Comparative study of LTE simulations with the ns-3 and the Vienna simulators

Thiago Abreu
Université Pierre et Marie Curie - LIP6
75005, Paris, France
thiago.wanderley@lip6.fr

Bruno Baynat
Université Pierre et Marie Curie - LIP6
75005, Paris, France
bruno.baynat@lip6.fr

Marouen Gachaoui
LIA/CERI
University of Avignon
84000, Avignon, France
marouen.gachaoui@univ-avignon.fr

Tania Jiménez
LIA/CERI
University of Avignon
84000, Avignon, France
tania.jimenez@univ-avignon.fr

Narcisse Nya
Université Pierre et Marie Curie - LIP6
75005, Paris, France
narcisse.nya@lip6.fr

ABSTRACT

In this paper we compare two simulators: ns-3 and Vienna, in the context of LTE networks, on four basic scenarios for which well-known analytical results exist. These scenarios differentiate themselves by the nature of the traffic (data or voice) and by the number of sources (infinite or finite). Our goal is twofold. First, by confronting the results of the two simulators with exact results, we can assess the accuracy of both simulators and compare their efficiency. Second, and maybe more importantly, we want to compare the ease of handling and use of both simulators, and list the difficulties encountered in the context of the four basic scenarios, that will necessarily arise in more realistic simulated scenarios, and explain how we worked around the problems. We hope this comparison will help researchers who work on LTE networks to choose the simulator that best suits their needs.

Keywords

Simulators, ns-3, Vienna, LTE.

1. INTRODUCTION

The Long Term Evolution (LTE) standard, specified by the 3rd Generation Partnership Project (3GPP) in release 8, is the next step forward in cellular 3G services. LTE offers significant improvements over previous technologies such as Global System for Mobile communications (GSM), Universal Mobile Telecommunications System (UMTS) and High-Speed Packet Access (HSPA) by reforming the core network and introducing a novel physical layer. The main reasons of these changes in the Radio Access Network (RAN) system design are the need to provide higher spectral efficiency, lower delay and more multi-user flexibility than the currently deployed networks [1].

The Idefix project [2] proposes intelligent network and service control mechanisms that ensure constancy of QoS in time and space, while maintaining energy consumption at reasonable levels. For this goal there is a strong need for testing the proposed mechanisms and algorithms by simulation. Many options of simulators were studied, among them: ns-3, Vienna, SimuLTE and home-made simulators. The project finally chose to compare ns-3 and Vienna, because they are felt as the best-suited simulators in the context of LTE networks. Indeed Vienna is purely dedicated to LTE networks, whereas ns-3 is much more general and is based on standard modules designed to simulate different technologies and scenarios. For our comparison we used the LTE module developed by the LENA Project. This module was designed following the 3GPP LTE standard specifications and is included in the main distribution of the ns-3 latest versions.

In order to provide an efficient and methodic comparison of ns-3 and Vienna LTE simulators, we have chosen to consider four basic scenarios, for which exact analytical results exist. These scenarios differentiate themselves by the nature of the traffic (Data or Voice) and by the number of sources (Infinite or Finite). The "Voice Infinite source (VI)" and "Voice Finite source (VF)" scenarios correspond to the well-known Erlang and Engset results, whereas the "Data Infinite source (DI)" and "Data Finite source (DF)" scenarios correspond to open and closed Processor Sharing queues. We compared the performance indexes obtained by both simulators to the analytical expressions provided by models. The objective was first to assess the accuracy of both simulators, and compare their efficiency in terms of run-time, memory usage and CPU occupation. Our goal was also to confront the ease of handling and use of the two tested simulators and evaluate the efforts that are necessary in both cases to handle more realistic scenarios. Finally we list all the dif-

SIMUTOOLS 2015, August 24-26, Athens, Greece

DOI 10.4108/eai.24-8-2015.2260782

ficulties we have encountered in the implementation of our four basic scenarios, and propose possible solutions to work around the problems. We believe that these problems will arise in more complex scenarios and we hope that our solutions may be helpful for researchers that will have to use simulation in the context of LTE networks.

To the best of our knowledge this is the first work that provides a methodic comparison of simulators in the context of LTE networks. However many comparisons of simulators have been made in other contexts. In [3] authors compare ns-3 with other simulators for a grid network simulation. The comparison focuses on scalability, and they show that ns-3 had the best overall performance of the five simulators in their study. More recently [4] compares ns-3 to three other simulators, including their precursor ns-2, by simulating a MANET routing protocol. This paper also concludes that ns-3 shows the best overall performance among the studied simulators. In [5] a detailed comparison of the main characteristics of many (discret event) network simulators is presented, but the comparison remains qualitative and they do not provide any performance evaluation of the tools. Authors in [6] compare ns-2 and JiST/SWANS on the specific case of Ad-hoc networks. Authors in [7] compare the accuracy of ns-2 and OPNET to an experimental setup. A comparison between the exact analytical model of several queues (M/M/1,M/D/1,D/M/1) and simulations in ns-2 can be found in chapter 9 of [8].

This paper is organized as follows. In the next section we briefly present the analytical models corresponding to the four basic scenarios used for comparison. Section 3 gives an overview of the two simulators. The main results of the comparison are provided in Section 4. Finally, Section 5 concludes this paper.

2. SIMULATED SCENARIOS

In order to provide an efficient and methodic comparison of ns-3 and Vienna LTE simulators, we have first chosen to consider four basic scenarios, for which exact analytical results exist. In all of these scenarios, we consider a single LTE cell and we only model the downlink traffic, i.e., traffic from central eNodeB to User Equipments (UEs). We assume that UEs are static (no mobility is considered) and make the assumption that all UEs use the same Modulation and Coding Scheme (MCS) over the whole surface of the cell. The total bandwidth of the cell can be divided into N_s elementary resources (each one corresponding to a pair of resource blocks). Assuming a single MCS for the whole cell corresponds to having a constant number m of bits that can be carried by any of the N_s elementary resource per frame. If we denote by T_f the transmission duration of a frame (1ms in LTE), the cell capacity is given by: $C = \frac{mN_s}{T_f}$bit/s. In this section we rehash the exact analytical results corresponding to the four scenarios. These scenarios differentiate themselves by the traffic (data or voice) and by the number of sources (infinite or finite). Although these results are well known (see e.g., [9]) we quickly rehash them so that the paper becomes self-contained.

2.1 First scenario: data transmissions with infinite source (DI)

We first consider data traffic and assume that there is an infinite number of sources that are likely to generate traffic in the cell. We thus assume that requests for transmission globally arrive to the cell according to a Poisson process with rate λ. Each request brings an identically distributed volume of data to be transferred of mean x_{on}. The transmission capacity C of the cell is equally shared among all active users, which is implemented by means of a Round-Robin discipline. It is well-known that this system can be modeled by an M/M/1/∞/∞/Processor-Sharing (PS) queue with an arrival rate λ and a service rate $\mu = \frac{C}{x_{on}}$. The straightforward Markov chain associated with the PS queue is given in Figure 1 (and is similar to that associated with a M/M/1/FIFO queue).

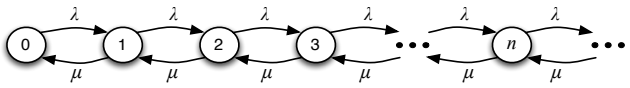

Figure 1: Markov chain associated with Scenario 1

If we denote by $\rho = \frac{\lambda}{\mu}$, it is widely known that the stability condition of the queue is $\rho < 1$, and provided it is satisfied, the stationary probabilities of having n ongoing transmissions are given by:

$$p(n) = (1-\rho)\rho^n, \quad n \geq 0 \tag{1}$$

Accordingly, the average number Q of active users in the system is:

$$Q = \sum_{n=1}^{\infty} np(n) = \frac{\rho}{1-\rho} \tag{2}$$

From Little's law we obtain the average sojourn time of a user in the cell:

$$T = \frac{Q}{\lambda} = \frac{1}{\mu - \lambda} \tag{3}$$

Finally, we can derive the average throughput obtained by users during their transfer in the cell (in bit/s) as:

$$\gamma = \frac{x_{on}}{T} = C(1-\rho) \tag{4}$$

2.2 Second scenario: data transmissions with finite source (DF)

We still consider data traffic but we now assume that there is a finite number N of sources that are likely to generate traffic in the cell. Each one can be either idle or active, and generate traffic in the cell only when it is active. We still assume that the average data volume that any active user has to transferred is x_{on}, and we denote by t_{off} the average time a user remains idle between two successive data transfers. The state of each user can thus be modeled as an ON-OFF process, and we assume that both the volume transferred during an ON period and the time spent during an OFF period are exponentially distributed (with respective means x_{on} and t_{off}). This system can be modeled by an $M/M/1/N/N/PS$ queue with a service rate still given by $\mu = \frac{C}{x_{on}}$ and an arrival rate $\lambda = \frac{1}{t_{off}}$. The Markov chain associated with this queue is given in Figure 2.

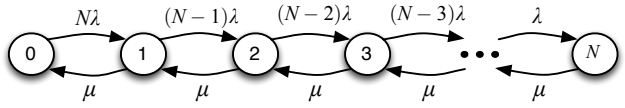

Figure 2: Markov chain associated with Scenario 2

If we still define $\rho = \frac{\lambda}{\mu}$, the queue length distribution is now given by:

$$p(n) = \frac{N!}{(N-n)!}\rho^n p(0), \quad n = 0, ..., N \tag{5}$$

with $p(0)$ obtained by normalization. Note that there is no stability condition for this system (as it is actually a closed system). We can deduce the average number of active users in the cell:

$$Q = \sum_{n=1}^{N} np(n) \tag{6}$$

The average number of users that become active by unit of time is:

$$D = \sum_{n=0}^{N-1} p(n)(N-n)\lambda \tag{7}$$

From Little's law we obtain the average time a user stays active in the cell (between two successive idle times):

$$T = \frac{Q}{D} \tag{8}$$

Finally, we can derive the average throughput obtained by users during their transfer as:

$$\gamma = \frac{x_{on}}{T} \tag{9}$$

2.3 Third scenario: voice transmissions with infinite source (VI)

We now consider voice traffic and assume, like in the first scenario, an infinite number of sources. A voice transmission requires a throughput of v bit/s. The total capacity C of the cell thus enables a maximum number of simultaneous voice calls given by:

$$S = \lfloor \frac{C}{v} \rfloor \tag{10}$$

If we assume that requests for voice calls arrive according to a Poisson process with rate λ, and that call duration are exponentially distributed with rate $\mu = \frac{1}{t_{on}}$ (where t_{on} is the mean call duration), the system is modeled by a classical $M/M/S/S$ Markovian queue with S servers and a total capacity limited to S. Figure 3 gives the associated Markov chain.

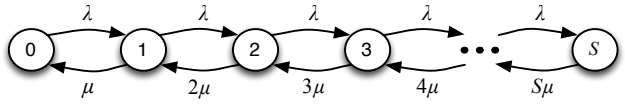

Figure 3: Markov chain associated with Scenario 3

Still taking $\rho = \frac{\lambda}{\mu}$, the queue length distribution is given by the classical Erlang-B formula:

$$p(n) = \frac{\rho^n}{n!}p(0), \quad n = 0, ..., S \tag{11}$$

with $p(0)$ obtained by normalization. Note again that there is no stability condition for this system (as it is limited). We can then calculate the average number of ongoing voice transmissions in the cell:

$$Q = \sum_{n=1}^{S} np(n) \tag{12}$$

Finally, PASTA property enables us to derive the rejection probability of a call request:

$$P_r = p(S) \tag{13}$$

2.4 Fourth scenario: voice transmissions with finite source (VF)

We finally consider voice traffic and assume, like in the second scenario, a finite number N of sources. Each source follows an ON-OFF process, an ON corresponding to a voice call and an OFF corresponding to the idle time between two successive calls. We assume that both the time spent during an ON period and the time spent during an OFF period are exponentially distributed, with respective means t_{on} and t_{off}. We define the arrival rate $\lambda = \frac{1}{t_{off}}$ and the service rate $\mu = \frac{1}{t_{on}}$.

We need to distinguish two cases: 1) $N \leq S$, in which case the capacity of the cell is enough to serve all the N users simultaneously. As a result, no call rejection can occur; 2) $N > S$, in which case the capacity of the cell is not enough to serve all users simultaneously, and call rejections can happen. Markov chains associated with the two cases are given in Figure 4. In both cases there is no stability condition for the system.

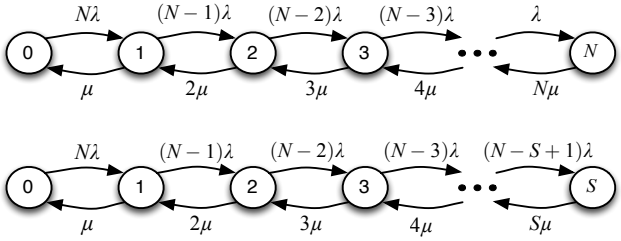

Figure 4: Markov chains associated with Scenario 4 (up: $N \leq S$, down: $N > S$)

For the first case, $N \leq S$, the queue length distribution is:

$$p(n) = \frac{N!}{n!(N-n)!}\rho^n p(0), \quad n = 0, ..., N \tag{14}$$

with $\rho = \frac{\lambda}{\mu}$ and $p(0)$ obtained by normalization. The average number of active users in the cell is:

$$Q = \sum_{n=1}^{N} np(n) \tag{15}$$

And as the capacity of the cell is enough to serve all users simultaneously, there is no possible rejection (the rejection probability is null).

If we now consider the second case, $N > S$, the stationary probabilities have the same expression, excepted that there

are limited to $n = S$:

$$p(n) = \frac{N!}{n!(N-n)!}\rho^n p(0), \quad n = 0, ..., S \tag{16}$$

with $p(0)$ obtained by normalization. The average number of active users in the cell is now:

$$Q = \sum_{n=1}^{S} np(n) \tag{17}$$

And the rejection probability can be expressed as the following rate ratio:

$$P_r = \frac{p(S)(N-S)\lambda}{\sum_{n=0}^{S} p(n)(N-n)\lambda} \tag{18}$$

3. VIENNA AND NS-3 LTE SIMULATORS

We present in this section the main characteristics of the two simulators we compare on the four scenarios of Section 2.

3.1 Vienna Simulator

The Vienna LTE System Level Simulator was created by the Institute of Telecommunications at the Vienna University of Technology. It is an open source software developed in MATLAB using the Object-Oriented programming (OOP) capabilities that have been introduced with the release 2008a. The simulator is available from [10]. The project first developed a link level simulator before extending it to a system level simulator. While link-level simulations allow for the research on issues such as Multiple Input Multiple Output (MIMO) gains, Adaptive Modulation and Coding (AMC) feedback, modeling of channel encoding and decoding, system level simulations is more dedicated to network-related issues such as scheduling, mobility handling, interference management or signals propagation [11]. In system level simulations, the physical layer is abstracted from link level results and used for evaluating network performance. The system level simulator follows the structure shown in Figure 5. Each network element is represented by a suitable MATLAB class object.

The network topology is created as follows: a specific region of interest (ROI) is generated and divided into transmission sites or cells, to each one an eNodeB is appended. Each eNodeB contains a scheduler (see Figure 5). After creating the topology, one or many User Equipments (UEs) are deployed according to a specific spatial distribution model (only a constant number of users per cell with random positions is supported). Users can be either fixed or mobile (with a specific mobility model). Once the network created, the Vienna simulation main loop is carried out for each Transmission Time Interval (TTI) of 1ms: Depending on the scheduling algorithm and the received UEs feedbacks the scheduler assigns Resource Blocks (RBs) and a MCS to each UE attached to an eNodeB. At the UE side, the Signal-to-Interference and Noise Ratio (SINR) is calculated per received subcarrier. At the end of each time-step, link performance measurements are performed to provide the simulation output which consists of traces, containing link throughput and error ratios for each user, as well as summary measures by cell, from which statistical distributions of throughputs and errors can be extracted.

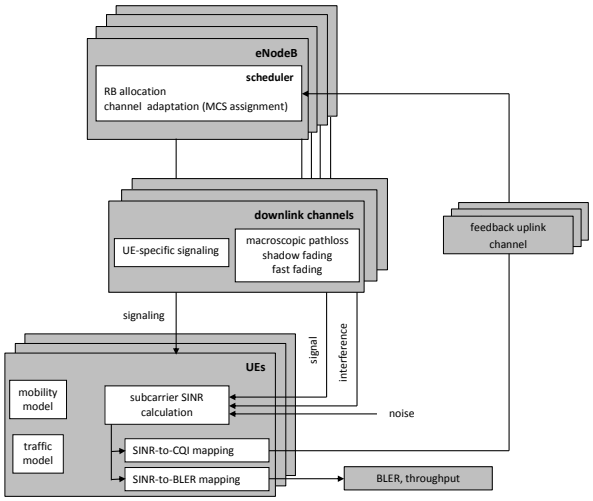

Figure 5: Block diagram of the LTE system level simulator [11]

The simulator contains a main MATLAB file *LTE_sim_main.m* which executes the pseudo-code below:

create the network
for each simulated TTI **do**
if mobile UEs **then**
move UEs
if UE outside ROI **then**
reallocate UE randomly in ROI
for each eNodeB **do**
receive UE feedback after a given feedback delay
schedule users
for each UE **do**
1- channel state → link quality measurements → SINR
2- SINR, MCS → link performance measurements → BLER
3- send UE feedback

For running the simulation, one has to write a MATLAB script which must perform the following tasks:

- Either loading a configuration file (in the +simulation_config subfolder) which applies a specific pre-configured simulation parameters, or configuring the parameters manually.
- Executing the main simulation file.

3.2 NS-3 Simulator

Network Simulator 3 (ns-3) is a discrete-event network simulator, developed as a simulation environment for networking research. It counts several modules, designed to simulate different technologies and scenarios. Here we work with the LTE module developed by the LENA Project [12]. This module was designed following the 3GPP LTE standard specifications and is included in the main distribution of the ns-3 latest versions.

NS-3 was developed using the C++ programming language and is a free simulator under GPLv2 License. It can be downloaded from the ns-3 webpage [13], where a manual and tutorials can also be found. We used for this comparison the ns-3.20 version. Figure 6 shows the hierarchy of the ns-3 classes.

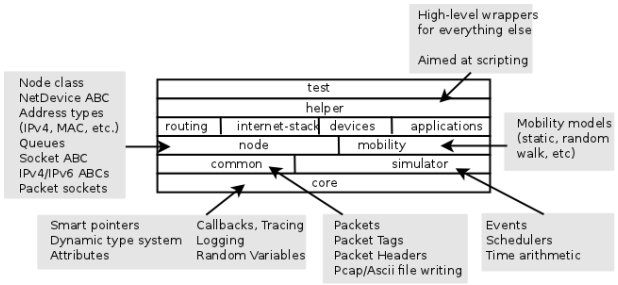

Figure 6: Software organization of the ns-3 simulator [13]

In ns-3 one has to write a script in C++ (or Python) with the include files, the topology of the network, the routing, the tracing and the applications. This script is run using a waf command: `waf --run myProgram --command-template="%s --RngRun=2"`, where "RngRun" gives the simulation run number. In order to compute statistics using independent runs one has to set a different number for each run.

4. SIMULATOR COMPARISON

In this section, we compare the easiness to simulate on both tools the four scenarios described in Section 2, and analyze their accuracy in terms of run-time duration and memory and CPU usage.

4.1 Generalities

As detailed in Section 2, for comparing the two simulators we defined four simple scenarios[1]: (DI) data traffic with infinite source, (DF) data traffic with finite source, (VI) voice traffic with infinite source, and (VF) voice traffic with finite source. The comparison is essentially carried out with respect to accuracy. However, for the most loaded scenario we also compare memory usage, computation time and CPU utilization. We also want to give a glimpse on their ease of handling, the realism of modeling assumptions and the ease of obtaining performance from traces of simulation.

The main differences between these two simulators are:

- NS-3 is a general network simulator that can be used to simulate many protocols and technologies, not only LTE. Instead Vienna is a specialized LTE simulator.
- Vienna advances the time by time-slices (of 1ms), while ns-3 does by the date of the next event: ns-3 is a discrete event simulator.
- Vienna is very precise on the modeling of the physical layer. At each TTI it performs many calculations at the physical layer level. These computations drastically slowdown the simulation which may make unfeasible the simulation of more realistic scenarios. On the other hand,ns-3 uses a correspondence table provided by the 3GPP LTE specifications, that directly links a certain MCS, the number of RBs the antennas are using and the transport block (TB) size [14]. Therefore, at each transmission, the simulator just retrieves such information, instead of calculating them, which results in an substantial gain.
- The code organization in ns-3 is very clear: each technology has a separate module and each class corresponds to a network real object. In Vienna modularity is not well exploited and it is not easy to create new classes without modifying many other files.
- Traces provided by Vienna are simple to treat and one can also easily add other type of traces if needed. In ns-3 traces collection and processing is cumbersome.

In order to perform the simulation of our four basic scenarios, we had to overcome some intrinsic limitations of the two simulators:

- Due to its lack of modularity, it is very complicated to make any code modifications in Vienna. The solution we found to improve the speed of the simulator was to perform the physical layer calculations only when an UE perceives a change in channel conditions, instead of doing them at each TTI.
- Vienna requires a constant number of users deployed per cell. For the DI and VI scenarios, where the total number of UEs in the cell varies with time, we had to make changes in the code of the simulator in order to allow the creation of new users on-the-fly. From the ns-3 side, the simulator limits to 320 the number of nodes to be created and connected to an eNodeB. We thus have to re-use nodes after a certain time of inactivity for overcoming such limitation.
- To simulate the DF scenario, we had to implement the ON-OFF traffic as a new application in both simulators. Indeed, such a traffic with an ON period characterized by a downloaded size (instead of a time) was not part of existing modules.
- For the voice cases (VF and VI) we had to implement a new scheduler policy in order to take into account rejection.

In Vienna, the mean number of active UEs can be easily measured using the traces from *traffic_model_type* where the state (ON or OFF) of each UE is stored at each TTI. The average throughput of a user can be obtained by dividing the average downloaded size (extracted from the *N_used_bits* output), by the mean ON period. The call rejection probability is calculated as the number of rejections divided by the total number of call demands. We added the *blocked_ids* vector as attribute of the scheduler. It is updated at each

[1]all code are available upon demand to authors

time-step to store the IDs of rejected UEs if a rejection takes place. At each time-step event and following the *blocked_ids* values, the *blocked* parameter, which we have created as attribute of each UE, takes"true" if he is rejected and "false" in the other case. An output vector, *rejected*, with total TTIs as length has been added to the *UEs_traces.mat* output trace file. This vector stores for each UE all the values taken by the *blocked* attribute from which we can extract the number of times that an UE was rejected.

In ns-3, we used the trace files to derive the performance measures. Among the files, the *DLPdcpStats.txt* provides for each TTI the amount of data downloaded by the active users. We filtered that file to extract the mean ON and OFF periods of each user, their average throughput and the mean number of active users. Moreover, we added to the system output the ability of identifying all rejections of UE's, whenever they take place. By combining the number of times a rejection occur for an UE with the statistics obtained from the *DLPdcpStats.txt* file, we obtained the rejection probabilities of the system.

4.2 DI: data traffic with infinite source

For this scenario we implement the mechanisms described in Section 2.1. In our simulation the transmission ends when the corresponding user completes the download of an exponentially distributed amount of data with mean x_{on}. We assume that new data transfers arrive according to a Poisson process, meaning that the time between two successive arrivals is exponentially distributed with mean $t_{off} = \frac{1}{\lambda}$.

As aforementioned, the Vienna simulator did not address this "open" case (where new UE demands arrive from the outside according to a Poisson process). We thus modified the simulator in order to be able to add, remove and reuse UEs during the simulation.

In order to simulate this scenario with ns-3, we came across a problem: the number of nodes (UEs that request service) that can be created is limited to 320. This limitation is due to LTE specifications. Therefore, instead of creating a node for each new arrival, we reused those already created and served. But this forced us to create a mechanism for differentiating the reused nodes in the traces.

In both simulators we consider a Single Input Single Output (SISO) single cell of diameter 500 meters. 10 UEs are static in the cell and make data transfers with mean parameters $x_{on} = 1.7$Mbits and t_{off} (the inter-arrivals time) mean distribution varied from 100ms to 150ms. We have configured the system bandwidth to 5MHz which is equivalent to 25 RBs. For the link quality measurements we used a simple channel model with only path-loss supported. The transmission power has been set to $1watts$ which allow all users to give the same and best MCS anywhere in the cell. With 25 RBs and for the best given MCS, we obtain a cell capacity C of 18336 bits/ms (according to the technical specifications described in [14]). Each simulation was executed 300 seconds.

For this scenario the simulated parameters are in Table 1. Each simulation has been executed 10 times and 95% confidence intervals were computed.

System frequency	2.6GHz
System bandwidth	5MHz (25 RBs)
Channel model	Only pathloss supported with the Free space model
Diameter of the cell	500m
Transmission power	1W (best MCS in the cell)
Transmission mode	SISO
Scheduler	Round Robin
Mean size x_{on}	1.7Mbits
Mean inter-arrival time t_{off}	from 100ms to 150ms
Simulation time	300s

Table 1: Simulation parameters of the DI model

Figure 7 depicts a performance comparison on the accuracy between the two simulators and the theoretical model in terms of the mean number of active users (eq. (2)) and the average throughput (eq. (4)).

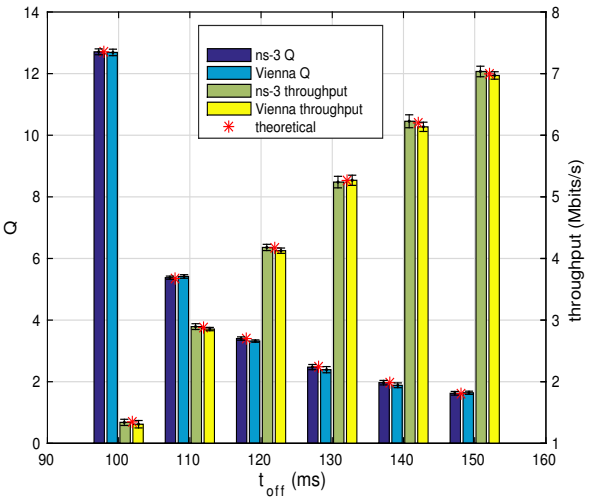

Figure 7: Average number of active users (Q) and mean throughput for the DI scenario.

One can see from this figure that both simulators are accurate. But ns-3 seems more accurate with a relative error (E_r) less than $0,5\%$ (i.e., $E_r < 0,005$) while Vienna shows a E_r inferior than $3,9\%$ for the mean number of active users. For the mean throughput ns-3 still having the same relative error and Vienna has a E_r less than $2,5\%$. Note however that in all cases the theoretical values are covered by the confidence intervals.

4.3 DF: data traffic with finite source

The theoretical model used to compare this scenario is describe in Section 2.2. We follow the same procedure of the DI model for implementing this scenario in Vienna. In this scenario, however, the number of users is constant and their behavior is modeled as an ON-OFF process. Each user alternates between an active period (ON state), during which traffic is generated, and an idle period (OFF state) of inactivity.

The two parameters that change with respect to the DI scenario are depicted in Table 2.

Number of users in the cell	10
Mean OFF period t_{off}	from 200ms to 1200ms

Table 2: Simulation parameters of the DF model

Figure 8 compares the average number of active users (eq. (6)) as well as the mean throughput (eq. (9)) for a varying t_{off}.

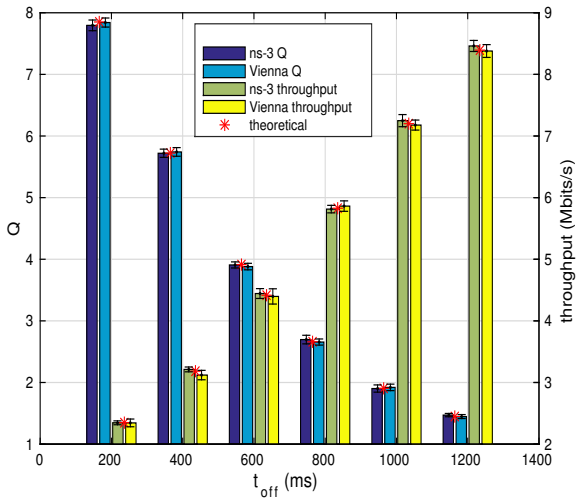

Figure 8: Average number of active users (Q) and mean throughput for the DF scenario.

Results reveal again no significant errors between simulation and analysis. In this case, Vienna provides a better accuracy on parameter Q than ns-3, with a E_r of 0.7% instead of 1.9%. Concerning the mean throughput, the E_r provided by ns-3 is of almost 1% while Vienna has a E_r of around 1.9%. Moreover, we observe that the 95% confidence intervals are quite narrow and include the true values in all cases for both simulators.

4.4 VI: voice traffic with infinite source

As explained in Section 2, the main difference between data traffic and voice traffic resides in the fact that ON periods are not characterized anymore by a size (of downloaded elements), but instead by a time (of conversations). We thus characterize the activity session by an exponential distribution of mean t_{on}.

As in this scenario the number of sources is infinite, we reuse the procedure for creating users, as explained in the DI scenario. We assume that new call demands arrive according to a Poisson process, meaning that the time between two successive arrivals is exponentially distributed with mean $t_{off} = \frac{1}{\lambda}$.

For the VI scenario we simulate a single cell with the parameters listed in Table 3. We are aware that t_{on} and t_{off} of the order of hundred of milliseconds do not correspond to reality. But based on theoretical results, we know that the performance parameters of the system is only sensitive to the ratio of t_{on} and t_{off}. Consequently we chose small values in order to significantly reduce the simulation run-times. This is even more important in the case of Vienna simulator that runs at a TTI time-step.

Diameter of the cell	500m
Scheduler	Round Robin (modified to reject calls)
Number of servers	10
Mean ON period t_{on}	1000ms
Mean OFF period t_{off}	from 100ms to 150ms
Simulation time	300s

Table 3: Simulation parameters of the VI model

Figure 9 compares simulation results with analytical values, for both performance parameters of interest, the average number of active users Q (eq. (12)) and the rejection probability P_r (eq. (13)). It shows that ns-3 yields little more precise estimation than Vienna. The relative error on Q is less than $0,38\%$ for ns-3 and less than $0,59\%$ for Vienna. Concerning P_r errors are inferior to $0,36\%$ and $4,86\%$ for ns-3 and Vienna respectively. Despite this difference, we can say that both simulators produce comparable results, as in both cases the 95% confidence intervals always contain the actual values and no significant deviation between the exact and the simulated statistics is observed.

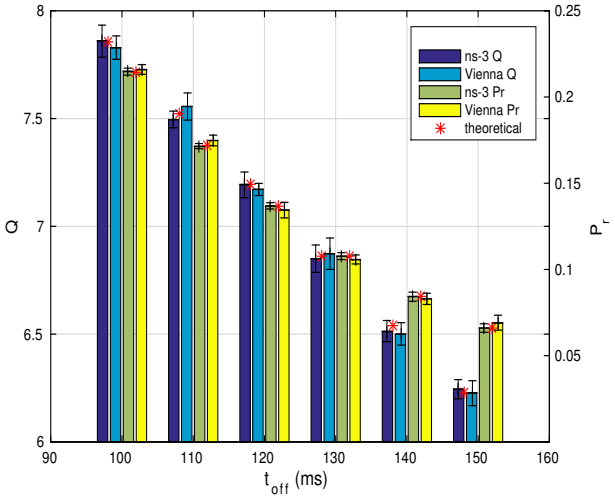

Figure 9: Average number of active users (Q) and rejecting calls probability (P_r) for the VI scenario.

4.5 VF voice traffic with finite source

For this scenario we configured our two simulation scripts to model a single-cell system with a constant number $N = 15$ of users that perform an ON-OFF traffic with a $t_{on} = 950ms$ for the exponential distribution. We vary t_{off} from $200ms$ to $1200ms$. The number of servers S was first set to 10, corresponding to the case where $N > S$, and secondly to 20, corresponding to the case where $N \leq S$. As before, each simulation was executed for 300 seconds in a cell area of

diameter 500 meters. The parameters are summarized in Table 4.

Diameter of the cell	500m
Scheduler	Round Robin (modified to reject calls)
Number of servers	10,20
Number of users	15
Mean ON period t_{on}	950ms
Mean OFF period t_{off}	from 200ms to 1200ms
Simulation time	300s

Table 4: Simulation parameters of the VF model

The average number of active users Q (eq. (17)) as well as the probability of rejecting a call P_r (eq. (18)) are compared in Figure 10 for the case where $N > S$. Figure 11 only gives Q in the case where the $N \leq S$ (as in this case the rejection probability is null, see eq. (15)). The 95% confidence intervals estimated from a sample of 10 replications are indicated for each value.

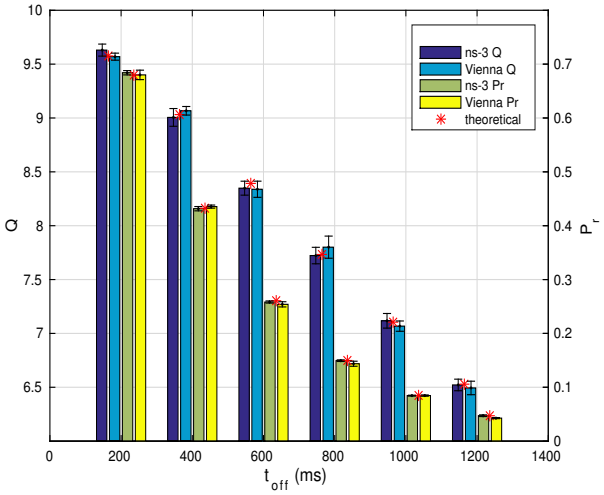

Figure 10: Average number of active users (Q) and rejecting calls probability (P_r) for the $N > S$ VF.

Again, the results of both simulators match exact results with a pretty good accuracy. In the first case ($N > S$), relative errors for Q are smaller than $0,59\%$ for ns-3 and less than $0,83\%$ for Vienna. Concerning P_r, the difference remains small for ns-3, less than $0,58\%$, but becomes bigger for Vienna, around $9,5\%$. In the second case ($N \leq S$), the relative errors on Q remain inferior to $0,9\%$ for ns-3 and to $0,64\%$ for Vienna. In this case, Vienna is thus slightly more accurate than ns-3.

4.6 Performance comparison

In order to compare the simulation run-time, memory usage and CPU occupation we conducted simulations on a 8 cores Intel(R) Xeon(R) CPU E5-2450 v2 @ 2.50GHz workstation with 64GB of RAM, running Ubuntu Linux 14.04 LTS.

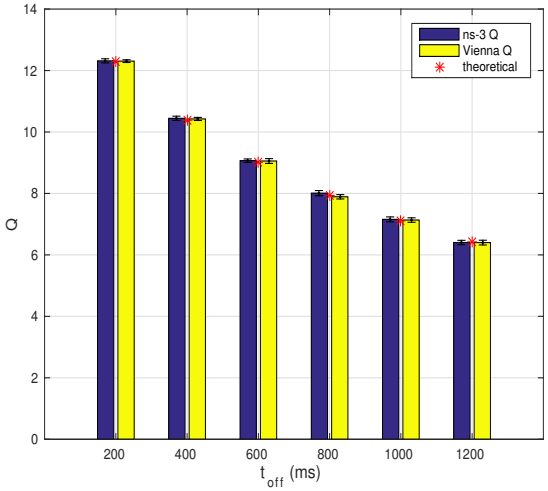

Figure 11: Average number of active users (Q) for the $N \leq S$ VF.

Tables 5, 6 and 7 show the different measured performance metrics for each simulator. These results correspond to the simulation of the most loaded scenario.

In both ns-3 and Vienna, the run-time performance is directly provided as an output of our scripts. For the measurement of the memory and CPU occupation performance, we used the ***top*** program of Linux which provides a dynamic real-time view of the system: when a simulation is running, we periodically observed the used amount of memory and the CPU utilization and the final results were averaged over the number of observations.

model	VF	DF	VI	DI
ns-3	37min	27min	465min	608min
Vienna	780min	390min	270min	300min

Table 5: Comparison of the run-time

The performance comparison discloses large differences between the two simulators. In the finite source scenarios *ns*-3 has proven to be largely faster than Vienna. The first needs around 37 minutes to simulate the VF scenario and 27 minutes for the DF model, while Vienna takes 780 and 390 minutes to execute these simulations. We conclude that *ns*-3 is advantageous and capable of carrying out long simulations of large-scale networks in an efficient way. Note that the VF scenario needs higher run-times than the DF ones for both simulators because the first model is more loaded and use a larger-scale network (15 UEs vs 10 UEs in DF). We observe for Vienna a big difference of almost 400 minutes between these two scenarios. We explain that by its behavior of advancing the time by time-step of 1ms and performing a lot of calculations for the PHY layer at each event. Therefore, executing these loops with five UEs more at each TTI makes an important difference. Moreover, the fact that we have more active UEs in the VF case (around 12 vs 8 in the DF case as shown in figures 12 and 9) leads to higher computational load. In the infinite source scenarios *ns*-3 need higher

computation time than Vienna. This difference is due to the fact that the two simulators use different networks with different scales to carry out infinite source simulations. Indeed, *ns*-3 models the infinite source by a large number of UEs per cell (320 nodes) while in Vienna we can create and eliminate nodes at runtime.

model	VF	DF	VI	DI
ns-3	52MB	44MB	71MB	64MB
Vienna	1.6GB	1.25GB	2.25GB	3GB

Table 6: Comparison of the memory usage

model	VF	DF	VI	DI
ns-3	3.05%	3.05%	3.1%	3.1%
Vienna	8.1%	8%	7.5%	7.6%

Table 7: Comparison of the CPU occupation

Regarding the memory we can clearly see the superiority of ns-3 over Vienna which uses as much as 46 times more memory. We saw a big difference in memory between infinite and finite source scenarios.

For CPU utilization, ns-3 uses less resources than Vienna but the difference is not as high as for memory.

5. CONCLUSIONS

We have compared two simulators, ns-3 and Vienna, in the context of LTE networks, on four basic scenarios for which well-known analytical results exist. We evaluated both of them in terms of accuracy, ease of handling and use, execution time, memory and CPU usage. Our conclusions are the following:

First, in terms of accuracy, both simulators provide good results. Both give small errors when compared to theoretical values and narrow confidence intervals for the four scenarios we tested.

Second, regarding the execution time, there are important differences between simulators. For the scenarios with a finite number of sources, ns-3 is clearly faster than Vienna. However, Vienna becomes faster for infinite source scenarios. This can be explained by the fact that in ns-3 we need to create at the beginning of the simulation a fixed and sufficiently big number of UEs, whereas in Vienna, UEs can be created and deleted on-the-fly.

Third, in terms of CPU and memory usage, ns-3 is much better than Vienna. While ns-3 is based on C++, Vienna is based on the MATLAB environment, which requires enormous amounts of memory to run.

Fourth, concerning the ease of handling and use, the choice is somehow subjective, it depends on the skills of the programmer, in C++ or in MATLAB language. However, we find that output traces are easier to manipulate in Vienna than in ns-3. In the other hand, ns-3 is quite well documented which is not the case for Vienna.

Finally, taking into account the four aforementioned points and the experience acquired with this simulation comparison, we believe that ns-3 is a better choice for simulating more complex scenarios, including different coding zones within the cell, different cells, inter-cell traffic and UE's mobility. These more complex scenarios correspond to the recent work we are carrying out in the framework of the IDEFIX project.

Acknowledgment

This work has been carried out in the framework of IDEFIX project, funded by the ANR under the contract number ANR-13-INFR-0006.

6. REFERENCES

[1] E. Dahlman, S. Parkvall, J. Skold, and P. Beming. *3G Evolution: HSDPA and LTE for Mobile Broadband*. Academic Press, 2th edition, 2007.

[2] Intelligent design of future mobile internet for enhanced experience. http://lia.univ-avignon.fr/idefix/.

[3] E. Weingartner, H. vom Lehn, and K. Wehrle. A performance comparison of recent network simulators. In *IEEE International Conference on Communications (ICC)*, pages 1–5, June 2009.

[4] A.R. Khan, S.M. Bilal, and M. Othman. A performance comparison of open source network simulators for wireless networks. In *IEEE International Conference on Control System, Computing and Engineering (ICCSCE)*, pages 34–38, Nov 2012.

[5] M. H. Kabir, S. Islam, Md. J. Hossain, and S. Hossain. Detail comparison of network simulators. *International Journal of Scientific & Engineering Research*, 5:203–218, october 2014.

[6] E. Schoch, M. Feiri, F. Kargl, and M. Weber. Simulation of ad hoc networks: ns-2 compared to JiST/SWANS. In *ICST International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops (SIMUTools)*, pages 36–43, march 2008.

[7] G. F. Lucio, M. Paredes-Farrera, E. Jammeh, M. Fleury, and M. J. Reed. Opnet modeler and ns-2: Comparing the accuracy of network simulators for packet-level analysis using a network testbed. In *WEAS International Conference on Simulation, Modelling and Optimization (ICOSMO)*, pages 700–707, october 2003.

[8] E. Altman and T. Jimenez. *NS Simulator for Beginners*. Synthesis Lectures on Communication Networks. Morgan, 2012.

[9] L. Kleinrock. *Queueing Systems, Volume 1: Theory*. Wiley-Interscience, 1975.

[10] LTE simulators. http://www.nt.tuwien.ac.at/research/mobile-communications/vienna-lte-a-simulators/.

[11] J. C. Ikuno, M. Wrulich, and M. Rupp. System level simulation of LTE networks. In *IEEE Vehicular Technology Conference (VTC)*, pages 1–5, May 2010.

[12] The LENA Project. http://networks.cttc.es/mobile-networks/software-tools/ns-3/.

[13] NS3 network simulator. http://www.nsnam.org/.

[14] European Telecommunications Standards Institute. LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Physical layer procedures (3GPP TS 36.213 version 8.8.0 Release 8), 2009.