Abstract
Developers of network applications or network hardware need simulations to test their products before introducing them into the market. Today, several traffic models are available that can generate packet streams with a behavior close to real-life network traffic. Most of these are discrete models, and thus generate a sequence of packet or byte counts for subsequent time intervals.

This generated traffic now has to be imported into the OPNET simulator. This paper will study the possibility of entering such bin counts into OPNET as (background) traffic flows. Optimal settings will be sought to make sure the desired traffic behavior is achieved.

Introduction
In the past, network designers often dimensioned their networks based on simulations with Poisson traffic. Meanwhile, researchers proved that real-life internet traffic does not behave like Poisson traffic at all [1-4]. Measured traffic showed to be a lot more bursty and, as a consequence, network elements are much heavier loaded than in the theoretical Poisson case.

New, more complex models that describe this fractal traffic were developed [5-9]. Most of these are discrete models. This means that the models generate a vector of values, with each element representing the number of packets that arrive in a certain time interval. To obtain a packet timestamp stream, for every time interval of this so-called bin vector, a number of timestamps have to be generated.

Simulation tools like OPNET Modeler can then be used to study the behavior of an application or the impact on network elements when loaded with this bursty, real-life traffic. The problem when entering these traffic flows into OPNET is often situated in their very large size. This causes at least two problems: the project files get very large in size, slowing down the use of the simulation tool, and the simulations need lots of computational effort, making simulation times quite long.

Both these problems could be solved by importing the modeled bin vector into OPNET instead of the timestamp trace. Now only two values (packet count and byte count) have to be specified per time interval. Details on how we try to import this traffic into OPNET are given in the next section. Next, the accuracy of this importing method is studied respectively for a generic packet stream with exponential interarrival times and for a measured traffic trace. In the last section conclusions are drawn from our study.

Importing traffic flows in OPNET
In this section, we’ll briefly explain the method we used to import traffic into an OPNET scenario. This can be done in several ways. OPNET makes a distinction between (analytical) background traffic and explicit traffic. The latter can for example be entered with the ACE module. All traffic is generated on a packet-by-packet basis, and simulations are thus very accurate, but can also be very time-consuming. The former can be imported in several ways, for example from an ASCII file or from several packet sniffer formats. Background traffic is not explicitly generated. When an explicit packet arrives at a certain queuing system, the delay for this packet is calculated by taking into account the background traffic passing through this component. The main advantage of this simulation technique is the increased speed, but with the drawback of being somewhat less accurate under certain circumstances. Another possibility of OPNET is combining these two simulation techniques, what is called “hybrid simulation”. This should make a trade-off possible between the advantages and disadvantages of simulating both traffic types.

We want to enter our traffic using a simple ASCII format, and we want to use the entered traffic in a hybrid simulation. Therefore, we imported the traffic flows into OPNET using the .tr2 format. With this file format, you can enter the mean packet interarrival time and the mean packet size in subsequent intervals. These intervals correspond to the bins used in discrete traffic models.

Entering a bin vector into OPNET is quite straightforward for this file format. For each bin – or equivalent time-interval – the number of packets inside this bin is divided by the bin length, and thus results in the average number of packets per second. The packet size is kept constant in this paper. The second parameter – the number of bits per second – is thus simply a multiplication of the number of packets per second and the constant packet size.

When we want to import a traffic stream consisting of a series of timestamps into OPNET, we can use the .tr2 format if we first divide the timeseries into intervals (or “bins”). To get an as exact as possible representation of the traffic stream, we need to choose these intervals as small as possible. The next steps are equal to the case of entering a bin vector.

For our simulations, we entered a simple scenario into OPNET with one “ethernet4_switch_adv” and two “ethernet_server_adv” components, all interconnected by 10 Gbps ethernet links (Figure 1). The background traffic is entered as a single flow

---

1 Kristof Sleurs is a Research Assistant of the Fund for Scientific Research Flanders (FWO-Vlaanderen), Belgium.
between the two Server nodes. The last component used is the “Background Traffic Config” block, which allows us to set the distributions (e.g. exponential) used for the packet interarrival times and the packet sizes.

![Used OPNET scenario](image)

**Figure 1: Used OPNET scenario**

**Generic packet stream**

We start our study with an artificially generated and thus well-known packet stream. Timestamps are generated with exponentially distributed interarrival times. The mean packet arrival rate is set at 90 packets per second, and a two-hour long packet stream is created with the “exprnd()” Matlab-command [10]. This stream will be used for all simulations in this section. The “Packet Service Rate” of the Switch is set at 100 packets/second, which corresponds to a load of 90 % on the switch. The packet size variability of the “Background traffic config” component is set at “constant”, and the packet interarrival time variability is set at “exponential (Auto calculated)”.

First, the theoretically expected values for the queuing delay and the average queue size will be briefly discussed. The chosen parameters for the packet interarrival times, the queue size and service times lead to the well known M/D/1 queuing system. This model assumes a service time with zero spread, as is the case in our simulation. The load $\rho$ can be calculated by dividing the service time by the mean packet interarrival time, and was set at 0.9 in this case. The packet arrival rate $\lambda$ and the packet service rate $\mu$ equal respectively 90 and 100 packets per second. The expected value of the number of packets in the system and the average packet delay in the queuing system are described by the Pollaczek-Khintchine equations [11], where the proper values for interarrival time mean and spread are filled in:

$$E(n) = \frac{\rho}{1 - \rho} \left(1 - \frac{\rho}{2}\right)$$

$$E(T) = \frac{1/\mu}{1 - \rho} \left(1 - \frac{\rho}{2}\right).$$

The relationship between average packet delay in the system $E(T)$ and average number of packets in the system $E(n)$ is also given by Little’s formula [11]:

$$E(n) = \lambda \cdot E(T).$$

From these equations, we calculate the theoretical average packet delay in the system to be 0.055 seconds, and the average number of packets in the system to be 4.95.

Next, some OPNET simulations were carried out. The packet stream was divided into an increasing number of bins, and then entered into OPNET using a .tr2 file. Two hour long simulations were run, and the packet arrival rate, queue size and queuing delay were measured at the switch. When we plot the average packet arrival rate versus the average number of packets per bin (Figure 2), we notice that the packet arrival rate diverges from the desired value of 90 packets per second for small numbers of packets per bin. This of course causes the buffers of the switch to fill up, resulting in increasing queuing delays. From this figure, we can set a minimal number of packets per bin, necessary to get realistic results for a packet stream with exponential interarrival times. Changing the load on the switch (by altering its packet service rate) doesn’t change this value because the problem is not a queuing phenomenon, but is inherently connected with the method of generating traffic by the simulator.

![Traffic received by the Switch, for different explicit to background traffic ratios](image)

**Figure 2: Traffic received by the Switch, for different explicit to background traffic ratios**

What could change this threshold is the ratio explicit/background traffic used for simulation. Previously, this was set at 100 % explicit traffic. In Figure 2, graphs are also given for ratios of 0 %, 0.1 % and 1 %. From this figure, we conclude that for an average number of packets per bin larger than $5e+09$ the generated number of packets per second gets close to the target value of 90 packets/second. This value holds for all traffic type ratios.

Figure 3 plots the queuing delay for an increasing number of packets per bin and for different explicit/background ratios. The zone of less than 10 percent deviation from the target value is designated with the dotted lines. We can see that the traffic ratios with some explicit traffic remain within these borders for an average number of packets per bin larger than $2.5e+10$. For the completely explicit traffic scenario, even half that value is good enough. The total background traffic scenario doesn’t remain within the 10 % borders, probably because not enough data points are generated for large bins.
The first problem when importing traffic like this, is that OPNET makes use of extra Tracer Packets when simulating background traffic, even if this traffic is set at “All Explicit”. These packets are used for creating delays and filling buffers on intermediate devices in the network, and they of course are very useful when performing simulations with analytic background traffic. The value of this simulation variable can be set with the “Global attributes” in the “Simulation Setup” dialog. In previous simulations, this variable was set at the default value of 2. The minimum number of Tracer Packets per interval that can be chosen is 1. When bins get smaller, this extra packet in each bin increases total traffic a lot and causes buffers to fill up and eventually overflow. When the number of Tracer Packets per interval is decreased from two to one, the increase of generated packets per second for low numbers of packets per bin gets less steep. This can be seen by comparing Figure 4 with Figure 2. This is an indication of the importance of the Tracer Packets. Note that the “Tracer Packet Redundancy” attribute is disabled.

A solution for this problem is comprised of two changes in the inputted bin vector. First, subsequent empty bins are grouped together. This does not imply any unwanted change because no packets should be generated in these bins. Of course, when grouped, only one tracer packet per group is generated, which decreases the number of unwanted packets generated when there are lots of subsequent empty bins. Second, subsequent non-empty bins with the same packet count are also grouped and the number of packets to be generated per second is decreased by one divided by the total grouped interval size. Decreasing the number of packets to be generated is also done for non-grouped bins. These two measures keep the number of packets per second generated in the simulation under control. Results are shown in Figure 5 and Figure 6. Especially for the “100 % Explicit” case, the average traffic received remains very close to the ideal value now. As can be seen, the queuing delay deviation starts at approximately the same number of packets per bin as before, but it remains close to the target value, opposed to Figure 3. This is due to the smaller error on the average traffic received by the switch.
bin are not modeled at all. There is no reason that these interarrival times are exponentially distributed. The previous technique already reduced these not-modeled interarrival times by grouping bins together. Further study on the packet generation process in OPNET should be done to address this problem.

**Measured packet stream**

Previously, we generated a packet stream with exponential interarrival times and also reproduced it with exponential interarrival times. In this case, a good approximation of the queuing characteristics is quite obvious. Therefore, we also used a measured packet trace as original stream to be imported into OPNET. We expect the packets per bin threshold to be close to that in case of the generic packet stream, but the queuing delay accuracy is more difficult to predict.

The used packet stream is part of the AbileneIII measurements, publicly available on the PMA NLANR website [12-13]. This trace was collected on the Abilene network at the link between the Indianapolis router node and Kansas City on June 1st, 2004. The part from Indianapolis to Kansas City between 20.00h and 20.02h was taken for further study. We first verified the stationarity of this stream.

The same OPNET scenario was used with a few adjustments. The average packet arrival rate $\lambda$ now equals 1.207128e+05, and to maintain a load of 0.9, the packet service rate $\mu$ of the switch was set at 1.341253e+05 packets per second. Traffic flows were again imported from a .tr2 file, and simulations were run for traffic mixes of “All Explicit”, “1 % Explicit”, “0.1 % Explicit” and “All Background”.

Because of the exponential packet interarrival time variability (Set at the “Background Traffic Config” component), we expect the packet stream to behave like in the previous section for very small numbers of bins. From the Pollaczek-Khinchine equations, we predict a theoretical average delay of 4.101e-05 seconds. When the number of bins grows larger, and the average number of packets per bin gets smaller, the distribution of interarrival times inside the bins gets less important. Thus, we expect the non-exponential behavior to become apparent for larger numbers of bins. As the measured traffic is more bursty than theoretical Poisson traffic, we expect the real average packet delay time to be larger than the value predicted by the M/D/1 formulas. To calculate this value, we fed the measured packet trace to a queue system simulated in Matlab. The average packet delay resulting from this simulation is 2.430e-04 seconds. This is the delay value we want to get out of the OPNET simulations. We’ll search for the number of bins – or equivalently the average number of packets per bin – necessary to get close to this value.

Figure 7 and Figure 8 show the simulated average number of received packets and the simulated average packet delay time for an increasing number of packets per bin. Figure 7 shows that the average number of received packets remains reasonably stable. The upper dashed line in Figure 8 indicates the target value for average packet delay of the measured packet trace. The lower dashed line shows the average packet delay in case of exponentially distributed interarrival times. We can see that for 100 % explicit traffic the predictions of the previous paragraph show up nicely. For more background traffic, the delay varies more between the two dashed lines.

![Figure 7: Traffic received by the Switch, for different explicit to background traffic ratios, one Tracer Packet per interval and altered OPNET input](image1)

![Figure 8: Average queuing delay in the Switch, for different explicit to background traffic ratios, one Tracer Packet per interval and altered OPNET input](image2)

**Conclusion**

After making some adjustments to the input bin vector, errors on the mean packet arrival rate can be kept under control. A certain deviation however remains present. An important requirement for entering a bin vector with very small bins into a simulator is that the generated number of packets should match the exact number of packets for each bin, given by the inputted bin vector. Second, not only intra-bin packet interarrival times, but also interarrival times between packets belonging to subsequent bins should be modeled.

It seems that these two requirements are satisfied in OPNET for 100 % explicit traffic, and as a consequence, the average packet queuing delay is quite well modeled for large numbers of bins. The approximation for smaller numbers of bins is not very good due to the exponential interarrival time distribution inside the bins. This could possibly be improved by choosing another distribution for the packet interarrival times inside the bins.
For large bin vectors, simulating the traffic explicitly can be very time-consuming though. Therefore, the usage of “Hybrid Simulation” was investigated. Results were however not as accurate as in the “100 % explicit” case. To explain (and improve) this behavior, more detailed insight in the OPNET Hybrid Simulation method than given by the manual is necessary.

Making it possible to import binned background traffic accurately into OPNET remains a very useful topic for future research. It could make simulations with real-life, measured traffic traces more feasible by using the hybrid simulation technique. Collaboration with simulation tool developers will however be necessary, since all implementation details should be known and optimized to create a solution for this problem.

References


Acknowledgements
We are indebted to the Flemish government, which partly sponsors this research through the generic university basic research program (GBOU) “End-to-End QoS in an IP Based Mobile Network”.


