the transitions between different states in the continuous model of the TCP protocol. The considered approach allowed to obtain a simple simulation model of TCP source. This model has great potential for expansion. It is possible to implement different types of TCP.

1. Introduction

While complex systems modeling there is the problem of choosing a model approach.When using a discrete-continuous dichotomy, we always have some inappropriate elements. These elements of the model do not well correspond to the selected approach. During simulation of TCP protocol the serious problems arised. As it turned out adequate TCP models are simply missing. Not even a common method for its modeling (see [1–3]) exists.

For modeling we used the continuous (fluid) model for TCP (see [4, 5]). However, this approach allowed us to model the TCP protocol only partially. In addition, we received differential inclusions instead of differential equations.

For further development of our model, it was decided to use a hybrid approach (see [6–10]). This article discusses a general approach to a hybrid modeling of TCP. The implementation is demonstrated on the basis of Modelica language (see [11, 12]).

The structure of the article is as follows. The section 2 describes the hybrid paradigm of mathematical modeling. Also the general information about the language of physical modeling named Modelica is given in the same section. Modelica implements continuous and hybrid paradigms. The congestion control mechanism in the TCP Reno protocol are presented in the

section 3. In the section 4 we construct the UML-diagram for TCP discrete transitions. In the section 5 we record the discrete equations for TCP window in a continuous form. All sections are illustrated by the program fragments in the Modelica language.

2. The hybrid approach to modeling

The hybrid (see [6–10]) system has both continuous and discrete aspects of behavior. The hybrid behavior may be due to different reasons.

- Hybrid behavior is due to the joint operation of the continuous and discrete objects. For example, the automatic control system with continuous control object and discrete control device.
- Hybrid behavior is caused by changes in the structure of the system. A system with variable number of components may be considered as an example.
- Hybrid behavior may be caused by instant qualitative changes in a continuous object. In this case, qualitative changes during the simulation of continuous systems are presented as discrete events. Asesult, the hybridism is not an inherent characteristic of the system, but the modeling technique.

Hybrid systems may be considered as discrete-continuous or continuous-discrete systems.

- The waiting time for the next input and the duration of the output action can be taken into account in discrete systems.
- The coexistence of instant and long-term processes in a continuous-time model.

We will add discrete elements to the initially developed continuous dynamic model.

Modelica language (see [11,12]) is developed by a nonprofit organization Modelica, which also develops a free library for this language. Modelica is positioned as an object-oriented physical modeling language. The equations are a special entity in the language. The number of equations and variables in the program must be the same.

Modelica supports continuous and hybrid (continuous-discrete) paradigms. However, the discrete elements are also present in the language.

3. TCP congestion control mechanism

The TCP protocol uses a sliding window mechanism to avoid a congestion. The implementation of this mechanism depends on the particular type of TCP protocol.

Since the original model (see [13–15]) is based on the TCP Reno protocol [16, 17], then this particular protocol will be simulated.

In TCP Reno protocol the congestion control mechanism consists of the following phases: slow start, congestion avoidance, fast recovery, and timeout. Dynamics of changes in congestion window size (CWND) depends on the specific phase (see Fig. 1).

If the total network arrival rate exceeds the maximum network rate, there is a network overload. If the overload condition lasts long enough for the buffer overflows, the data loss occurs in the network. Lost segments are retransmitted by TCP and promote continued growth of congestion. The transport protocol control algorithm should prevent the occurrence of congestion and facilitate the end of the network overload conditions in the case of its occurrence.

TCP uses a greedy algorithm for the use of bandwidth. At first it attempts to use all the available bandwidth. To do this, it increases the window exponentially (multiplicative increase). In the case of packet loss, the TCP first reduces the window size. Then TCP increases the window size linearly. TCP determines the occurrence of an overload condition expecting that the cause of packet loss is the network congestion.

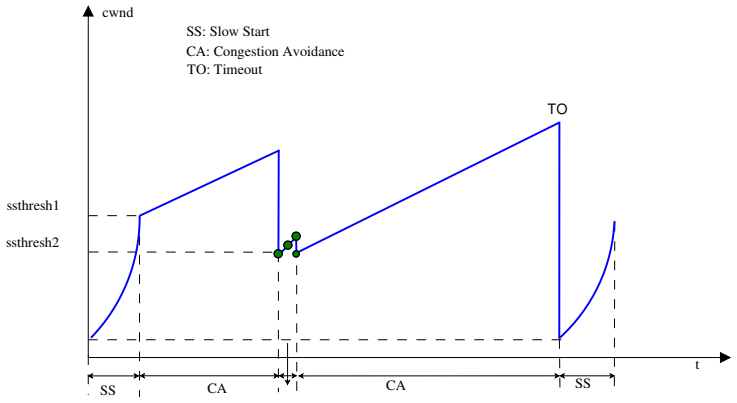

Figure 1. TCP phases

Overall congestion control algorithm belongs to AIMD algorithms type (Additive Increase, Multiplicative Decrease) — an additive increase of the window size and multiplicative decrease of it.

The congestion avoidance algorithm is described in RFC 5681 [17].

Each time the source receives a delivery notification (Acknowledge, ACK), in slow start phase congestion window is increased. The source increases the congestion window size depending on the number of confirmed segments (Segment Size, SS): $cwnd = cwnd + 1$ for each transmitted ACK. Initial congestion window size (Maximum Segment Size, MSS) can take a value of 1, 2 or 10 segments. The receiver sends an ACK for each packet, but in reality it can be assumed that confirmation come together at the end of the double turnaround time (Round-Trip Time, RTT). Thus, the congestion window is doubled after the round-trip time.

When TCP Reno window size takes the certain value the protocol mechanism enters the congestion avoidance phase. In this phase, the congestion window is increased by the amount of $1/cwnd$ for each acknowledgment ACK, which is equivalent to increasing of the window by one packet for the double-turn.

Protocol TCP Reno monitors two options of packet loss:

- Triple Duplicate ACK (TD). Let n-th package is not delivered, and subsequent packets ($n+1$, $n+2$, etc.) are delivered. For each packet delivered in violation of prioritization(for $n+1$, $n+2$, and so on), the recipient sends ACK message for the last undelivered (n-th) package. With receiving three such packets the source resends the n-th package. In addition, the window size is decreased by 2 times $cwnd \to cwnd/2$.
- Timeout (TO). While sending a package the timeout timer is started. After receiving the confirmation the timer is restarted. Wherein the window size is set to the initial value of the congestion window. The first lost package is resent. The protocol passes into a slow start phase.

The TCP protocol uses multiple timers. The main one is *Retransmission timer*. It restarts each time when a segment is sended. If an acknowledgment does't received before the timer triggering, then the segment is retransmitted.

The retransmission timer doesn't change in the slow start phase and in the congestion avoidance phase. It linearly decreases over time in the fast recovery phase and in the timeout phase.

Thus we can write the equations for the window evolution in TCP protocol in Modelica language (Listing 1).

Listing 1. Window evolution in TCP protocol

```
// Fast Recovery
when (pre(state) == TCPState.slowStart or pre(state) ==
   TCPState.congestAvoid) and state == TCPState.fastRecov then
  reinit(retr_timer, o.RTT);
  reinit(ssth, w / 2);
  reinit(w, w / 2);
end when;
// Timeout
when (pre(state) == TCPState.slowStart or pre(state) ==
   TCPState.congestAvoid) and state == TCPState.timeOut then
  reinit(retr_timer, RTO);
  reinit(ssth, w / 2);
  reinit(w, 1);
end when;
```

4. Modeling of discrete state transitions

Let's describe the TCP state transitions. Note that since we have already described the internal states of each phase as the equations in Modelica, we can't use the equations to describe the state transitions. Otherwise, the resulting system will be an overdetermined. For such cases, Modelica language provides the *algorithm* statement.

The slow start is the initial state. We use the `ssth` variable for transition from a slow start phase to a congestion avoidance phase. when a new connection is opened, this variable is initialized to the maximum possible size of the window. The slow start phase continues until the value of the `cwnd` reaches the `ssth`, then a transition occurs to the congestion avoidance phase.

When packet loss occurs the TCP goes either into a fast recovery phase, or into a timeout phase. We model this transition empirically, depending on the window threshold (`timeout_th`) [18].

Similarly, the TCP protocol can either go from the congestion avoidance phase into a fast recovery phase, or into a timeout phase.

After retransmission time the TCP protocol goes from a fast recover phase into a congestion avoidance phase, and from a timeout phase into a slow start phase.

Based on the description of the state transitions we can construct the UML-diagram (Fig. 2). The resulting chart can be converted to a Modelica program. We give a fragment of the listing (Listing 2). Here we demonstrate only the state transition algorithm for TCP. As can be seen, it is made by almost verbatim copying of UML-diagrams (it is theoretically possible to carry out the code generation based on the corresponding chart).

5. The transition to the continuous model

Because we want to construct the hybrid continuous-discrete model, we need pass to the continuous-time model in order to describe the operation of each TCP phase. The transition between the phases will be described by discrete states.

Using the results of section 3, the behavior of our model may be formalized. A congestion window change is described by an elementary event, which corresponds to a single acknowledgment or confirmation of all. Let us assume that the elementary event is the arrival of all acknowledgments that occurs during the round-trip time (RTT).

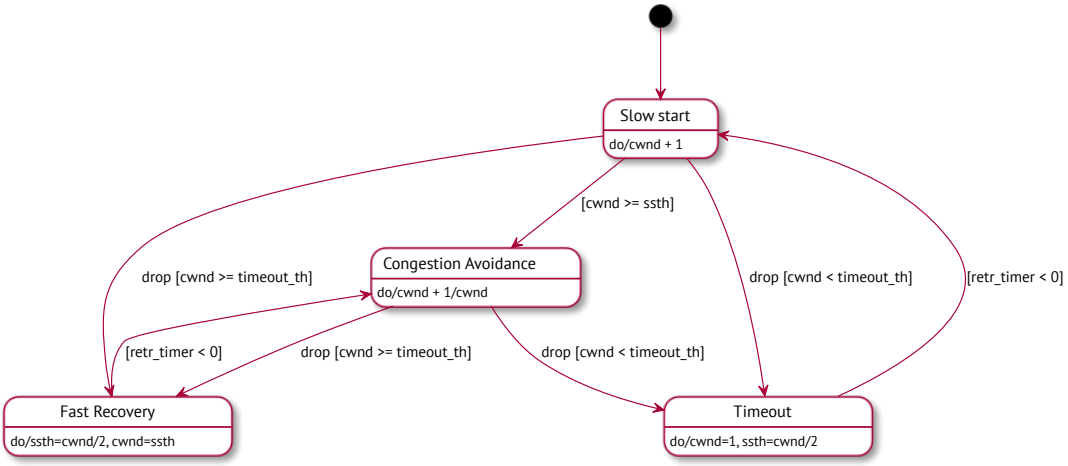

Figure 2. TCP state diagram

Listing 2. State transition algorithm for TCP protocol

```
algorithm
  state := TCPState.slowStart;
  when edge(drop_delay) and w >= timeout_th and (state ==
     TCPState.slowStart or state == TCPState.congestAvoid) then
    state := TCPState.fastRecov;
  elsewhen w >= ssth and state == TCPState.slowStart then
    state := TCPState.congestAvoid;
  elsewhen edge(drop_delay) and w < timeout_th and (state ==
     TCPState.slowStart or state == TCPState.congestAvoid) then
    state := TCPState.timeOut;
  elsewhen retr_timer < 0 and state == TCPState.fastRecov then
    state := TCPState.congestAvoid;
  elsewhen retr_timer < 0 and state == TCPState.timeOut then
    state := TCPState.slowStart;
  end when;
```

In the slow start phase a congestion window size increases with each occurrence of confirmation (ACK):

$$W(t_n^{ACK} + \Delta t^{ACK}) = W(t_n^{ACK}) + 1. \tag{1}$$

We now rewrite (1) relative to the round-trip time T:

$$W(t_n + \Delta t) = W(t_n) + 1 \cdot W(t_n),$$
$$\frac{W(t_n + \Delta t) - W(t_n)}{\Delta t} = \frac{W(t_n)}{\Delta t}.$$

Assuming that $\Delta t = T$, we obtain

$$\frac{\mathrm{d}W}{\mathrm{d}t} = \frac{W}{T},$$
$$\mathrm{d}\ln W = \frac{\mathrm{d}t}{T}, \quad \ln W = \frac{1}{T}; \quad W = \exp\left\{\frac{1}{T}\right\}.$$

Listing 3. Equations for TCP protocol

```
if state == TCPState.congestAvoid then
  // additive increase
  der(w) = 1 / o.RTT;
  o.rate = w * L / o.RTT;
  der(retr_timer) = 0;
elseif state == TCPState.fastRecov then
  der(w) = 0;
  o.rate = w * L / o.RTT;
  der(retr_timer) = -1;
elseif state == TCPState.timeOut then
  der(w) = 0;
  o.rate = 0.001;
  der(retr_timer) = -1;
else
  // state == slowStart
  // exponential increase
  der(w) = w / o.RTT;
  o.rate = w * L / o.RTT;
  der(retr_timer) = 0;
end if;
```

Thus, the window grows exponentially, as it should be in a slow start phase in accordance with the TCP description.

Similarly, we examine congestion avoidance phase. For each occurrence of an ACK the window size is increased:

$$W(t_n^{ACK} + \Delta t^{ACK}) = W(t_n^{ACK}) + \frac{1}{W(t_n^{ACK})}.$$

We rewrite this for a round-trip time:

$$W(t_n + \Delta t) = W(t_n) + \frac{1}{W(t_n)} W(t_n);$$
$$\frac{W(t_n + \Delta t) - W(t_n)}{\Delta t} = \frac{1}{\Delta t}.$$

Assuming that $\Delta t = T$, we have

$$\frac{\mathrm{d}W}{\mathrm{d}t} = \frac{1}{T},$$
$$\mathrm{d}W = \frac{\mathrm{d}t}{T}, \quad W = \frac{1}{T}.$$

The result is a linear increase of the window, as described in the specification of the TCP.

Using these equations, we can write the equations for the TCP protocol in Modelica language (Listing 3).

6. Conclusions

We investigated the TCP protocol as part of a larger system. Some mathematical models (both analytical and simulation) of this mechanism using different paradigms and techniques

were presented. On closer inspection, the presented modeling techniques have shown their shortcomings.

The considered in the article hybrid (continuous-discrete) approach seems to be the most appropriate for network protocol modeling.

Acknowledgments

The work is partially supported by RFBR grants No's 14-01-00628, 15-07-08795, and 16-07-00556. Also the publication was supported by the Ministry of Education and Science of the Russian Federation (the Agreement No 02.a03.21.0008).

References

[1] Paxson V and Floyd S 1997 *Proc. of the 29th conference on Winter simulation - WSC '97* (New York, New York, USA: ACM Press) pp 1037–1044
[2] Paxson V and Floyd S 1995 *IEEE/ACM Transactions on Networking* **3** 226–244
[3] Leland W E, Taqqu M S, Willinger W and Wilson D V 1994 *IEEE/ACM Transactions on Networking* **2** 1–15
[4] Demidova A V, Korolkova A V, Kulyabov D S and Sevastyanov L A 2015 *6th Int. Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)* (IEEE Computer Society) pp 557–562 (*Preprint* 1504.00576)
[5] Eferina E G, Korolkova A V, Gevorkyan M N, Kulyabov D S and Sevastyanov L A 2014 *Bulletin of Peoples' Friendship University of Russia. Series "Mathematics. Information Sciences. Physics"* 46–59 (*Preprint* 1503.07342)
[6] Maler O 1992 *Workshop on Theory of Hybrid Systems* (Lyndby, Denmark: Springer-Verlag)
[7] Maler O 2002 *Annual Reviews in Control* **26** 175–187
[8] Färnqvist D, Strandemar K, Johansson K H and Hespanha J P 2002 *The 2nd Int. Modelica Conf.* pp 209–213
[9] Hespanha J P, Bohacek S, Obraczka K and Lee J 2001 *Lncs* 2034 pp 291–304
[10] Bohacek S and Lee J 2001 *Proc. of the 39th Annual Allerton Conf. on Communication, Control, and Computing* pp 1–10
[11] Fritzson P 2003 *Principles of Object-Oriented Modeling and Simulation with Modelica 2.1* (Wiley-IEEE Press)
[12] Fritzson P 2011 *Introduction to Modeling and Simulation of Technical and Physical Systems with Modelica* (Hoboken, NJ, USA: John Wiley & Sons, Inc.)
[13] Misra V, Gong W B and Towsley D 1999 *Proc. of PERFORMANCE* **99**
[14] Misra V, Gong W B and Towsley D 2000 *ACM SIGCOMM Computer Communication Review* **30** 151–160
[15] Velieva T R, Korolkova A V and Kulyabov D S 2015 *6th Int. Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)* (IEEE Computer Society) pp 570–577 (*Preprint* 1504.02324)
[16] Fall K and Floyd S 1996 *ACM SIGCOMM Computer Communication Review* **26** 5–21
[17] Allman M, Paxson V and Blanton E 2009 TCP Congestion Control Tech. rep.
[18] Padhye J, Firoiu V, Towsley D and Kurose J 1998 *ACM SIGCOMM Computer Communication Review* **28** 303–314