Traffic Generation for Telecom Cloud-Based Simulation

Alba P. Vela*, Anna Vía, Fernando Morales, Marc Ruiz, and Luis Velasco
Universitat Politècnica de Catalunya (UPC), Barcelona, Spain
**e-mail:apvela@ac.upc.edu*

ABSTRACT
With the incremental amount of applications running over the telecom cloud architecture it is becoming of paramount importance being able to run simulations aiming at evaluating the performance of such applications. To that end, one of the key elements in the simulation is how to generate network traffic. In this paper we propose realistic traffic functions that can be used for such purposes and present how those functions have been integrated in our OMNET++-based simulator.
Keywords: traffic generation, network simulation, traffic profiles.

1. INTRODUCTION

Traffic generation is a useful technique that enables studying and evaluating the network performance through simulation, when real traffic traces are not available. In fact, as pointed out by authors in [1], there is an absence of public availability of real world network traces, specifically for traffic in the operators' and Internet service providers core transport networks.

Therefore, due this unavailability, one option is to attempt generating simulated traces which resemble to real ones. Authors in [2] state that it is quite easy to generate traffic, but it is far more difficult to produce traffic that exhibit real characteristics, such as the ones observed through the Internet. Several traffic generators have been developed until present but, to the best of our knowledge, literature is mainly focused on generating representative IP traffic for packet-based networks or connection arrival based on the Poisson distribution for circuit-switched networks [6].

However, to evaluate the performance of operators' network architectures, an intermediate traffic generation is needed in between packet generation and connection arrival modelling to reproduce continuous traffic. In this paper we face this problem and provide details of several traffic generation models to reproduce common traffic profiles. In addition, we present a framework that we integrate into the OMNeT++ discrete event-driven simulator, where its main feature is its modularity.

2. TRAFFIC PROFILES GENERATION

This section presents general traffic profiles and the way they can be generated. Let us denote $f(t;T)$ as the periodic function with period T returning the mean value of the model complex against time traffic. This function $f(t;T)$ can be generated in multiple ways. For instance, one can define a piecewise linear function, i.e., a function composed of a number of linear segments, aiming to reproduce the traffic behavior against time. Another way could consist on a polynomial of some degree, or even a summation of functions, e.g., trigonometric sines.

In addition, some random values around the average value are usually observed as a result, among others, of the monitoring process. That random function ε_t can be modeled as a probability distribution function, e.g., following the normal (Gaussian) distribution. In such case, $\varepsilon_t \sim (\mu, \sigma^2)$ defined by the mean μ and the standard deviation σ. In consequence, the complex traffic profile $Y(t;T)$ to be reproduced can be generated as:

$$Y(t;T) = f(t;T) + \varepsilon_t \quad (1)$$

Let us now present three different traffic profiles, where $f(t;T)$ functions were generated using piecewise linear functions and $\varepsilon_t \sim \mathcal{N}(0,\sigma^2)$ (Fig. 1).

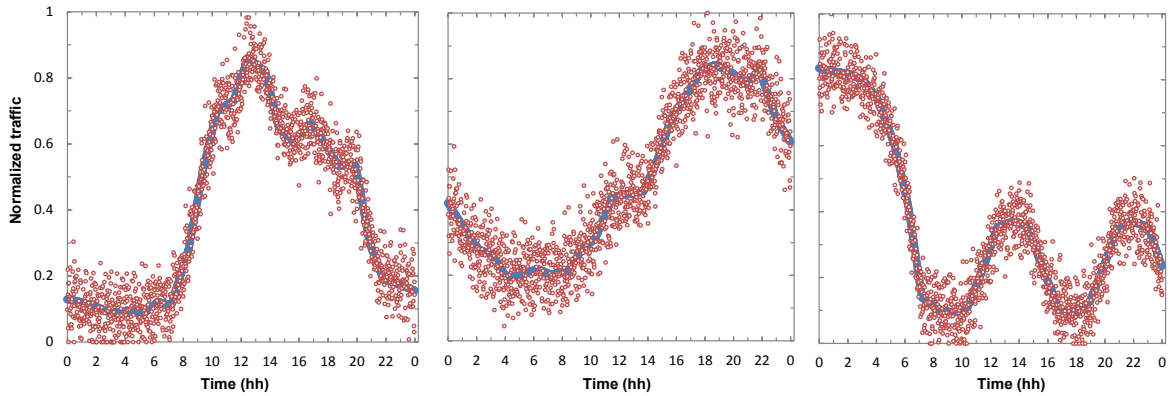


Figure 1. Traffic profiles generated: (a) Business, (b) CDN, and (c) DC2DC.

Figure 1 presents three different cases of traffic generation where the mean function $f(t; T)$ and the random function ε_t are represented as strong blue line and red dots , respectively. Figure 1(a) plots a *business*-like traffic with differentiated traffic intensities during working and night hours; intensity experiences a significant increase around 9h reaching its maximum at midday. Figure 1(b) shows a similar traffic profile focused on the traffic generated by residential users; refer to this as *content delivery network* (CDN). It can be observed how the maximum value is around 17 h. Finally, Fig. 1(c) plots a totally different traffic profile reproducing the traffic exchange among servers in distant datacenters, e.g., for database synchronization; we call this profile as *DC2DC* traffic. The main characteristic of this traffic is that the maximum value is reached during night, whereas day time intensity is virtually zero.

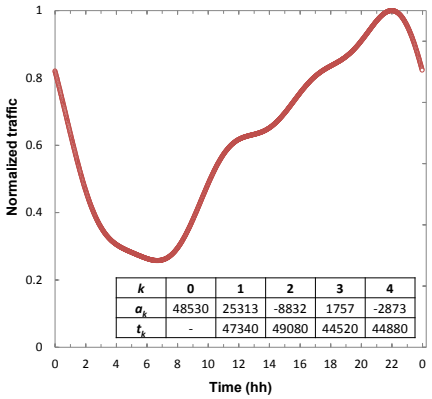
As anticipated above, $f(t;T)$ can be modeled as a summation of trigonometric sines. This alternative way has been used to model periodic traffic due to its repetitive pattern, e.g. in [3], where the authors reproduced traffic obtained from monitoring the Telecom Italia IP national backbone network. Daily profiles from Monday to Friday were very similar but also profiles on Saturday and on Sundays, hence only one model was developed to represent any day of the week (Fig. 2). Equation (2) reproduces the model, where k is the index for the corresponding term in the sinusoidal series expansion, a_0 is the mean traffic intensity, a_k is the amplitude, t_k is the time shift in seconds, and daily period T (in seconds).

$$f(t;T) = a_0 + \sum_{k=1}^{K} a_k \cdot \sin\left[2 \cdot \pi \cdot k \frac{t - t_k}{T}\right] \tag{2}$$

The curve in Fig. 2 represents the traffic intensity generated with eq. (2), applying the values of the coefficients in the table. The equation is limited to the first four terms of the harmonic expansion since higher order terms add hardly any information.

The previous traffic profiles showed how periodicity can be modeled assuming that the profile observed in one day is similar for the rest of the days. However, traffic volume in usually growing when looking at long periods (e.g., one year) and even changes its daily pattern. Equation (3) presents a general formula to generate evolutionary traffic profiles, where $\alpha(t)$ represents the line with value 1 for $t = 0$ and 0 for $t = max\_time$, and parameter β is the growing factor (e.g. 1.5). For instance, Fig. 3 plots a traffic profile where $f_1(t)$ is the business profile and $f_2(t)$ is the CDN profile, previously presented.

$$Y(t;T) = \alpha(t) \cdot (f_1(t) + \varepsilon_{t1}) + (1 - \alpha(t)) \cdot (\beta \cdot f_2(t) + \varepsilon_{t2}) \tag{3}$$



k	0	1	2	3	4
a_k	48530	25313	-8832	1757	-2873
t_k	-	47340	49080	44520	44880

Figure 2. Daily network traffic generated as a summation of trigonometric sines. Coefficients are specified.

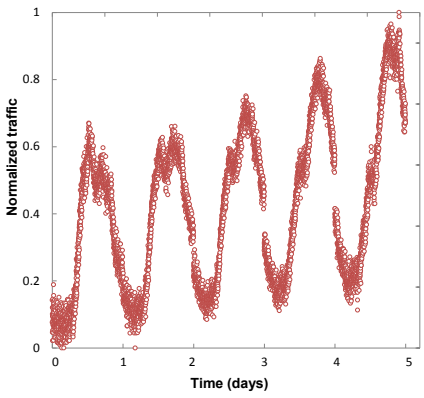


Figure 3. Daily traffic variation with evolutionary traffic profile and incremental intensity.

3. TRAFFIC CORRELATION

Aiming at a more realistic traffic generation, we add *correlation* dependency that accounts previous values of traffic. Note that this is in contrast to completely independent traffic generation, as generate by the models in the previous section. Correlation is a statistical measure that indicates the dependence between two or more random variables; it can be expressed by the correlation coefficient, c, which represents a quantitative measure of the dependence between variables. In our study, a positive correlation is being applied, where related variables are linearly dependent, so they increase or decrease simultaneously; current traffic (in t) values relate with previous k values (in times t-k to t-1), as in eq. (4). Assuming k=1 and operating, we obtain eq. (5).

$$Y(t;T) = f(t;T) + \varepsilon_t + \sum_{i=1}^{k} c_i \cdot Y(t - i;T) \tag{4}$$

$$Y(t;T) = f(t) + \varepsilon_t + c \cdot \varepsilon_{t-1} \tag{5}$$

Figure 4 illustrates the same traffic profiles without [Fig. 4(a)] and with [Fig. 4(b)] correlation. Correlation was generated assuming $k = 1$ with c parameter equal to 0.5. Although is difficult to observe naked-eye, plots in Fig. 4(c) present the normalized values of generated traffic, i.e. $Y(t;T) - f(t)$, where the effect of considering past values of the random function can be slightly intuit in the view of the more extreme values observed.

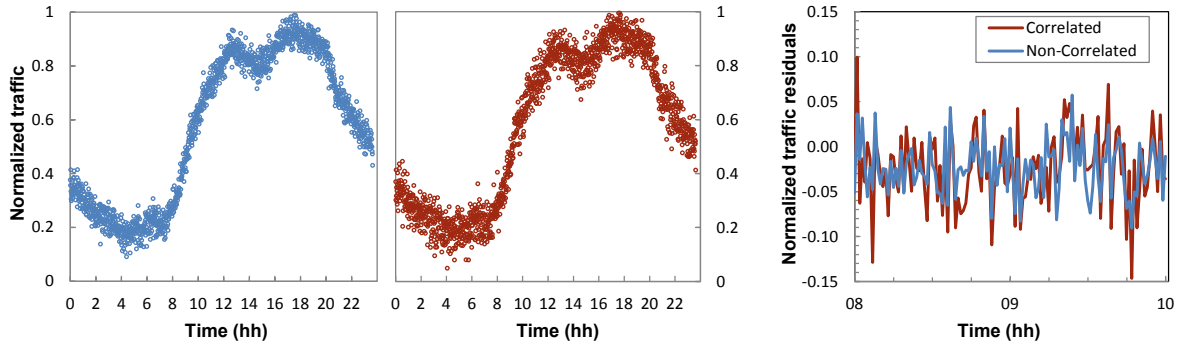


Figure 4: (a) Non-correlated traffic generation; (b) Correlated traffic generation; (c) Normalized traffic residuals for both cases.

4. NETWORK SIMULATOR MODULES AND FUNCTIONALITIES

In this section, we present a traffic generation framework based on OMNeT++ [4]; an object-oriented modular discrete event network simulation framework. OMNeT++ provides a component architecture, where components communicate among others by passing messages. Connectivity between modules (topology) is specified using a high-level language, named as NED.

Our traffic generator, depicted in Fig. 5, consists of a number of functions that can be combined to create traffic profiles as sophisticated as those described in eq. (3). We define three function types: *i*) *unitary functions* are functions defined in the area defined by corner-points (0,0), (1,0), (0,1), and (1,1) and are intended to serve as a base to develop more complex functions. Examples of unitary functions are piecewise and polynomials; *ii*) *distribution functions* are generated with a random seed and are also constrained in amplitude (y axis) but not in the x axis. As an example of distribution functions, there are normal and uniform distributions; *iii*) *custom functions* are unconstrained complex functions defined as OMNeT++ modules. These functions can be created based on unitary and distribution functions with a configurable amplitude value.

The output of custom functions can be connected to the *traffic combiner*. This OMNeT++ module allows generating traffic intensity values by performing arithmetic operations from multiple custom functions. The traffic combiner performs the arbitrary arithmetical expression defined in an input string. Such arithmetical expression might contain any number of additions, subtractions, and multiplications to be performed on the inputs. The traffic combiner uses the Reverse Polish Notation (RPN) [5] to compute the value of the expression.

Aiming at facilitating the composition and reading of arithmetical expressions, they are initially passed in infix notation; once the string defining the expression is loaded, the generator executes a first phase to convert the character string to a vector of tokens (abstract data types). A family of tokens encapsulating the basic elements of arithmetic is created to be able to parse a wide range of arithmetical expressions. These tokens allow identifying subsets of characters in a string with some particular functionality: binary operators (addition, subtraction, and multiplication), custom functions, parentheses (to allow nested sub-expressions) and constants. Additional tokens (e.g. non-arithmetic operators) can be easily added to the family to increase its algebraic extent or to simplify the arithmetic composition. In the next step, the Shunting-Yard algorithm is executed to convert the token vector in infix notation to a token vector in RPN. Once the RPN expression is ready, it is straightforward to compute the value of the expression (given the values of the custom functions) using a stack data structure. Finally, the numerical output is used as incoming traffic for a network node, as presented in Fig. 5.

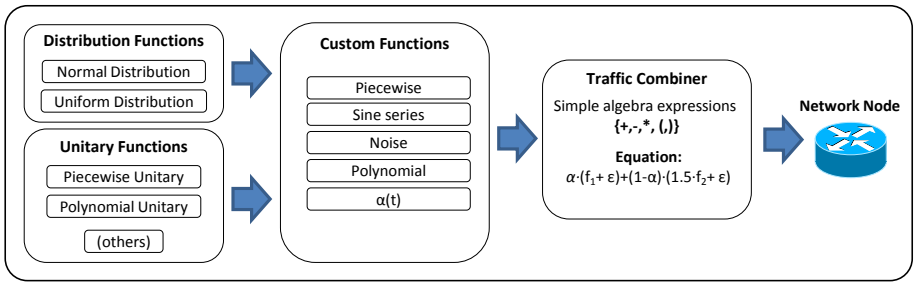


Figure 5. Design of a flexible framework to model traffic generation.

For illustrative purposes, let us show an example of the arithmetical expression implementation in the *traffic combiner module.* As stated above, *custom functions* are unconstrained complex functions defined as OMNeT++ modules that inherit their properties from a basic module. Two functions are defined in that basic module:

```
const std::string& getCustomFunctionName()
virtual double getValue(const simtime_t& simtime)
```

The `getCustomFunctionName` function returns the name of the custom function. That name is defined in a configuration file, where custom functions are given a special name that unequivocally identifies them. The `getValue(simtime)` function receives the current simulation time and computes the traffic value at that time, e.g. Eq. (2). It is worth noting that the `simtime` value needs to be converted to a portion of time defined in the interval [0,T]. That conversion can be easily done as t = `simtime`%T.

To illustrate the arithmetical expression implementation in the *traffic combiner module*, let us consider the traffic pattern generated in Fig. 3 implementing eq. (3); in that example, $f_1(t)$, $f_2(t)$, ε_t, and $\alpha(t)$ are the different custom functions that we want to combine. Particularly, *custom function* $f_1(t)$ implements the business-like traffic profile and its `getCustomFunctionName()` function will return the string "*Business"*. The same applies to *custom function* $f_2(t)$, which represents CDN-like traffic profile and is assigned the "*CDN"* string, the random component ε_t is assigned the string "*Random*", and finally $\alpha(t)$ is assigned the string "*Alpha*".

Equation (6) details the input string for the Traffic Combiner module so as to generate the output traffic $Y(t)$. Note that every custom function name is enclosed inside the underscore character. In addition, real numbers can be also used inside the expression.

$$Y(t) = \text{"\_Alpha\_ * (\_Business\_+\_Random\_) + (1-\_Alpha\_) * (1.5 *\_CDN\_ + \_Random\_)"} \quad (6)$$

5. CONCLUSIONS

Traffic generation is an important tool to study and characterize the network. As presented through the paper, different type of functions like piecewise linear, polynomial or trigonometric sine summation can be useful to model traffic profiles. Besides, traffic is characterized by a random function around the average, which can be modelled by a normal distribution. Notwithstanding traffic generation is usually focused on modelling traffic, in this paper time evolutionary traffic profiles have also been covered.

A traffic generation framework based on OMNeT++ is developed to provide the component architecture needed to build up our traffic generator; consisting of four connected modules with differentiated purposes. Two basic function modules are connected to a more sophisticated function custom module, which in turn is connected to the traffic combiner where the numerical value output for traffic is computed following a configurable equation. Traffic values are used as input for the network node.

ACKNOWLEDGEMENTS

The research leading to these results has received funding from the Spanish MINECO SYNERGY project (TEC2014-59995-R) and from the Catalan Institution for Research and Advanced Studies (ICREA).

REFERENCES

[1] P. Kamath, K. Lan, J. Heidemann, J. Bannister, and J.Touch, "Generation of high bandwidth network traffic traces," in *Proc. IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer Systems,* 2002.

[2] A. Varet and N. Larrieu, "How to generate realistic network traffic," in *Proc. IEEE 38th Annual International Computers, Software & Applications Conference (COMPSAC)*, 2014.

[3] A. Castro, L. Velasco, M. Ruiz, M. Klinkowski, J. P. Fernández-Palacios, and D. Careglio, "Dynamic routing and spectrum (re)allocation in future flexgrid optical networks," *Computer Networks*, vol. 56, pp. 2869-2883, 2012.

[4] STRONGEST Project, Deliverable 2.1, "STRONGEST: Scalable, Tunable and Resilient Optical Networks Guaranteeing Extremely-high Speed Transport," 2010.

[5] OMNET++: http://www.omnetpp.org/

[6] A. Burks, D. Warren, and J. Wright, "An analysis of a logical machine using parenthesis-free notation," *Mathematical Tables and Other Aids to Computation*, vol. 8, pp. 53-57, 1954.