UNDERSTANDING CONGESTION CONTROL ALGORITHMS IN TCP USING OPNET

Teresa Álvarez, Virginia Álvarez, Lourdes Nicolás

Dpt. Engineering Science and Automatic Control, Industrial Engineering School (Doctor Mergelina Building) (SPAIN)
tere@autom.uva.es, viralvcue@gmail.com, lour.nico@gmail.com

Abstract

Simulation is a widespread procedure for testing and understanding real systems. Internet has a great impact in our society. Everyone wants fast and reliable links. But there are situations when congestion happens because the network delay and the number of users fluctuate from one moment to the next. There are different approaches for dealing with congestion: classics (such as drop tail or RED) or process control techniques (PID, fuzzy or predictive control among others). Future engineers need to learn about this topic. It seems logical to simulate this problem, but, should all the different techniques be implemented? Or should the students learn how to program some mechanisms? The answer is neither unique nor easy. This paper presents a combined approach of ready and self-programmed methods combined with tutorials using OPNET.

Keywords: Simulation, computer networks, congestion control, OPNET, TCP, AQM.

1 INTRODUCTION

The importance of simulation cannot be denied. Simulation is basic for testing and learning about real systems. Depending on the process, we will choose a tool or another. Although there are many papers that present results with real networks experiments, most of the work rely on the use of simulation software [1].

Future Computer Science and Telecommunication Engineers (in Spain) learn about Communications Networks (Internet, TCP/AQM) and they also learn process control techniques. Sometimes is a difficult task to find systems that appeals them from both sides: communications and control. Most control books present electrical or mechanical examples. Only in [2] we have found a computing vision.

The congestion control problem in Internet fulfilled our expectations. Huge amounts of data are transferred from one point to another in a matter of seconds, or even less! However, there can be problems: long delays in delivery, lost and dropped packets, oscillations and synchronization problems ([3]). Congestion is responsible for these drawbacks. It occurs when there are too many sources sending too much data too fast for the network to handle, and it is a very serious problem. Thus, it is necessary to reduce this problem as much as possible. At present, there are methodologies to deal with this issue ([5], [6]): congestion control, which is used after the network is overloaded; and congestion avoidance, which takes action before the problem appears. Feedback control techniques can be openly and easily applied to the congestion control problem.

Internet congestion control is carried out in the transport layer at the sources (end systems) and has two parts: the end-to-end protocol TCP (Transmission Control Protocol), and the active queue management (AQM) scheme, implemented in routers. About eight years ago, researchers published the first mathematical models of AQM. From that moment onwards, control theory based approaches have been used to analyze and design AQM congestion control algorithms ([7], [11], [12], [13]).

This problem is rather interesting and it seems reasonable to simulate it, but, which tools should we use? Should all the different techniques be implemented? Or should the student learn how to program some mechanisms? The answer is not unique. It depends on the type of engineers and the course level. Usually, Computer Science students prefer to program something in order to understand it. Telecommunication people do not feel so urgently this need.

We have chosen Opnet as simulation tool because it works with State Finite Machines (SFM), so real network events are handled as close as reality as it could be. The possibilities for adding new
algorithms are great but not straightforward. A very good knowledge on how the software works is required.

So our strategy is two fold:

- We provide the students with some basic examples and step by step instructions on how to simulate routers, queues and basic congestion control algorithms (drop tail and RED).
- Another tutorial explain what to change and what to include and where if new algorithms are going to be tested. CHOKE and DRED are described and results compared with the first tutorials.

From this point onwards, students can simulate more complex topologies and program new strategies without having to spend quite a lot of time in programming issues.

The examples described in the paper are based on a dumbbell topology (figure 1).

This paper presents a step by step description of the tutorials and the experiences with students. Section 2 briefly describe OPNET. Section 3 deals with basic congestion control algorithms. Next, the tutorials are described and finally some considerations and preliminary students’ comments are given.

## 2 OPNET

OPNET Modeler ([19], Figure 2) is a very powerful network simulator. It can be used for research or development. Communication networks, protocols, and all sort of applications can be implemented and studied with great flexibility. The graphical interface allows the user to build models easily. Results can be presented graphically and/or stored for later analysis.

One of the main advantages of OPNET is that it works with finite state machines (FSM). This way, real situations can be implemented more closely to the real world. It provides a graphical editor interface to build models for various network entities from physical layer modulator to application processes. All
the components are modeled in an object-oriented approach which gives intuitive easy mapping to the real systems. It is a flexible platform to test new ideas.

But, there are always inconveniences or difficulties. So many capabilities come with a price. Adding new models requires quite a good knowledge of the structure of OPNET. Our students have the knowledge to program in several languages, but the time required to master OPNET is greater than the slot available.

OPNET Modeler represents the different network components with different modeling levels or paradigms. Each level is related with a domain and an editor. Editors are hierarchically organized: the Project editor depends on the Node Editor that depends on the Process Editor. The objects represents the system's structure and the procedures describe the behaviour.

There are three domains:

- Network domain: the topology of the network is defined.
- Node domain: the nodes are described.
- Process domain: protocols are implemented here.

3  AQM CONGESTION CONTROL TECHNIQUES

This section will present the algorithms included in the tutorials. RED is included in the OPNET's distribution package. But GRED and CHOKE are not.

3.1 RED

The Random Early Detection Algorithm (RED, figure 2) was presented by [8]. A RED gateway calculates the average queue size, using a low-pass filter with an exponential weighted moving average. The average queue size is compared to two thresholds (minimum and maximum). When the average queue size is less than the minimum threshold, no packets are marked. When the average queue size is greater than the maximum threshold, every arriving packet is marked. If marked packets are in fact dropped, or if all source nodes are co-operative, this ensures that the average queue size does not significantly exceed the maximum threshold. When the average queue size is between the minimum and the maximum threshold, each arriving packet is marked with probability $p$, where $p$ is a function of the measured queue length $q$.

3.2 GRED

Gentle Red (GRED, [9]) is a variant of the RED algorithm. The discarding function of GRED is different (figure 1). Going from $\text{max}_p$ to 1 is not an abrupt change; a linear function with slope $(1-\text{max}_p)/\text{max}_th$ is followed until $t=2\text{max}_th$ is reached. It is defined as a Define abbreviations and acronyms the first time they are used in the text, even after they have been defined in the abstract. Do not use abbreviations in the title or heads unless they are unavoidable.

3.3 CHOKE

CHOKE ([10], CHOOSE and Keep for responsive flows, CHOOSE and Kill for unresponsive flows) aims to approximate max-min fairness for the flows that pass through a congested router. Figure 3 describes how it works.
The basic idea behind CHOKe is that the contents of the FIFO buffer form a “sufficient statistic” about the incoming traffic and can be used in a simple fashion to penalize misbehaving flows.

![Diagram of CHOKE algorithm](image)

Figure 3. CHOKE algorithm ([10])

### 3.4 Remarks on the algorithms

RED is one of the default congestion control techniques in OPNET. GRED and CHOKE are not. So, we have included as new methods. The interesting point is that although these three mechanism only differ in the discarding function, quite a lot of changes have to be done in the source code (see next section).

### 4 HOW TO MODIFY OPNET SOURCE CODE

This section describes the changes in the OPNET’s source code. We only present the GRED case, as modifications for CHOKE are rather straightforward.

#### 4.1.1 External files source code changes

In omq_m.h a new constant is defined:

```c
#define OMSC_GRED 3
```

This constant will identify the type of congestion control algorithm to be applied.

In omq_m.ex.c the function omq_m_red_packet_drop is modified:

```c
if ((avge_queue_size >= min_threshold) && (avge_queue_size < max_threshold))
{
    ...}
else {
    if ((red_status == OMSC_GRED) && (avge_queue_size >= max_threshold)
    && (avge_queue_size < max_threshold*2))
        maxp = 1.0 / mark_prob_denominator;
        dropping_probability = 
        (((1.0-maxp)* (avge_queue_size- max_threshold))/max_threshold + maxp;
        random_number = op_dist_uniform (1.0);
        FRET ((Boolean) (random_number <= dropping_probability));
}
```

The new code implements the methodology for discarding packets in GRED.

#### 4.1.2 Process model changes

The process model ip_output_iface needs some changes. How to access it?. From the project editor, we open the node model of one of the routers (mouse double click), open the ip module process
model. Then, in the process model ip_dispatch, using the menu: File>Open Child Process Model >
ip_output_iface. Once there, go to the function block of ip_output_iface.pr.c. A new state that shows the GRED mean queue size is incorporated:

```c
static void output_iface_stats_register(OmsT_Qm_Queue_Pool* qpool_ptr, int q_index)
{
    if (red_status == OMSC_RED)
    {
    ...
    }
    else if (red_status == OMSC_GRED)
    {
        sprintf (new_name, "GRED Average Queue Size %s Q%d%s", intf_name, q_label,
            queue_category_str);
        dim_stathandle = Oms_Dim_Stat_Reg (my_id, "IP Interface", "GRED Average Queue
Size", stat_annotation_str, OPC_STAT_LOCAL);
        Oms_Dim_Stat_Rename (dim_stathandle, new_name);
        oms_qm_statistic_set(qinfo_ptr,OmsC_Qm_RED_Avg_Queue_Size, dim_stathandle);
    }
}
```

Save changes: File>Commit and compile the process model (Compile menu, process editor).

![Figure xx: Adding a new attribute](image)

### 4.1.3 QoS node: attributes modifications

The source code has been modified, but now we have to allow the user to choose the new algorithm
from the OPNET’s menus. Following the next steps, we create a new option in the attribute tree of the
QoS node. Steps:

- Access to the node model of the QoS node.
- Access to the node process model: qos_attribute_definer.
- Edit the attributes:
  - FIFO Profiles > Details > RED parameters > RED Status
- Click Edit Compound Attribute Properties.
- Window (Attribute: RED Parameters), choose Attribute Properties as Private and click OK.
- Select RED Status, and Edit Properties.
- A new window opens with a new Symbol map. GRED is included with the value 3 (this value
  was assigned to the OMSC_GRED constant). Additional methods will be given different
  numbers). Press OK to come back to the previous window.
5 STUDENTS’ RESPONSE

The students’ approach has been very positive. We have not tried the methodology in big groups. It has been more a voluntary assignment. Those students willing to test the program and to learn the insights of it were given extra points. The new method make the undergraduates that followed it obtain higher marks.

The tutorials and files can be downloaded from [www.isa.cie.uva.es/~tere/opnet.html](http://www.isa.cie.uva.es/~tere/opnet.html). Students are given two basic topologies (Figure xx). Taking them as starting point they can build their own configurations. We thought that it was interesting to work with this simple topology because it is much easier to find complex examples than fundamental ones. OPNET’s examples are more complex that ours. Figure xx shows some simulation results.
6 CONCLUSIONS

This paper has presented an approach on how to use OPNET in the classroom to learn congestion control algorithms. Experience has proved rewarding and will be improved in the future. Future enhancements will incorporate the students’ add-ons as new tutorial modules.

REFERENCES


[23] Reference 1 [Arial, 10-point, left alignment, upper and lower case]

[24] Reference 2

[25]