Demo: CASTLE over the Air– Distributed Scheduling for Cellular Data Transmissions

Sandesh Dhawaskar Sathyanarayana*, Jihoon Lee*, Jinsung Lee*, Youngbin Im*, Parisa Rahimzadeh*,
Xiaoxi Zhang†, Max Hollingsworth*, Carlee Joe-Wong†, Dirk Grunwald*, Sangtae Ha*
* University of Colorado Boulder, †Carnegie Mellon University

ABSTRACT

We present the demonstration of a fully distributed scheduling framework called CASTLE (Client-side Adaptive Scheduler That minimizes Load and Energy) that jointly optimizes the spectral efficiency of cellular networks and battery consumption of smart devices. To do so, we focus on scenarios when many smart devices compete for cellular resources in the same base station: spreading out transmissions over time so that only a few devices transmit at once and improves both spectral efficiency and battery consumption. To this end, we devise two novel features in CASTLE. First, we explicitly consider inter-cell interference for accurate cellular load estimation in our machine learning algorithm.Second, we propose a fully distributed scheduling algorithm that coordinates transmissions between clients based on the locally estimated load level at each client. Our formulation for minimizing battery consumption at each device leads to an optimized back off-based algorithm that fits practical environments. Our comprehensive experimental results show that CASTLE's load estimation is up to 91 % accurate, and that CASTLE achieves higher spectral efficiency with less battery consumption, compared to existing centralized scheduling algorithms as well as a distributed CSMA-like protocol. Furthermore, we develop a light-weight SDK that can expedite the deployment of CASTLE into smart devices and evaluate it in a commercial LTE network.

CCS CONCEPTS

• **Networks → Mobile networks**; *Packet scheduling*; *Network measurement*;

KEYWORDS

LTE; Cell Load; Distributed Scheduling; Energy Saving

1 INTRODUCTION

We observe that the number of mobile devices is rapidly increasing on daily basis, which in turn increasing the mobile data traffic exponentially. To cope up with such increase in the traffic the new focus is to develop new low power wide area communication technologies. These kinds of devices are extensively used in Internet-of-Things (IOT) application. IOT is new area that will make human life much easier. Autonomous driving, smart cities/infrastructures and connected healthcare are few of the IOT based systems. Hence in future traffic will increase and most of it will be from these smart devices. Therefore these devices must build with long battery life, low device cost and extended coverage [1]. But the present cellular networks are ill equipped to handle the growing number of cellular devices and their data demand. Huge number of simultaneously active devices can lead to very low per device through-puts and even short-term unfairness [6]. Also, devices can incur significant battery consumption due to enlarged transmission times when the cellular load is very high or congested [2].

Given these limitations, many devices would prefer to utilize the network when the cellular load is low, and few other devices are active. By doing so, they can significantly increase spectral efficiency, and thus help the network cope with the increased amount of data traffic. However, this requires client devices to estimate the cellular load in real time with negligible energy expenditure, which presents the most interesting research challenge. For instance, the cellular load is impacted by the amount of traffic generated by other devices in the network and changes dynamically over time [5], as well as across geographical locations and environmental conditions [3].

Hence, we have developed the CASTLE, distributed framework that optimizes the spectral efficiency of cellular network and battery consumption in cellular devices. CASTLE achieves this by building light weight machine learning model which considers the RSRQ (Reference Signal Received Quality) and SINR as features, one of the important aspects in CASTLE is that it also considers the inter cell interference which improves the machine learning model prediction. This cell load information is locally coordinated between the different UE's and the distributed algorithm uses this for optimizing the spectral efficiency in network and reduces the battery consumption in each cellular device. The algorithm is qualitatively like LoadSense's Peek-n-Sneak protocol [2], in which each client performs a simple CSMA-like operation before obtaining a scheduling opportunity from the BS based on the binary estimation of cellular load. However, we take a theory-driven approach that guarantees optimal performance. Further We have implemented CASTLE as a light-weight software development kit (SDK) on the Android platform that provides cellular load estimation as well as a distributed scheduling service. To minimize the SDK's energy expenditure, as is needed for smart applications, we exploit several techniques including a lookup-based load inference using passive UE's measurements. Note that our full paper can be found in [4].

2 CASTLE SDK

At each UE, the channel inference module of CASTLE maintains a mapping table (Cell Load Lookup Table) that inputs the RSRP, RSRQ, and SINR values and outputs the estimated load class. The table consists of 45,198 entries where each entry is encoded with 2 bits to represent four classes of cell load. The total table size is

MobiSys '19, June 17–21, 2019, Seoul, Republic of Korea

ACM ISBN 978-1-4503-6661-8/19/06…$15.00
https://doi.org/10.1145/3307334.3328576

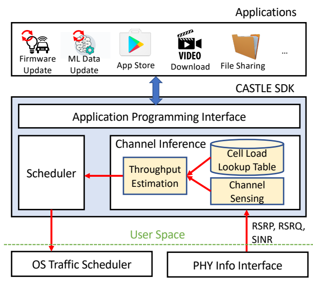

Figure 1: CASTLE architecture.

about 89 KBytes which is reasonably small for today's smart devices. We also implement CASTLE's Scheduler module. It estimates the throughput using the RSRQ, RSRP, SINR and mapping table at every time slot (i.e., 1 sec), and decides when the download starts. The CASTLE SDK is available in [9], which includes the following APIs:

- castle_predict_class_short() - predict the class with instantaneous RSRQ, RSRP, SINR measurements.
- castle_predict_class_long() - return the class mostly picked from the recent 10 consecutive measurements. In general, it is more accurate than castle predict class short(), since it can ignore a UE measurement error.
- castle_schedule() - schedule a download. It compares the estimated throughput $A(t)$ and the threshold A^* to decide when the download starts. If $A(t) < A^*$, it stays silent for this time slot.

3 DESCRIPTION OF DEMONSTRATION

As shown in Figure 1, CASTLE SDK's expected throughput is estimated by the Channel Inference module and fed into the Scheduler module. The Channel Sensing module periodically obtains PHY-layer information. To minimize the computation overhead, CASTLE uses the Cell Load Lookup Table, which populates the cell load level for each RSRP, RSRQ, CQI and SINR pair by using both analytical models and machine learning for the inter-cell inference scenario. This enables an efficient O(1) lookup. The Scheduler module then decides the transmission schedule for the application traffic according to the CASTLE algorithm.

We built an end-to-end LTE test bed to evaluate CASTLE, as shown in Figure 2. The testbed consists of UEs, eNodeBs, EPC (Evolved Packet Core) and application servers. For the eNodeBs, we use two commercialized indoor LTE Band3 small cell products, Juni JL620 [7], which are connected to GPS to correct frequency offsets. We placed multiple UEs from various vendors and models inside a shield box, so that the UE's can communicate with the eNodeBs via a pair of antennas inside the box. The signal attenuator installed between the antenna located in the box and eNodeBs outside provides appropriate signal strengths to the UE's and further allows us to emulate a variety of RF situations, including interference from neighbor cells. For EPC, we use the open source software from NextEPC [8]. The application servers gather cell-specific information from both the UE's and eNodeB's.

For our demo, we choose 8 UE's inside a shield box with lookup table installed in each of them. Each UE will download file from FTP server. All the UE's will start sensing the channel and decide

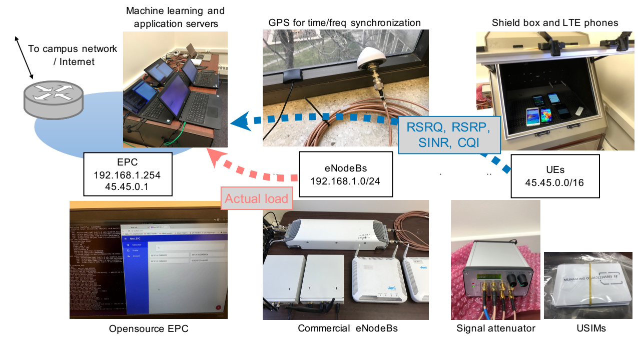

Figure 2: Our controllable end-to-end LTE network testbed.

whether to download the file from the server using the CASTLE algorithm. We will see only those UE's will download the file which has got the decision to download file instead of all the UE's trying to download simultaneous. Figure 3 shows the individual UE's download process with progress bar. The video clip is available at https://youtu.be/oEq0jz9wY0s, which demonstrates the same scenario from our in-lab LTE testbed and compares it with normal download. We will show that CASTLE can jointly optimize the spectral efficiency of cellular network and the battery consumption of smart devices.

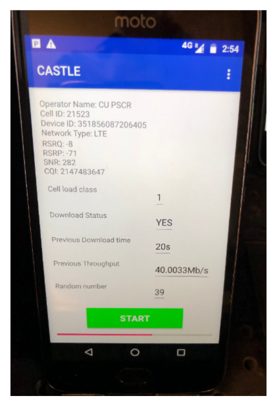

Figure 3: Snapshot of CASTLE running on a single UE.

REFERENCES

[1] 3GPP RAN WG 3. 2016. 3GPP Standards for the Internet-of-Things. https://goo.gl/DAzMLT. (2016).
[2] Abhijnan Chakraborty et al. 2013. Coordinating Cellular Background Transfers Using Loadsense. In *Proceedings of ACM MobiCom*.
[3] Fengli Xu et al. 2017. Understanding Mobile Traffic Patterns of Large Scale Cellular Towers in Urban Environment. *IEEE/ACM Transactions on Networking* 25, 2 (April 2017), 1147–1161.
[4] Jihoon Lee et al. 2019. CASTLE over the Air: Distributed Scheduling for Cellular Data Transmissions. In *Proceedings of ACM MobiSys*.
[5] Swarun Kumar et al. 2014. LTE Radio Analytics Made Easy and Accessible. In *Proceedings of ACM SIGCOMM*.
[6] Yasir Zaki et al. 2015. Adaptive Congestion Control for Unpredictable Cellular Networks. In *Proceedings of ACM SIGCOMM*.
[7] Juni. 2017. Enterprise Small Cell JL620. (2017). http://www.juniglobal.com/product/jl-620fdd-jlt-621tdd/
[8] NextEPC. 2018. Open source implementation of EPC. (2018). https://nextepc.org/
[9] Sandesh Dhawaskar Sathyanarayana. 2019. CASTLE SDK. (2019). https://github.com/cu-pscr/CASTLE_LIBRARY.git