LTE Emulation over Wired Ethernet

Roman Chertov, Joseph Kim, and Jiayu Chen

The Aerospace Corporation
2310 E. El Segundo Blvd
El Segundo CA 90245, USA
{Roman.O.Chertov,Joseph.Y.Kim,Jiayu.Chen}@aero.org

Abstract. Long-Term Evolution (LTE) standard merges an all IP voice and data communications with dynamic spectrum resource scheduling. The resource scheduler must balance the QoS requirements, traffic demands, and physical channel conditions to create desirable wireless end-user performance. The purpose of our research and the focus of this paper is a development of a unified testbed platform based on Emulab that can be used to examine the key aspects of an LTE system in realtime, including real time uplink and downlink scheduling, QoS parameters, and Android end-user applications. Our validation studies demonstrate that the testbed is capable of achieving delay, loss, and jitter that can be associated with an LTE communication system, and can be easily used to study a variety of LTE scheduling algorithms.

1 Introduction

Over the past several years, mobile carriers began the transition towards the 3rd Generation Partnership Project (3GPP) Long-Term Evolution (LTE) standard. The LTE standard relies on Orthogonal Frequency-Division Multiple Access (OFDMA) for the downlink radio access. The uplink in LTE utilizes the Single-Carrier FDMA (SC-FDMA) radio access scheme. Both the uplink and the downlink can support multiple users concurrently by allowing the users to be partitioned in time and frequency. A centralized resource scheduler is required to create the time and frequency mappings for the users and the base station for the uplink and the downlink. LTE differs from the previous 3GPP standards as all of the communications are Internet Protocol (IP)-based, including the regular phone calls. The all-IP based nature of the communications requires Quality of Service (QoS) to ensure that realtime traffic such as voice gets priority over non-realtime traffic such as HyperText Transfer Protocol (HTTP). This feature of LTE means that the scheduler must be cross-layer-aware and must consider physical channel conditions as well as the Layer 3 QoS requirements.

A great breadth of research has been conducted regarding LTE up and downlink scheduling. However, the majority of works rely either on pure simulation [1,2] or on specialized hardware emulators [3]. The problem with simulation approaches is that the simulation does not allow the complete high-fidelity simulation of a real-world LTE system (end users, services, base station, etc.) as the complexity severely impacts validation and simulation times. In addition, a simulation approach requires rewriting the handset and server applications for the simulation environment. Naturally, this approach requires extensive validation to ensure that the rewritten applications follow the same behavior as their real-world counter parts. The hardware-based emulation solutions can emulate the entire system, but the hardware settings might be closed to the experimenter because of the proprietary nature of the hardware emulator. Hardware emulation systems typically suffer from scalability problems as they are primarily designed for testing and troubleshooting. Finally, the emulation hardware can be prohibitively expensive to some experimenters. In order to address these two limitations, we sought to create an LTE emulation system that incorporates (1) end-user handset IP stacks and applications, (2) QoS management, (3) real-world services, and (4) up/downlink LTE scheduling. Finally, the emulation system must allow for repeatable and reproducible experiments, use commodity hardware, and scale to several hundred or more users.

An LTE network is a system of systems, where the individual systems can have a direct impact on other systems. Hence, we were interested in creating a complete LTE emulator that could support the following types of scenarios: "How would an Android application, or the cloud service X react to the QoS or scheduler changes on the up/down LTE link?" "What is an impact of application Y in a heavily congested cell?" "What are the measurable impacts of multicast video distribution in an LTE network?" "What is an optimal QoS and pricing strategy for a carrier for a given network and end-user applications?"

As the foundation for the emulator, we have chosen to use the Emulab (`http://www.emulab.net`) [4] testbed platform from University of Utah. The testbed is composed of several hundred commodity PCs with multiple network cards and Cisco 6000 series enterprise switches. The high degree of connectivity between the commodity PCs coupled with custom switch Virtual Local Area Network (VLAN) management software allows experimenters to create arbitrary networks. Because of this capability, the Emulab testbed is well known in the networking community for its flexibility in creating arbitrary network topologies and its ability to conduct repeatable and reproducible experiments. Another notable feature of Emulab is that the testbed management source code is freely available for download, meaning that anybody can create an Emulab instance given sufficient hardware.

The other components of our LTE emulator were built around the Click Modular Router [5] (referred as Click from now on), Android Software Development Kit (SDK), and the Linux operating system. All of the chosen components are open source, thus allowing us to perform any necessary modifications. The resulting LTE emulation system is capable of running Android-based "soft" handsets, using highly configurable QoS settings, utilizing any Linux-based service (httpd, Session Initiation Protocol (SIP) proxy, etc.), and performing realtime uplink and downlink LTE scheduling duties. The majority of this paper focuses on the emulator architecture and the validation of the emulator. However, we did perform several showcase experiments that demonstrate the interaction between the LTE uplink scheduling, QoS settings, and the end-user application performance.

The remainder of the paper is organized as follows. Section 2 provides an overview of the related work. Section 3 describes our emulation architecture. Section 4 tests the fidelity of the emulation. Section 5 presents the results of our experiments. Finally, Section 6 concludes this paper.

2 Related Work

Several prominent scientific instrumentation companies such as Rohde & Schwarz, Agilent Technologies, and Nomor Research provide products that can integrate LTE hardware-based emulators with the user-supplied equipment to test end-to-end performance in the LTE environment [6,7,8]. The two main drawbacks of the hardware-based testers are the high price of the devices and the closed nature of the equipment. From a research standpoint, the proprietary nature of the LTE testers can possibly prevent experimenters from implementing and testing their own scheduling algorithms.

The closest works in the literature that is comparable to our LTE emulator is the Mobility Satellite Emulation Testbed (MSET) developed by Chertov et al. [10] and the OpenAir LTE emulator [9]. MSET was primarily designed to emulate single-carrier satellite Time Division Multiple Access (TDMA) links and did not implement a dynamic scheduling scheme that could operate on 10 ms time boundaries. Additionally, MSET did not provide a highly configurable QoS management system that is supported by our LTE emulator. The OpenAir LTE emulator developed by EURECOM is capable of emulating LTE physical and Media Access Control (MAC) layers over Ethernet, but it is more geared towards modeling the radio channel, while our system makes more emphasis on the scheduling aspects of LTE.

3 Architecture

The following section describes the components of the architecture that we have developed to run a complete emulated LTE network on The Aerospace Corporation testbed. Our testbed is based on Emulab [4] and utilizes 150 nodes. Emulab control software allows an experimenter to reserve a number of nodes and create an arbitrary network via Virtual Local Area Network (VLAN) manipulation on the experimental switches, which interconnect the testbed nodes. In addition, each reserved node is solely dedicated to the experiment owner who has root privileges. On Emulab, Base Stations (BSs) and end-user handsets can be implemented by using testbed nodes connected by a Local Area Network (LAN), as shown on Figure 1. In addition, all of the nodes utilize a separate 1 Gbps network interface for control traffic only to ensure that experimental and control packets do not interfere with each other. The rest of the section described the individual components that are necessary to turn an Emulab-type testbed into an LTE emulator.

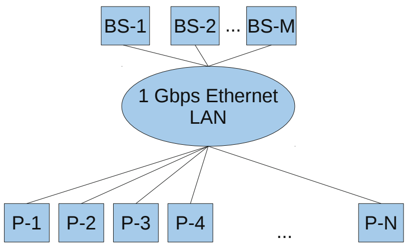
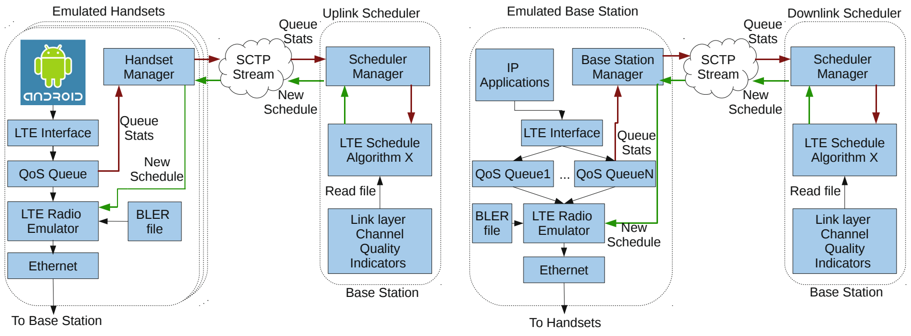


Fig. 1. Emulated LTE topology, where the base station and the end-user handset functions are performed by commodity PCs

3.1 LTE Radio Emulation

Radio emulation is the foremost critical component of our framework. The radio emulator is what allows the access point or the end-user handsets to control the bandwidth, jitter, and latency of the LTE network. In Figures 2(a) and 2(b), the `LTE Radio Emulator` component is responsible for the LTE radio emulation. Currently, the radio emulator does not model any physical layer parameters such as noise, multipath, etc. The primary function of the radio emulator is to ensure that the right number of bits is sent during an appropriate time slot. However, the radio emulation element can be easily modified to read from scenario files that specify Block Error Rate (BLER) for a given time instance in order to determine if packets need to be corrupted or not. The BLER scenario files can be derived by running physical layer simulations that take into account fading, multipath, antenna parameters and interference over a given time period.

Figure 3 shows a sample 10-subframe LTE uplink schedule where the dark gray blocks denote Transport Block (TB) allocations to a particular user. In an LTE schedule, if a user is assigned a set of K TBs in a frame, then the sum of the transmitted bits can be computed by adding up the usable bits for each



(a) Uplink (b) Downlink

Fig. 2. End-user handset and base station emulation

TB in the set K. Lets assume that each assigned TB uses the 16 Quadrature Amplitude Modulation (16QAM) modulation with a code rate of $\frac{1}{2}$. This coding when using an LTE waveform translates to 336 bits per TB[1]. This would imply that the user can transmit 672 bits during subframe 0, 1008 bits during subframe 5, and 1344 during subframe 9. Since a radio emulator permits the user or the base station to transmit only the assigned number of bits during the allotted subframes, the bandwidth, jitter, and latency will be affected as a result.

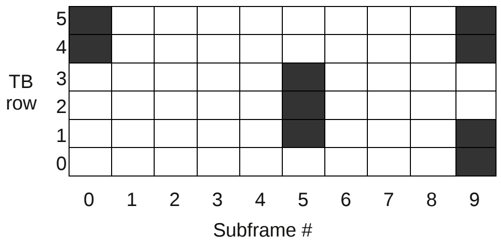


Fig. 3. Sample LTE schedule

We implemented the LTE radio emulation as Click element just like in the previous work on satellite TDMA emulation [10]. The radio emulation element in the emulated handset dequeues bits from an upstream queue only if the current LTE scheduler assigned TB(s) to the current user. The queues in Click are packet-based, hence we implemented an accounting scheme to keep track of how many bits of the current packet have been sent. Once all of the TBs assigned to the current packet have been "sent", the packet is dequeued from the queue and sent over the Ethernet. This accounting approach allows a packet to span several TBs, subframes, or even schedules. Alternatively, a single TB can hold multiple packets if its usable bit volume is high enough. It is easy to see that the time it takes to transmit a given packet is entirely dependent on the LTE scheduler and the allocations it creates for a given user. Therefore, the radio emulator can influence the network effects such as delay and jitter solely based on the schedule allocations just like the real-world LTE network. In this version of the emulator, we did not implement the Hybrid Automatic Repeat Request (HARQ) retransmission scheme, but we do plan to add it in the future.

The downlink can be emulated using the same radio emulation Click element such that the base station emulates only the downlink (packets to be transmitted to the end users). This arrangement allows for a complete LTE emulated network to run on the testbed.

3.2 QoS Queue Management

Instead of using a simple drop-tail Queue, we opted to develop a highly customizable queue management Click element, which is labeled as `QoS Queue` in

[1] $7\frac{symbol}{subcarrier} \times 12\frac{subcarrier}{RB} \times 4\frac{bits}{symbol} * \frac{1}{2} * 2\frac{RB}{TB} = 336\ bits$, (each TB is composed of two Resource Blocks (RB)).

Figures 2(a) and 2(b). The queue management element was designed to allow priority queues based on Differentiated Services Code Point (DSCP) values. The queue management element permits any user-specified mapping of DSCP code points into N priority queues, where queue zero is the highest priority and queue $N - 1$ is the lowest. The queues have an adjustable queue depth. Finally, each priority queue can be configured to delete packets that have been enqueued for more than X seconds. This feature can be useful when experimenting with real-time traffic, where packets that have experienced severe queuing delays might as well be dropped. Since the end-user handsets send packets to the base station, the queue management element maintains a set of priority queues only for the base station, as shown in Figure 2(a). On the base station, however, the queue management element creates a set of priority queues for each individual end user as shown in Figure 2(b).

The queue management element is tied to the LTE radio emulator element, such that the radio element pulls the packets out of the highest priority queue first, when the user has been allocated LTE resources. Alternatively, the LTE radio emulator can specify from which queue to dequeue if such information is provided in the schedule. Finally, the queue manager can provide a variety of statistics, such as queue depth, bytes seen, bytes dropped, etc., via the Click read handler.

3.3 Schedule Management

One of the key aspects of an LTE network is the centralized scheduler that creates up and downlink schedules. The scheduler is responsible for responding to user demands and assigning non-overlapping frequency and time resources to the users. In the real-world LTE network, the control channels are used to convey user channel quality conditions and traffic demands to the scheduler. In our emulated LTE architecture, we also utilize a control channel in the form of an Ethernet LAN that is separate from the experimental Ethernet LAN. Figure 2(a) shows the relationship between the end-user emulated handsets and the uplink scheduler. The emulated handsets submit the queue statistics to the uplink scheduler by accessing the Click read handler of the queue management element. The uplink scheduler aggregates all of the individual queue reports and submits them to the currently selected uplink scheduling algorithm. The uplink scheduler considers the aggregated queue statistics, Channel Quality Indication (CQI) values, scheduler parameters (end-user priorities, traffic type preferences, etc.) and then creates a global schedule that is transmitted to the emulated end-user handsets. The emulated end-user handsets in turn install the new schedule such that the LTE radio emulator can follow the new schedule. The downlink schedule management functions in exactly the same fashion. Figure 2(b) shows the relationship between the base station and the downlink scheduler. The CQI values for uplink and downlink can be derived in the same off-line simulation fashion as the BLER values used by the radio emulation element.

The schedule management system is created by using a client and server architecture that relies on Stream Control Transmission Protocol (SCTP) for communication. The handset or the base station managers can interact with the Click elements via the read/write handlers. The scheduler manager's primary goal is to aggregate the data from the clients and then submit the data to a scheduling algorithm. The output of the scheduling algorithm can then be disseminated to the handset or the base station managers. The scheduler manager architecture allows for the scheduling algorithm to be chosen at startup via a command line argument. This feature allows the user to rapidly experiment with a variety of schedulers just by restarting the scheduler manager. We have opted to use SCTP over Transmission Control Protocol (TCP) to ensure that queue aggregation, schedule creation, and schedule dissemination cycle can occur in under 10 ms. Additionally, we disabled the delayed transmissions to force SCTP to transmit the data immediately. This setting change was necessary to achieve the aggregation of end-user traffic demands and creation of a new LTE schedule in the 10 ms time frame.

The schedule management system also has the ability to time-synchronize the handset managers (not applicable for the downlink as there is only one base station). The time synchronization uses a technique similar to the Network Time Protocol (NTP) to measure the offset between the scheduler and the clients [11]. However, the clients do not change their local clocks to match the clock of the server. The clients adjust the start time of the next schedule based on the offset between the server by telling the LTE radio emulation element to either increase or decrease the start time of the next LTE frame. Such an approach can ensure that the clients synchronize their packet transmissions according to the global uplink schedule.

3.4 "Soft" Handsets

As one of the primary goals of our emulation effort was to run real-world cell phone applications, we have installed the Android emulator SDK on our testbed nodes. The Android emulator runs the actual Android firmware on a Qemu (`http://www.qemu.org`) Advanced RISC Machine (ARM) emulator. When running, the Android soft handset uses Qemu to send packets via the Linux IP stack. Using Click, we created a fake Ethernet device[2] called LTE, which receives all of the packets from the Linux IP stack destined for the experimental IP subnet. In turn, the fake Ethernet device injects packets into the kernel-level Click, which then uses our QoS queue and LTE radio elements to emulate the LTE MAC layer (see Figures 2(a) and 2(b)).

3.5 Admission Control and Mobility

Admission control can be accomplished by having a scheduler for a given base station deny a connection attempt by an end user in case there is a resource shortage. If the experimental scenario specifies that the end user is in reach of

[2] `FromHost` Click Modular Router element.

several base stations, then end user can be scripted to try the base stations in order until a connection can be established.

Our architecture allows for creating experiments where the users can migrate from one base station to another and the schedulers for the base station can admit or not admit new users. In our system, base station migration is nothing more than establishing a connection to a scheduler instance that manages a particular base station. In addition, the handset emulator needs to start utilizing an Ethernet MAC that corresponds to a new base station. MAC addressing ensures that the packets will be delivered only to the newly associated base station.

4 Emulator Validation

It was important to ensure, prior to conducting the experiments, that the emulation components produced the expected results. This required running a suite of calibration experiments to ensure that the specified delay, loss, and jitter of a given LTE network were achieved.

4.1 Experimental Layout

For our validation experiments, we have used the topology shown in Figure 1. The topology has 40 testbed nodes that run Fedora14 and use the Android emulator with Android firmware version 2.3.3. Even though the testbed supports 150 nodes, only 40 nodes were available to us as other nodes were down for maintenance or were used by other experimenters. One node serves as a base station. All of the nodes use Click Modular Router 2.0 and use our custom Click elements described in Section 3. For the validation experiments, we have chosen to concentrate on the uplink as it is more challenging from the emulation perspective: forty nodes must perform in unison to abide by the global uplink schedule versus just the base station shaping its own downlink. The schedule manager was configured to check the timing offsets between the base station node and the handset nodes every 500 ms and could adjust the frame offset by as little as 0.01 ms. In addition, an Exponential Moving Average (EMA) was used to keep track of the time offset between the handset node and the base station.

4.2 Validation Results

The uplink LTE emulator can be deemed successful if it achieves timing synchronization between the participants and the correct delay and jitter effects are produced. To test the uplink emulation, we have used a proportional schedule (equal allocation to each handset), shown in Figure 4. The proportional schedule evenly divides an LTE 10-ms frame with 6 TB rows among 30 handsets. Each handset gets two TBs ($\frac{60}{30} = 2$). The schedule in Figure 4 also shows to which

TB row										
5	2	5	8	11	14	17	20	23	26	29
4	2	5	8	11	14	17	20	23	26	29
3	1	4	7	10	13	16	19	22	25	28
2	1	4	7	10	13	16	19	22	25	28
1	0	3	6	9	12	15	18	21	24	27
0	0	3	6	9	12	15	18	21	24	27
Subframe #	0	1	2	3	4	5	6	7	8	9

Fig. 4. Proportional LTE schedule that assigns two TBs per handset

handset ID the TBs are assigned. For example, handset IDs 12, 13, and 14 can transmit bits only during the fifth millisecond of the 10-ms frame.

We have used the 16QAM modulation with a code rate of $\frac{1}{2}$ (one of the allowed LTE uplink settings), which equates to 672 bits per handset per frame (see Section 3.1). With a 672-bit allotment, each handset can transmit an 84-byte packet every 10 ms. Even though a proportional scheduler is used, every handset submits its queue statistics to the uplink scheduler, which disseminates the global schedule. This was done to ensure that all parts of the emulation were active.

The nodes that emulated the handsets were configured to transmit 84-byte UDP packets (includes Ethernet/IP/UDP headers) at 200 packets per second to the base station node for 100 seconds. The packet transmission rate was intentionally set too high, to determine if the LTE emulation shaping would take effect. The base station node used Click's `StoreUDPTimeSeqNum` element to embed time stamps into the packets as soon as they arrived and then used `ToDump` element to save the captured packets to disk in Packet CAPture (PCAP) format.

The packet inter-arrival times computed from the base station node's time reference point have the following statistics: mean – 10.0 ms, 5th percentile – 9.97 ms, 50th percentile – 9.985 ms, and 95th percentile – 10.036 ms. The inter-arrival statistics do indeed show that LTE emulation allowed only one packet per handset every 10 ms.

A much more interesting test is to determine if all the 30 handset nodes were synchronized on the uplink, and if the packet arrival times coincided with the global uplink schedule. The packet arrival times of the individual handset nodes were compared against a static repeating 10 ms schedule shown in Figure 4. We computed the time delta between the observed packet arrival time and its logical slot location in the schedule. The time deltas for all 30 handset nodes have the following statistics: mean – 0.279 ms, 5th percentile – 0.108 ms, 50th percentile – 0.23 ms, and 95th percentile – 0.623 ms. Visually, the data can be represented as shown in Figure 5. The numbers and their position represent the IDs of the handset nodes and the time of the packet arrival from that node, the solid vertical lines signify start/end of a 10-ms frame, and the dashed vertical lines signify 1-ms subframes. The data almost exactly matches the schedule shown in Figure 4 except for the time shift of around 0.23 ms (50th percentile time delta). Even though there is a small violation in the schedule, the fact that 30 handset nodes were able to almost exactly synchronize with the global schedule shows great promise in the emulator's capability. In the future, we can improve the timing by using Precision Time Protocol (PTP) or by coupling Global Positioning System (GPS) devices to the testbed nodes [10].

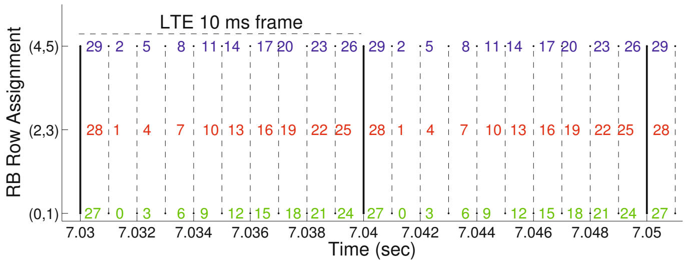


Fig. 5. Packet arrival times overlaid with an uplink LTE schedule

5 Experiments

In this section, we present several uplink experiments that show the interaction between scheduling, QoS, and end-user application performance. For the experiments, we used all 40 handset nodes and used the same LTE parameters as in Section 4, which equates to an uplink capacity of just over 2 Mbps. The primary goal of the experiments was to show that the interactions can be quite significant and warrant more detailed future studies.

5.1 HTTP Performance

To study the performance of HTTP from the end user's perspective, we have written a simple Android application that can time how long it takes to load a given webpage. Using this application and the Android Debug Bridge (ADB), we created a framework where we could instruct a handset to visit a web page, wait until it loads, and log the result to a log file.

As we were using a closed test environment, we developed a suite of nine synthetic web pages that included graphics and text. The pages were sized to mimic the webpage distribution observed by Google crawlers [12]. The base station node was configured to run an Apache web server and was loaded with the synthetic web pages. The handset nodes were configured to visit the web pages at random for 900 seconds, and used an exponential "reading" time with a mean of 30 seconds before visiting the next web page. The "reading" time distribution was taken from the IEEE C802.16m-07/074r1 evaluation document [13].

For this test, we have used two LTE uplink schedulers: proportional and on-demand round robin. Just like in the validation section, the proportional scheduler simply allocates $\lfloor \frac{60}{40} \rfloor$ TBs per handset and does consider the queue reports. On the other hand, the on-demand round robin scheduler allocates the resources to the handsets only if they report non-zero queues. The allocations are performed in a first fit fashion such that the first subframe must get filled before the TBs in the next subframe can be allocated. Finally, the handset scanning order is round robin to ensure fairness.

Figure 6 shows the average, min, and max web page load times when using only 1 Gbps Ethernet, LTE uplink proportional scheduling, and LTE uplink round robin on demand scheduling. It is interesting to see that the web page load times for on-demand LTE scheduling is comparable to the no-emulation results. Not surprisingly, the proportional scheduling does not perform as well, primarily because the resources are assigned to the handset nodes even when no HTTP get requests are issued, thus disallowing active handset nodes to take advantage of the unused resources.

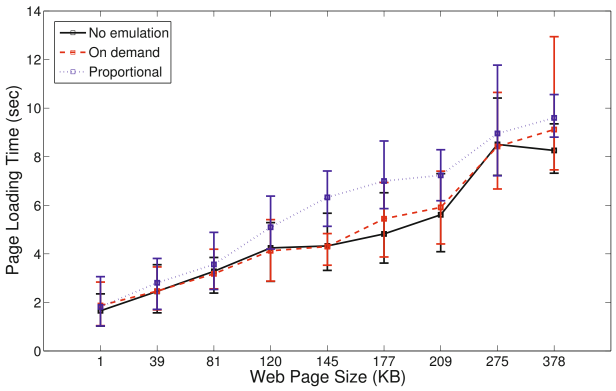


Fig. 6. Web page loading times

5.2 VoIP Performance

As cellular networks are primarily used for voice communications, we have performed several experiments with only VoIP calls. In our experiments, we used a Constant Bit Rate (CBR) application to generate UDP packets such that the packet size and the packet rate was representative of the G.711 codec [14]. The G.711 codec averages around 87 Kbps on Ethernet and the packets cover 20 ms of voice data, which equates to 50, 217-byte (including headers) packets per second. We chose the G.711 codes as it offers relatively high fidelity and can occupy a significant part of the uplink bandwidth.

To generate realistic call duration times observed by a single base station, we used the lognormal distribution with $\mu = 3.287$ and $\sigma = 0.891$, as was observed by F. Barcelo and J. Jordan [15]. We chose the lognormal instead of the recommended lognormal-3 distribution as we wanted to utilize the readily available Perl probability packages. For the call arrival times, we have chosen a Poisson distribution and varied the $\frac{1}{\lambda}$ (inter arrival mean) parameter between

60, 30, and 20 seconds. The expected number of concurrent calls can be derived from the following equation:

$$E[C] = N \times \frac{E[D]}{E[D] + \frac{1}{\lambda}}$$

where $E[D] = e^{\mu + \frac{1}{2}\sigma^2}$. The $\frac{1}{\lambda}$ values of 60, 30, and 20 equate to 16, 22.85, and 26.66 expected concurrent calls, respectively, for our 40-node topology.

Prior to conducting experiments with the LTE uplink emulation, we ran the voice scenario on the testbed for 10 minutes without any emulation using the arrival $\frac{1}{\lambda} = 60$ to measure jitter and loss values. As the VoIP protocol is most sensitive to jitter (inter arrival times) and packet loss ratios, we primarily concentrated on these network statistics. The inter arrival times for the VoIP packets were measured and the following statistics were recorded: mean – 20 ms, 5th percentile – 19.967 ms, 50th percentile – 20.017 ms, and 95th percentile – 20.018 ms. Just as expected, the inter arrival times were almost exactly 20 ms (50 pps) and there were no losses.

Next, we enabled LTE uplink emulation with an on-demand round robin scheduler and configured the queue manager to drop voice packets that sat in the uplink queue for more than one second. Additionally, the uplink queues were sized to hold 64 packets as it is a typical size for many network cards. Table 1 shows the obtained inter arrival times for call arrival $\frac{1}{\lambda}$ values of 60, 30, and 20. It is interesting to note that some losses occurred and that even though the mean inter-arrival values are almost 20 ms, the 5th, 50th, and 95th percentile values indicate increased levels of jitter compared to the pure Ethernet scenario. Scheduling is one of the primary reasons for jitter, as packets cannot be sent from a handset node until the scheduler grants the resources, and this can take 10 ms at a minimum. Additionally, even though the average call volume bandwidth does not exceed the 2 Mbps up link capacity, there can be instances when too many calls are in the system, thus leading to queue-based delays and queue overflows.

5.3 HTTP and VoIP

For the final set of experiments, we have combined HTTP and VoIP traffic. The handset nodes were configured to browse web pages and make VoIP calls as was described in the above sections. The HTTP and VoIP calls were allowed to occur independently of each other to mimic users that can browse and call at the same time.

As the handset nodes were allowed to browse and call at the same time, we have conducted two sets of experiments where we have given priority to VoIP over HTTP and where VoIP and HTTP were treated equally. Just like before, we ran the experiments for 10 minutes where 40 handset nodes requested webpages and made phone calls. To ensure a heavily loaded up link, we have used the call arrival $\frac{1}{\lambda}$ value of 20 sec. Also, the uplink LTE scheduler was configured to provide on-demand round robin allocations, and the queue manager was set to time-out

Table 1. VoIP Inter-Packet Delays (ms)

	Call arrival $\frac{1}{\lambda}$		
	60 sec	30 sec	20 sec
5th	0.998	0.999	0.998
50th	19.968	19.967	19.967
95th	38.968	39.985	40.031
mean	20.344	20.316	20.257
loss ratio	0.015	0.015	0.012

Table 2. VoIP Inter-Packet Delays (ms) in the Presence of HTTP

	Priority	No Priority
5th	0.997	0.995
50th	17.972	19.02
95th	42.979	41.032
mean	20.257	20.505
loss ratio	0.012	0.021

priority traffic after one second. Both of the priority and best-effort queues were configured to allow 64 packets before dropping any additional incoming packets.

Table 2 shows the VoIP packet inter arrival times when VoIP was given priority and when it was treated the same as HTTP. The jitter values for priority and non priority cases are quite similar. The main difference between priority and non priority cases was the loss ratios. As expected, the non priority experiment produced more VoIP packet losses on the uplink. Also, the results in Table 2 are similar to the results in Table 1. The slightly lower 50th percentile values in Table 2 can be attributed to a somewhat higher volume of random VoIP traffic during the experimental runs, which led to an increased amount of jitter.

Figure 7 shows the web page loading times when no VoIP is present, VoIP is given priority, and VoIP has the same priority as HTTP. When VoIP is given priority, the web page load times are considerably longer compared to when no VoIP is present. In addition, HTTP page load times experience a large amount of variance as HTTP get requests can be processed only after all of the VoIP queues have been drained. The HTTP page load times when VoIP was not given priority are similar to when no VoIP was used. This is the case because HTTP traffic on the uplink is primarily composed out of TCP SYN/ACK and HTTP get request messages. Since the HTTP uplink traffic is light, the VoIP loss ratios and jitter values are not significantly impacted, as Table 2 demonstrates. Based on this observation, a cellular provider can choose not to prioritize voice-over HTTP in order to significantly improve web page loading times while not significantly affecting VoIP communications.

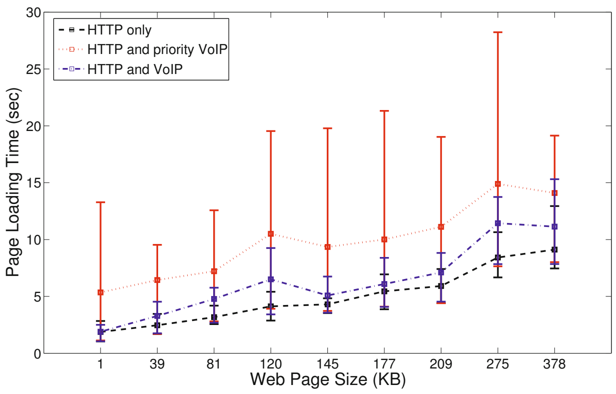


Fig. 7. Web page loading times in the presence of VoIP

6 Conclusion

In this paper, we have described an Emulab-compatible LTE emulation system that relies on commodity PCs and wired Ethernet. The emulation system is capable of emulating the LTE radio, LTE scheduler, QoS management, and the Android handsets and scales with the number of the available PC nodes. The all-encompassing aspect of the system permits experimentation with a wide variety of components: (1) end-user handset IP stacks and applications, (2) QoS management, (3) real-world services, and (4) up/downlink LTE scheduling. Our validation experiments have shown that the LTE emulation system can accurately produce network-level effects that are expected from an LTE setting including data transmissions on 10 ms time boundaries. Finally, our limited set of experiments has shown that scheduling and QoS settings can have a significant impact on end user performance. In our future work, we plan to investigate scheduling approaches that take into consideration CQI and BLER values, expected traffic arrival rates, and QoS settings. We plan to extend the emulator to allow mobility, an HARQ mechanism, and increase the number of possible end users by four fold via Xen virtualization (`http://xen.org`). Finally, we plan to run experiments that utilize several base stations and allow for user mobility between the stations.

Acknowledgments. This work was supported under The Aerospace Corporation's Independent Research and Development Program. In addition, we would like to thank David Blumenfeld for creating the synthetic web sites used in this paper.

References

1. Lee, S., Pefkianakis, I., Meyerson, A., Xu, S., Lu, S.: Proportional fair frequency-domain packet scheduling for 3GPP LTE uplink. In: Proc. of INFOCOM (2009)
2. Calabrese, F., Rosa, C., Anas, M., Michaelsen, P., Pedersen, K., Mogensen, P.: Adaptive transmission bandwidth based packet scheduling for LTE uplink. In: Proc. of Vehicular Technology Conference (VTC) (2008)
3. Tappayuthpijarn, K., Liebl, G., Stockhammer, T., Steinbach, E.: Adaptive video streaming over a mobile network with TCP-friendly rate control. In: Proc. of IWCMC (2009)
4. White, B., Lepreau, J., Stoller, L., Ricci, R., Guruprasad, S., Newbold, M., Hibler, M., Barb, C., Joglekar, A.: An integrated experimental environment for distributed systems and networks. In: Proc. of OSDI (2002)
5. Kohler, E., Morris, R., Chen, B., Jannotti, J., Kaashoek, M.F.: The Click Modular Router. Transactions on Computer Systems 18(3) (2000)
6. Rohde, Schwarz, R&S CMW500-PT HSPA+ and LTE Protocol Tester (2011), http://www2.rohde-schwarz.com/product/CMW500-PT.html
7. A. Technologies, E6621A PXT Wireless Communications Test Set (2011), http://www.home.agilent.com/agilent/product.jspxcc=US&lc=eng&ckey=1314599&nid=-33762.752176.00&id=14599cmpid=zzfindpxt
8. N. Research, LTE HSPA WiMAX application tester (2011), http://www.nomor.de/home/solutions-and-products/products/application-tester
9. EURECOM, OpenAir interface (2011), http://www.openairinterface.org
10. Chertov, R., Havey, D., Almeroth, K.: MSET: A mobility satellite emulation testbed. In: Proc. of INFOCOM (2010)
11. Mills, D.L.: Internet time synchronization: the Network Time Protocol. Transactions on Communications 39 (1991)
12. Google, Web metrics: Size and number of resources (2011), https://code.google.com/speed/articles/web-metrics.html
13. Novak, R., et al.: Proposed text for evaluation methodology and key criteria for p802.16m. IEEE C802.16m-07/074r1 (2007)
14. Cisco, Voice over ip - per call bandwidth consumption (2011), https://www.cisco.com/en/US/tech/tk652/tk698/technologies_tech_note09186a0080094ae2.shtml
15. Barcelo, F., Jordan, J.: Channel holding time distribution in cellular telephony. Proc. of Wireless Communications (1997)