Analysis and Experimentation with a Realistic Traffic Generation Tool for Emerging Application Scenarios

Aymen Hafsaoui
EURECOM, Mobile Communications Department,
06410 Biot, France
hafsaoui@eurecom.fr

Navid Nikaein
EURECOM, Mobile Communications Department,
06410 Biot, France
nikaeinn@eurecom.fr

Christian Bonnet
EURECOM, Mobile Communications Department,
06410 Biot, France
bonnet@eurecom.fr

ABSTRACT

With the emergence of new networks and available applications, emulation of computer networks is an efficient technique for evaluating the performance, transport reliability and application-level protocols. Traffic generation is one of the key challenges in modeling and simulating the application and network load.

In this work, we present a tool, called OpenAirInterface Traffic Generator (OTG), for the generation of realistic application traffic that can be used for testing and evaluating the performance of emerging networking architectures. In addition to the traffic of conventional applications, OTG is capable of accurately emulating the traffic of new application scenarios such as machine-type communication (MTC) and online gaming. We aim in this work to present OTG architecture and through experimentations to present main novelties and features of the presented tool. To highlight the capability and new features of the tool, the one-way delay of machine-type communication traffic and the case of M2M aggregated traffic is analyzed over the LTE network using OpenAirInterface in-lab system validation platform. For our delays analysis we will focus on two scenarios with bidirectional and aggregated traffics.

Categories and Subject Descriptors

I.6.6 [**Performance Analysis and Design Aids**]: Simulation and Modeling—*Simulation Output Analysis*

General Terms

Performance, Measurement.

Keywords

Network simulation, Performance Analysis, Machine-type communication, Data aggregations, LTE.

1. INTRODUCTION

Over the last decade, the heterogeneity of the Internet is constantly increasing, with new access technologies, new client devices and with more and more services and applications.

High-performance online gaming and machine-to-machine (M2M) are two examples of emerging massive applications for next generation networks. Both applications are expected to create an increasing number of connected devices over the following years and to be an integral part of the traffic transported by the network [11]. The market for M2M application will is continuously in growth for the upcoming years. In fact, this traffic has been with us for now more than 65 years, the setup was only in specialized networks build in expert systems. But now, it becomes a public market with traffic routed though the internet between the node and the server. Support for such applications with a massive number of connected devices and their heterogeneous traffic have deep implications on the end-to-end network architecture [4, 10]. Consequently, understanding and modeling the traffic of such applications are a key for designing and optimizing a network and the applicable QoS scheme capable of providing adequate communication services without necessarily compromising the conventional services such as data, voice, and video. This is critical as the current networks are primarily designed and optimized for a continuous flow of information, at least in terms of the time-scales needed to send several IP packets, and mostly from the server to the client, while the traffic of emerging applications are considerably sporadic (not continuous) with low-throughput packet arrivals and mostly originated from the client to the server [2].

In the present paper, we extend the previous work on a *realistic* packet-level traffic generator tool, called OpenAirInterface traffic generator (OTG) [5]. The main contribution is that in addition to conventional traffics, OTG also takes into account the intrinsic traffic characteristics of the emerging applications such as MTC and online gaming. In particular, it implements a state-aware multi-source traffic models suitable for the majority of the M2M application scenarios such as sensor-based alarm or event detection as well as models for the first-person shooter (e.g. OpenArena), racing (e.g. kart rider), and background traffic applicable to the online gaming application scenarios. The modeling approach is derived from both the traffic specification and measurement for the considered applications[2, 1].

Different from the existing traffic generator [3, 13, 12, 14], OTG captures the specific characteristics of the emerging M2M and online gaming application scenarios and provides models for the background traffic (e.g. file download, web, email, update). Thus it is capable of generating mixed human and machine type traffic patterns. Furthermore, it has a dual operation modes: soft realtime and hard realtime, based on RTAI under Linux (i.e. LXRT module) [16], to meet application and/or protocol timing constraint.

EMUTools 2013 March 5, Cannes, France.

The reminder of this paper is organized as follows. In Section 2, we present the main idea and the architecture of the OTG. In Section 3, we provides further details about the implemented MTC traffic models as well as the background traffic. The measurement and analysis of the end-to-end one-way delay (OWD) for two MTC applications, namely virtual bicycle race for bidirectional traffic and and sensor-based alarm and event-detection for the aggregated traffic are presented in Section 4. Finally, we summarize and conclude this work.

2. OPENAIRINTERFACE TRAFFIC GENERATOR (OTG)

OTG is a realistic packet-level traffic generation tool for emerging application scenarios. It is developed in C under Linux and deployed for the OpenAirInterface LTE/LTE-A platform allowing the traffic to be generated with soft realtime and hard realtime constraint [15]. The main difference is about the timing: soft realtime operation is designed to respect the timing on average making this mode of operation more suitable for large scale experiment, while realtime operation is designed to respect the timing strictly as would be in a real application. If OTG is attached directly to user-plane protocols, it is capable of reproducing the packet headers as in a real networking protocol stack according to the user-defined configuration. Both transmitter and receiver traffic statistics are generated and analyzed to derive the various measurements on the application-specific key performance indicators (i.e. throughput, goodput, loss-rate, latency). In OTG, the traffic generation is defined by five *ordered processes* as described below:

- **State:** handling randomly distributed sojourn time at each state and probabilistic traffic state transitions (e.g. On-Off traffic models). Note that no traffic is generated within the idle/off state and during the state transition.
- **Inter-departure time (IDT):** determining the time between the transmission of two successive packets.
- **Packet size (PS):** generating the amount of payload being transferred by the packet.
- **Aggregation:** combining traffic of multiple sources into a single packet for specific nodes such as gateways.
- **Background (BG):** generating the traffic of background applications such as file download, email, and syncs/updates, modeled by PS and IDT derived from [9].
- **Multiple flows management:** this offers the possibility to generate and manage simultaneously multiple flows (i.e. more than one stream between a pair of source and destination).

Both IDT and PS processes are modeled as i.i.d. series of variables following a user-defined distribution (e.g. constant, uniform, gaussian, exponential, poisson, weibull, pareto, gamma, cauchy and lognormal). OTG allows to reproduce exactly the same stochastic experiment by choosing the same seed values for IDT and PS random processes. In a given traffic state, the decision upon the transition to the next state is made when the sojourn time in that state expires. During the sojourn time, packet payloads are generated according to the application-specified traffic model (i.e. IDT and PS distributions). If the traffic aggregation is applicable, payloads of multiple nodes are combined together to generate a single packet (for the similar aggregated flows with a same transport protocol we use the same header, we duplicate header for the case of heterogeneous aggregated flows). The background traffic is a parallel process that generates packets from/to server in order to emulate the cell load. Note that OTG is reproducing the traffic of each single device, which in turn does not mean that any correlations between machines can be captured [8]. For example, assume hundreds of temperature sensors are spread over a small area, on which temperature is uniformly passing a threshold at a certain point of time. In that case all sensors would trigger simultaneously, causing strong correlation in network traffic. Such cases could be also captured by OTG if the state transition is controlled (e.g. with a predefined frequency) such that a group of devices are in the same state at a given time. The OTG high level architecture is shown in Figure 1 and composed of five main components as described in the following subsections.

- **Configuration:** sets up the OTG parameters using user-defined xml traffic descriptor or predefined realistic configuration templates.
- **Transmitter (TX):** builds packets according to OTG packet generation process with additional control information used for traffic statistics, and updates and log transmitter statistics.
- **Receiver (RX):** captures the packets and updates and log receiver statistics.
- **Log generation (Log Gen):** collects and formats OTG TX and RX statistics for each traffic flow.
- **Statistics and KPIs (Stats/KPIs):** analyzes and derives the measured statistical data sets for both data and background traffics, and computes key performance indicators (KPIs).

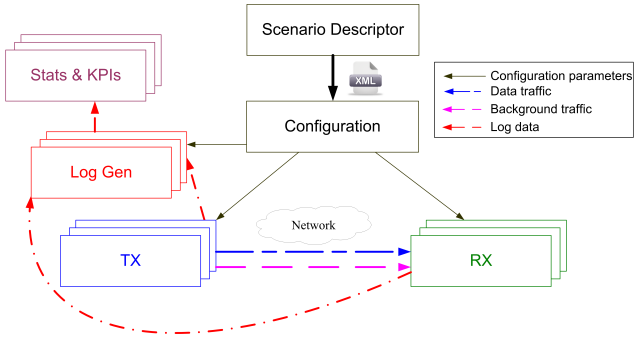

Figure 1: High-level architecture of OTG

2.1 Configuration

The first step of traffic generation consists of the setup of OTG configuration parameters, with the configuration module. One of the main novelty of our tool is that it offers a large flexibility in terms of usage (adapted for itinerant and expert users). Through a web interface and XML files, users can select predefined scenarios (several realistic models for emerging applications are available) or to build their own simulation parameters for the desired architecture and traffic. As configuration file is composed of four components: environment/system configuration, Topology configuration, application configuration and emulation configuration. We focus below in the Application configuration which allows to define the main characteristics of the traffic to generate, in upper layers.

To configure the traffic to simulate user can select preconfigured or customized traffics. For the preconfigured traffic user identify flows source/destination and select the traffic to generate. While, for the customized configuration, users have the possibility to manually set-up data packet header size, IDT and packet size distributions. To model and M2M traffic, our tool offers several realistic template with realistic MTC and sensors scenarios such Auto Pilot, Virtual Game and sensors for alarms. One of the main novelties of OTG is that it allows to generate traffic with several states for each flows. Our traffic models include multiple traffic states where the transition dynamics are probabilistic and based on timing. Our implementation for the M2M state machine incorporate traditional deterministic characteristics of automated mechanisms and random characteristics of the network, application characteristics and constraints. For a convenient modeling of MTC traffic, the implemented algorithm uses the Hidden Semi Markov Models (HSMM). It takes into account M2M traffic parameters, it will presented next in section 3.1.

2.2 Transmitter (TX)

To transmit data OTG offers two methods: the first one is at the network layers using standard Linux sockets APIs and the second one is below IP, based on Linux IPC or RPC. For the Above IP (socket mode): only the packet payload is generated by OTG and packet header and the actual transmission and reception is managed by the Linux protocol stack based on socket configuration (IP version, transport protocol, destination address and port). In the Below IP (IPC/RPC), the packet payload and header generations as well as the OTC-specific control information including a application identifier, aggregation level, sequence number, time, header size and payload size are managed by OTG.

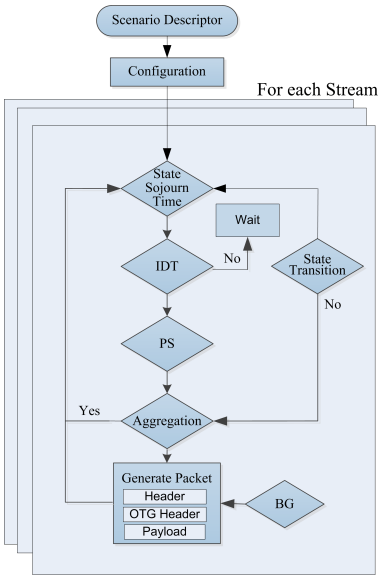

Figure 2: OTG Transmitter

The packet generation process is performed as follows. As shown in Figure 2, multiple flows can be handled simultaneously for each pair of transmitter and receiver. If the traffic aggregation is enabled, traffics originating from multiple source may be aggregated. Each flow is described by the packet inter-departure process and the packet size process. Both processes are user-defined and modeled as independent and identically distributed series of random variables. OTG currently implements the following distributions: uniform, normal, gaussian, exponential, weibull, poisson, pareto, on-off, on-off-active.

2.3 Receiver (RX)

After receiving the data packet two cases are possible. For the above-IP mode, for each correctly received packet, an acknowledgement is sent back to the transmitter. For the below-IP mode as depicted in Figure 3, the receiver performs the following operations: (i) extracts data packet size from each received payload and checks if it corresponds to the same size received from the PDCP layer (ii) selects from the header packet type (data or background) and application ID (since we have multiple traffics per pair) (iii) compares the sequence number from the payload with the expected one to detect losses and out of sequence packets (iv) updates log files with measurements and simulation information.

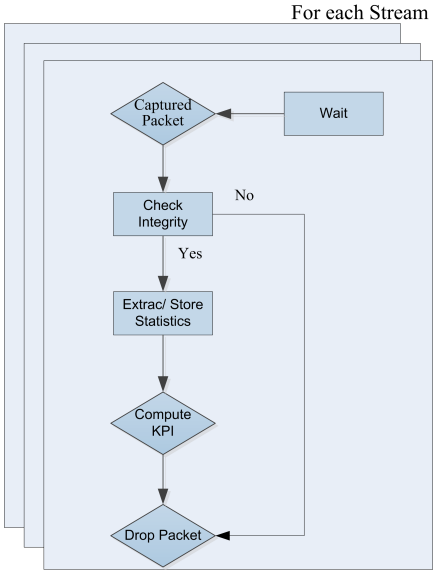

Figure 3: OTG Receiver

2.4 Log generation, Statistics and KPIs

Log generation collects and processes each transmitted and received packets in the experiment in order to derive and format the statistics for each traffic flow. The specific statistics are defined during the scenario configuration allowing log generation to compute relevant KPIs in realtime accessible to the user. Figure 4 shows main inputs and outputs of Stats/KPIs module.

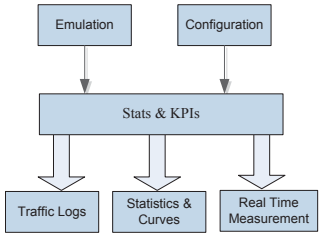

Figure 4: OTG KPI and Measurements Analysis Process

For performance analysis task, OTG offers several levels of investigations. In fact, user can follow in real time measurement results

(latency, throughput and losses) for the uplink and downlink. Also latencies and instantaneous throughput were recorded in log files. After the end of the simulation, an automated report with statistics and curves for the KPIs are generated.

3. SUPPORT FOR EMERGING APPLICATION SCENARIOS

The OTG mainly targets two emerging applications: machine-to-machine communication and highly interactive online gaming [2]. For the M2M traffic, it implements both the predefined models for the auto-pilot, virtual game, sensor-based alarm and event detection, and team-tracking as well as user-defined fully customized state-aware traffic model. For the gaming traffic, it implements the OpenArena, TeamForteres and Dirt 2 as well as the background traffic (i.e. email, web browsing, data transfer) derived from [2]. In the following subsection, we provide further details about the supported states for the M2M traffic and present the model for the background traffic.

3.1 M2M Traffic Model

Nowadays, mobile networks are dimensioned using standard mobile wireless network traffic models, which are based on the typical behaviour of human subscribers. It may be expressed in typical time spent using speech service, number of sent/received messages (SMS, MMS) and the amount of downloaded data. These traffic models do not take into account traffic generated by machines, thus new traffic models are required. For instance, in the case of meteorological alerts or monitoring of the stability of bridges, MTC devices will infrequently deliver a small amount of data. Another type of application is event detection requiring fast reaction time to prevent potential accidents; one example is the detection of pressure drop through the pipelines (gas/oil). Moreover, in the field of surveillance and security, the sensing devices send periodic reports to the control center until a critical event happens. Once the event is triggered, *event driven* data traffic is first sent by the sensor to a central control unit or other types of infrastructure. Subsequently more packets may be exchanged between parties to handle this event.

The Analysis of the functions of the majority of the applications has revealed that the MTC has three elementary traffic patterns [2]. We describe briefly below the main process and constraints to fulfil in order to switch from one state to an other. We present in Figure 5 the M2M traffic modeling framework implemented in OTG.

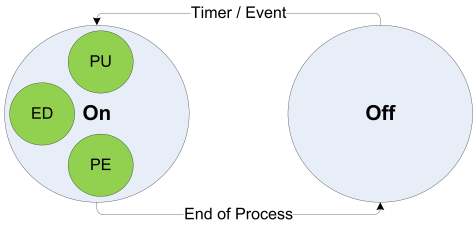

Figure 5: M2M Traffic Modeling Framework

- **OFF state:** It corresponds to the starting state, where application have no data to transmit. We call $P_{OFF/PU}$, $P_{OFF/ED}$ and $P_{OFF/PE}$ the probability to move from OFF state to respectively PU, ED and PE states. $T_{OFF/PU}$, $T_{OFF/ED}$ and $T_{OFF/PE}$ correspond to the holding/sojourn time to wait in OFF state before to move respectively to PU, ED and PE states. Each time where the application is in the OFF state, we generate a random probability value. Depending on this values and the application configuration, the system will switch to the next state, but before it waits for the corresponding waiting time.

- **PU state:** This type of traffic occurs if devices transmit status reports of updates to a central unit on a regular basis. It can be seen as an event triggered by the device at a regular interval. PU is a non-realtime and has a regular time pattern and a constant data size. The transmitting interval might be reconfigured by the server. A typical example of the PU message is smart meter reading (e.g. gas, electricity, water). We call $P_{PU/ED}$, $P_{PU/PE}$ the probability to move from PU state to respectively ED and PE states. Only one packet is sent, after that according to the generated probability and the application configuration the system will switch to the next state.

- **ED state:** In case an event is triggered by an MTC device and the corresponding data has to be transmitted, its traffic pattern conforms to this second class. An event may either be caused by a measurement parameter passing a certain threshold or be generated by the server to send commands to the device and control it remotely. ED is mainly a *realtime* traffic with a variable time pattern and data size in both uplink and downlink direction. An example of the realtime ED messages in the uplink is an alarm / health emergency notification and in the downlink is a Tsunami alert. In some cases, ED traffic is non-realtime, for example when a device sends a location update to the server or receives a configuration and firmware update from the server. We call $P_{ED/PU}$, $P_{ED/PE}$ the probability to move from ED state to respectively PU and PE states. Only one packet is sent, after that according to the generated probability and the application configuration the system will switch to the next state.

- **PE state:** This last type of data-traffic is issued after an event, namely following one of the previous traffic types (PU or ED). It comprises all cases where larger amount of data is exchanged between the sensing devices and a server. This traffic is more likely to be uplink-dominant and can either be of constant size as in the telemetry, or of variable size like a transmission of an image, or even of data streaming triggered by an alarm. This traffic may be real time or non-real time, depending on the sensor and the type of the event. We call $T_{PE/OFF}$ the holding/sojourn time in PE state. Once the sojourn time is exceeded, the application moves to the OFF state.

3.2 Background Traffic

To assess a realistic implementation for background traffic in LTE networks we implemented a background traffic generator based on a realistic measurements in HSPA and LTE networks proposed in [9]. We distinguished between uplink and downlink traffic. To model background data for uplink and downlink we used Lognormal distributions, where each traffic direction is characterized by different packet size.

4. EXPERIMENTATION

The experimentation is performed on the top of OpenAirInterface emulation platform [15], which is an integrated tool allowing large-scale networking experimentation applicable to both evolving cellular (i.e. LTE/LTE-A) and adhoc/mesh topologies. In the cellular

configuration, the platform mainly implements the radio access network (E-UTRAN) following the 3GPPP specifications. The hardware platform is a laptop equipped with a quad-core CPU running OAI emulator and protocol stack using Linux on Ubuntu 12.04. An overview of the experimentation setup is given in Figure 6. We carried out one-way delay (OWD) measurements in the soft realtime mode for LTE operating in TDD frame configuration 3 for 5MHz bandwidth. The rest of the network including M2M capillary .[1], mobile core network, IP backbone, and application server are emulated in terms of additional latency as the purpose of the experiment is to measure the end-to-end OWD in the data-plane. We make use of OAI scenario descriptor to layout the experiment such that the reproducibility is preserved and results can be regenerated.

4.1 Virtual Race

One example of the many possible MTC games is the virtual race (e.g. virtual bicycle race using real bicycles)[2]. The opponents are on different location. During the race they are periodically informed with sensor data on the order of 1KB. Which reflects a PU traffic with data packet size of 1KB. We have to notice that update frequency depends on opponents speeds. For the case of our simulation we considered the case of opponents with medium speed. In the case that the are more competitors or team competitions, application servers are required to process all competitors data. For the simulation we used a simple cellular network topology composed of one eNB (enhanced-NodeB) and one static UEs (User Equipment) to measure the best case performance. To measure the OWD, device synchronization is required. For the RAN, we used the time synchronization between eNB and UE in terms of frame and subframe number and convert it to time in milli-seconds (i.e. each frame represents 10 ms and each sub-frame represents 1ms). For mobile core network, we applied the latency measurement in [7], and for IP backbone and application server, we applied the latency estimation in [11]. The end-to-end network setup is emulated on the same physical machine, thus avoiding additional time-synchronization. The simulation is run for 1 minute (i.e. 6000 LTE TDD frames).

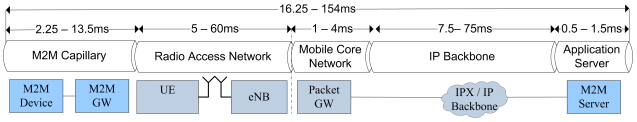

Figure 6: Experimentation setup and OWD latency budget

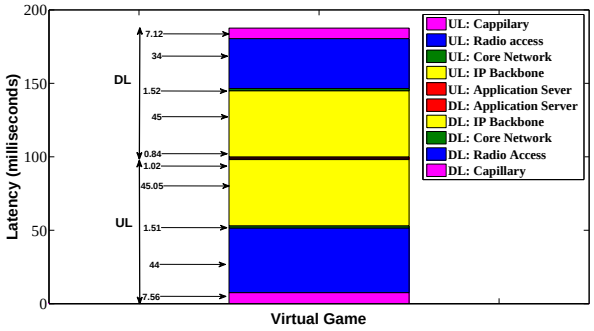

Figure 7: OWD Analysis for m2m Virtual Game Application

Figure 6 and 7 present the measured and estimated OWD performance for virtual game per network segment. It can be seen that the delay performance is better in downlink (from server) than in uplink (to server). The reason is that the LTE TDD frame configuration 3 has more downlink subframes (i.e. 6) than that of uplink (i.e. 3) indicating that the DL and UL subframe allocation should be balanced to meet the load conditions. Similar results have been reported in an LTE FDD frame format with unknown number of users [7]. Furthermore, we can observe that the delay of the radio access network is much larger than that the mobile core network, which calls for further optimization of cell configuration, uplink channel access method and scheduling especially when the number of users increases. When comparing with the IP backbone (Internet), we note that this segment has a large delay in both directions, which depends on the region, the number of nodes in the network, and their processing delays [11]. The IP backbone represents a high delay variation, which can be improved by providing service locally closer to the client (e.g. within the mobile packet gateway). From the results, we observed for virtual race that is characterized by a large constant sized packets with random constant times in uplink (it depends on players speeds), and constant packets with constant arrival time in the downlink. The main observation is that uplink still characterised by highest delays, which can impact the quality of experience of each player. Since, for example for "First-person shooter" the survival of each player depends on his reactivity time, which is strongly correlated with the network delay. Finally, we can conclude that we obtained acceptable OWD results with non real-time application and operators still have to improve radio access delays.

4.2 Aggregated Traffic

The aggregate traffic performance represents an important metric for Internet Service Providers (ISP). ISPs are interested in maximizing the goodput perceived by end users and minimizing the delay of the traffic traversing their networks. There is no such evaluation available for sensors traffics in LTE networks. For instance, we focus our interest on several heterogeneous sensor traffics crossing a gateway.

Sensor	Refrigerator	Clothes Washer	Clothes Dryer	Dishwasher	Stoves/ Ovens
Updating Period (hour)	1	24	24	24	24
Payload Size (bytes)	30	8	8	8	8

Table 1: M2M Traffic Parameters

The experimentation is done based on 5 realistic sensor traffic sources [6]. In order to reduce daily and weekly effects, the measurements are carried out for seven days. Each sensor traffic corresponds to an M2M uplink traffic with two states (OFF and PU). Periodically each sensor sends a small data via the LTE network to a remote server. Table 1 summarize the traffic characteristics for different type of sensors. We observe that for the refrigerator we have a PU updating time of one hour, and for the other sensors the PU updating time is one days.

The OWD for each IP packet was calculated by comparing the timestamps for each packet in OTG header and the receiver timestamps. While packet sizes were derived from OTG header. We presented respectively in Figure 8 and Figure 9 transmitted packets size and the corresponding OWD for a simulation time of 7 days. From Figure 8 we observed that for each end of the day we have peaks of data packet with 125 bytes, while the size of the rest packets is 93 bytes. Peaks corresponds to the aggregated packet (sent on the same time corresponding to their PU updating time)

[1] An M2M system consists of M2M devices connected to an UE in the evolved UTRAN, either directly or via M2M gateways (M2M GW)

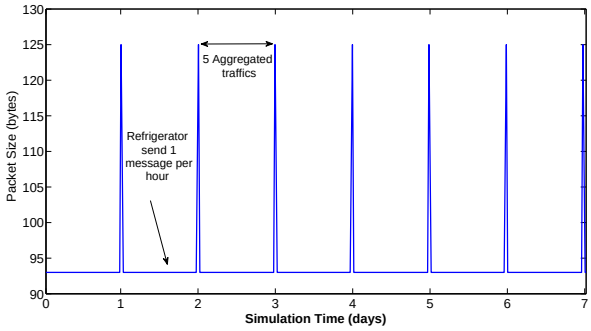

Figure 8: Packet Size

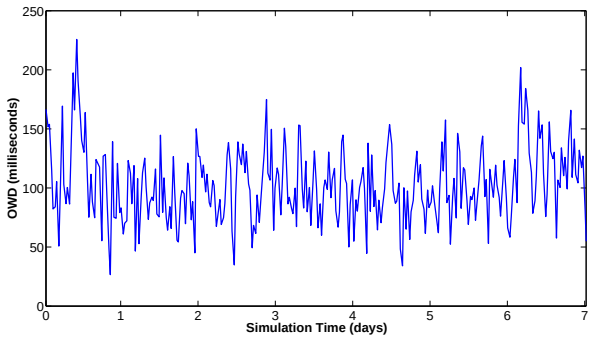

Figure 9: One Way Delay

from the 5 sensors. We have to mention that for the aggregation, if data to aggregate have a same header characteristics, the size of the generated packet is the sum of the header with the totals payload sizes. For the case of the present scenario, we have a header size of 63 bytes, then the obtained peaks corresponds to 63 + 30 + (5 * 8), due to multi-source flows aggregation. If we focus on a time slice of one day we observed that refrigerator sensor send 24 packets of 93 bytes, which is in line with PU updating time in Table 1. Which highlight the ability of OTG to manage state-full flows with multi-source flows and to aggregate data when it is necessary. We plot in Figure 9 the OWD evolution during the simulation time. We notice that OWD delays are not correlated neither with simulation time nor with the packet size (correlation factor less that 0.03). The OWD measurements show higher delay than the expected values presented in Figure 6, which may induce a negative impact in view of massive deployment of such sensors in a city.

5. CONCLUSION

Accurate modeling and generation of realistic application traffic are difficult and challenging tasks for emerging application scenarios. In this work, we briefly discussed four new traffic features - state-aware, aggregated, multi-source, and background - that OTG implements to generate realistic application traffic. For the case of the present paper, the tool is used to measure and analyze the one-way-delay over the different segments of the M2M network when deployed in the context of LTE system. Tow scenarios have been considered, namely virtual game and sensor-based alarm and event detection, to illustrate the supported features. The results show that the LTE may not be ready to support certain type of applications and that it is not fully optimized for the realtime traffics and for the massive deployment in M2M applications. As a next step, We plan to extend the scope of our analysis to a large number of sensors and study the performance of LTE network for the case of more complex scenarios with heterogeneous user profile (mobility, user speed for virtual game,etc.) and application on top (voice over IP, gaming, and sensors traffic).

Acknowledgment

The research leading to these results has received funding from the European Research Council under the European Community Seventh Framework Programme (FP7/2007- 2013) n^o 248993 (LOLA), n^o 257616 (CONECT) and WL-BOX-4G (SCS).

6. REFERENCES

[1] LoLa Consortium, LOLA project (Achieving Low-Latency in Wireless Communications), "D2.1 Target Application Scenarios," Available on: www.ict-lola.eu. 2010.

[2] LoLa Consortium, LOLA project (Achieving Low-Latency in Wireless Communications), "D3.5 Traffic Models for M2M and Online Gaming Network Traffic". 2011.

[3] A. Botta, A. Dainotti, and A. Pescapé. A tool for the generation of realistic network workload for emerging networking scenarios. 2012.

[4] M. Claypool and K. Claypool. Latency and player actions in online games. 2006.

[5] A. Hafsaoui, N. Nikaein, and L. Wang. OpenAirInterface traffic generator (OTG): A realistic traffic generation tool for emerging application scenarios. In *MASCOTS 2012, IEEE 20th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, 2012, Arlington, Virginia, USA*, 2012.

[6] IEEE802.16p-11/0014. IEEE802.16p M2M Evaluation Methodology Document (EMD). In *IEEE80*, 2011-05-24.

[7] M. Laner, P. Svoboda, P. Romirer-Maierhofer, N. Nikaein, F. Ricciato, and M. Rupp. A comparison between one-way delays in operating HSPA and LTE networks. In *8th International Workshop on Wireless Network Measurements (WinMee)*, 2012.

[8] M. Laner, P. Svoboda, and M. Rupp. Modeling randomness in network traffic. In *Proceedings of the Sigmetrics/Performance*, 2012.

[9] M. Laner, P. Svoboda, S. Schwarz, and M. Rupp. Users in cells: a data traffic analysis. WCNC, 2012.

[10] H. Lenz and J. Koss. M2m communication - next revolution on wireless interaction, 2008.

[11] N. Nikaein and S. Krco. Latency for real-time machine-to-machine communication in LTE-based system architecture. In *EW'11, 17th European Wireless Conference, Sustainable Wireless Technologies, April 27-29, 2011, Vienna, Austria*, 2011.

[12] J. Sommers and P. Barford. Self-configuring network traffic generation. In *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, IMC '04. ACM, 2004.

[13] Vishwanath, K. Venkatesh, and A. Vahdat. Realistic and responsive network traffic generation. In *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '06. ACM, 2006.

[14] www.netperf.org/.

[15] www.openairinterface.org.

[16] www.rtai.org/.