RAN Resource Usage Prediction for a 5G Slice Broker

Craig Gutterman†, Edward Grinshpun‡, Sameer Sharma‡, Gil Zussman†
†Electrical Engineering, Columbia University, ‡Nokia Bell Labs

ABSTRACT

Network slicing will allow 5G network operators to offer a diverse set of services over a shared physical infrastructure. We focus on supporting the operation of the Radio Access Network (RAN) slice broker, which maps slice requirements into allocation of Physical Resource Blocks (PRBs). We first develop a new metric, REVA, based on the number of PRBs available to a single Very Active bearer. REVA is independent of channel conditions and allows easy derivation of an individual wireless link’s throughput. In order for the slice broker to efficiently utilize the RAN, there is a need for reliable and short term prediction of resource usage by a slice. To support such prediction, we construct an LTE testbed and develop custom additions to the scheduler. Using data collected from the testbed, we compute REVA and develop a realistic time series prediction model for REVA. Specifically, we present the X-LSTM prediction model, based upon Long Short-Term Memory (LSTM) neural networks. Evaluated with data collected in the testbed, X-LSTM outperforms Autoregressive Integrated Moving Average Model (ARIMA) and LSTM neural networks by up to 31%. X-LSTM also achieves over 91% accuracy in predicting REVA. By using X-LSTM to predict future usage, a slice broker is more adept to provision a slice and reduce over-provisioning and SLA violation costs by more than 10% in comparison to LSTM and ARIMA.

CCS CONCEPTS
• Networks → Wireless access points, base stations and infrastructure; • Computing methodologies → Neural networks;

KEYWORDS
Mobile Traffic Forecasting, Deep Learning

ACM Reference Format:

1 INTRODUCTION

It is expected that 5G networks will support a variety of services including smart cities, autonomous and network assisted driving, augmented reality, and virtual reality. Such services will impose an extremely diverse set of requirements on the mobile network, ranging from ultra high throughput to ultra low latency at the order of milliseconds [14].

Network slicing will allow 5G operators to split a shared physical infrastructure into virtual slices to meet these diverse requirements (see Fig. 1). The Next Generation Mobile Network (NGMN) Alliance defines a network slice as a set of network functions and associated resources, forming a complete virtualized end-to-end logical network meeting certain network characteristics required by the associated service [1, 2, 16]. Namely, slices will provide virtualized resource separation for different services, while still allowing for statistical multiplexing of the resources.

An anticipated challenge is managing a large number of tenants, each with multiple services, resulting in separate slice instances. Each such instance may have unique Service Level Agreement (SLA) requirements in terms of bandwidth, latency, reliability, mobility, and security. Each slice can contain multiple bearers from multiple User Equipments (UEs)†, and each bearer can have a Guaranteed Bit Rate (GBR)‡ or a non-GBR service.

The complexity of management and orchestration will increase with 5G slicing. As illustrated in Fig. 1, a Radio Access Network (RAN) slice broker (to which we will refer to as a broker) is used to manage and orchestrate the slice life cycle [23]. The broker monitors each slice’s SLA and predicts its future RAN resource usage. The prediction is utilized to dynamically provision resources to slices. Admission control decisions are based on the slice priority

†A bearer is defined as a path for the traffic with a common QoS from a UE to the Packet Data Network Gateway.
‡GBR bearers have bandwidth guarantees (e.g., min throughput, max throughput, packet delay variation) from the LTE network.
Slice admission control is out of the scope of this paper and is for future work.

Accordingly we define a new metric, REVA, that precisely measures the average amount of wireless Physical Resource Blocks (PRBs) available to very active bearers for each slice. The metric could be easily used for slice provisioning. The second objective is to develop an accurate and short time scale prediction model of that metric. Once the prediction is obtained, the broker can use each slice’s predicted PRB usage for slice provisioning.3

Efficiently utilizing the RAN resources is crucial to enable multiplexing gains and cost effectiveness for service providers [19]. Therefore, an accurate prediction model of the usage must be developed to efficiently utilize the RAN. An overestimation in the amount of allocated resources results in a decrease in revenue for the service provider, while underestimation results in SLA violations.

Specifically, the paper has two related objectives. The first is to develop a metric that measures the amount of Physical Resource Blocks (PRBs) available to very active bearers for each slice. The metric could be easily used for slice provisioning. The second objective is to develop an accurate and short time scale prediction model of that metric.

Figure 2: (a) An example of provisioning resources to slices which is based on the broker’s admission control decisions, where in the second decision interval a 5th slice is admitted. (b) An example of monitoring REVA for a single slice and the corresponding dynamic resource provisioning.

- 3Slice admission control is out of the scope of this paper and is for future work.

Vol. 51, No. 1, January 2019

3 RELATED WORK

Network Slicing: Architectural aspects of 5G RAN slicing are developed by the 5G NORMA project [4] within 5G-PPP, by utilizing Software Defined Networks and Network Function Virtualization. Wireless RAN virtualization will offer greater flexibility for network infrastructure operators, while also adding benefits to their customers (typically called tenants [21]). By enabling RAN virtualization, Mobile Network Operators can share common RAN resources leading to reduced costs and increased energy efficiency. The concept of network virtualization will enable infrastructure as a service for end to end networking [1]. An optimization framework was developed in [18] for resource allocation of network bandwidth and cloud processing. A network slice broker will enable mobile operators to request and lease infrastructure dynamically [23]. Orion was developed to enable dynamic virtualization of the base station [9]. In [7, 25], the authors develop an admission control decision algorithm for RAN slice requests based on the knapsack problem and propose solutions using a greedy algorithm and online-reinforcement learning. The resource efficiency and cost-effectiveness of resource management in network slicing is studied in [19]. Slice overbooking has been shown to maximize the revenue of mobile operators with minimal impact of SLAs [22].

Cellular Traffic: Recent work has characterized and modeled citywide traffic in cellular networks [29, 30, 32, 35, 36], where congestion is characterized by measuring the traffic load. In [11], the authors improve on previous congestion metrics by also including to collect traces of hundreds of hours of RAN resource allocation under a variety of network usage patterns. We used one, two, and three overlapping periodic time patterns to emulate the temporal patterns that occur in cellular networks [29].

The data collected from the testbed is used to develop and evaluate the prediction models for REVA (see Sec. 6). Current time series models [6, 13, 26] are inadequate for multistep prediction of network resource usage over a short time scale. These models are designed for predicting one step into the future, but there is a need for higher accuracy over multiple time steps for dynamic provisioning and other network optimization techniques (e.g., VM migration). Therefore, we design a modified Long Short Term Memory (LSTM) model, X-LSTM, to improve prediction accuracy. To evaluate the performance (see Sec. 7), we use X-LSTM to predict REVA tens of seconds in advance. We show that the gains of X-LSTM over traditional models such as Autoregressive Integrated Moving Average Model (ARIMA) and LSTM neural networks increase as the number of components in the time pattern increases. Given time patterns composed of one, two, and three independent semi-periodic components, X-LSTM outperformed ARIMA and LSTM by 10%, 22%, and 31% respectively. We show that X-LSTM achieves accuracy predictions of over 91%.

To evaluate the impact of each prediction model on slice provisioning, we introduce a simple slice provisioning algorithm. The algorithm exploits the prediction models to minimize cost for service providers. The cost is measured by the amount of over-provisioned PRBs and violating the SLA. We show that X-LSTM offers the service provider a greater than 10% cost reduction compared to ARIMA and LSTM.

2 RELATED WORK

Network Slicing: Architectural aspects of 5G RAN slicing are developed by the 5G NORMA project [4] within 5G-PPP, by utilizing Software Defined Networks and Network Function Virtualization. Wireless RAN virtualization will offer greater flexibility for network infrastructure operators, while also adding benefits to their customers (typically called tenants [21]). By enabling RAN virtualization, Mobile Network Operators can share common RAN resources leading to reduced costs and increased energy efficiency. The concept of network virtualization will enable infrastructure as a service for end to end networking [1]. An optimization framework was developed in [18] for resource allocation of network bandwidth and cloud processing. A network slice broker will enable mobile operators to request and lease infrastructure dynamically [23]. Orion was developed to enable dynamic virtualization of the base station [9]. In [7, 25], the authors develop an admission control decision algorithm for RAN slice requests based on the knapsack problem and propose solutions using a greedy algorithm and online-reinforcement learning. The resource efficiency and cost-effectiveness of resource management in network slicing is studied in [19]. Slice overbooking has been shown to maximize the revenue of mobile operators with minimal impact of SLAs [22]. Cellular Traffic: Recent work has characterized and modeled citywide traffic in cellular networks [29, 30, 32, 35, 36], where congestion is characterized by measuring the traffic load. In [11], the authors improve on previous congestion metrics by also including to collect traces of hundreds of hours of RAN resource allocation under a variety of network usage patterns. We used one, two, and three overlapping periodic time patterns to emulate the temporal patterns that occur in cellular networks [29].

The data collected from the testbed is used to develop and evaluate the prediction models for REVA (see Sec. 6). Current time series models [6, 13, 26] are inadequate for multistep prediction of network resource usage over a short time scale. These models are designed for predicting one step into the future, but there is a need for higher accuracy over multiple time steps for dynamic provisioning and other network optimization techniques (e.g., VM migration). Therefore, we design a modified Long Short Term Memory (LSTM) model, X-LSTM, to improve prediction accuracy. To evaluate the performance (see Sec. 7), we use X-LSTM to predict REVA tens of seconds in advance. We show that the gains of X-LSTM over traditional models such as Autoregressive Integrated Moving Average Model (ARIMA) and LSTM neural networks increase as the number of components in the time pattern increases. Given time patterns composed of one, two, and three independent semi-periodic components, X-LSTM outperformed ARIMA and LSTM by 10%, 22%, and 31% respectively. We show that X-LSTM achieves accuracy predictions of over 91%.

To evaluate the impact of each prediction model on slice provisioning, we introduce a simple slice provisioning algorithm. The algorithm exploits the prediction models to minimize cost for service providers. The cost is measured by the amount of over-provisioned PRBs and violating the SLA. We show that X-LSTM offers the service provider a greater than 10% cost reduction compared to ARIMA and LSTM.
round trip times. In [17] a single cell load measure is defined as a combination of the number of connected bearers, achieved throughput, and percentage of total PRBs per bearer. In [28] congestion is characterized via skewness of measured aggregate throughput.

**Network Analytics:** Numerous research efforts have been devoted to improving network performance using network analytics. In [5, 27] traffic congestion and mobility is predicted across a university WLAN network. Improvements for adaptive video streaming performance over LTE are studied in [33, 34, 37] where the LTE bandwidth is estimated by monitoring the broadcast messages or LTE bandwidth availability is given.

**Time Series Modeling:** Statistical and machine learning models for forecasting time series have received significant attention. Common statistical models are ARIMA and the Seasonal ARIMA (SARIMA) models [6, 26]. Neural Networks (NNs), a popular model in machine learning, are used to approximate non linear multivariate functions. Recurrent Neural Networks (RNNs) is a NN that uses feedback from previous steps. A specific type of RNN is an LSTM NN, which has memory cells to maintain information for longer periods [13]. Additional work has been done to improve the performance of LSTMs for specific applications [8, 10, 20].

### 3 WIRELESS RAN

In this section, we provide background on RAN resource allocation and then discuss the limitations of the existing metrics available for monitoring resource utilization. Note that we use LTE terminology while describing and evaluating our methods, but they should be applicable to 5G schedulers which are currently in development.

#### 3.1 Background on RAN Resource Allocation

3GPP defines wireless resource allocation in the time and frequency domains. LTE and LTE Advanced utilize a resource allotment unit called a PRB. A PRB consists of 180 kHz in the frequency domain and one slot of 0.5 ms in the time domain. Every Transmission Time Interval (TTI) (1 ms in LTE), the scheduler distributes the available PRBs for the Downlink (DL) and Uplink (UL) among LTE bearers. The total number of PRBs assigned depends on the number of TTIs and the system bandwidth configuration. According to the LTE standard, there are 6 PRBs per TTI for 1.4 MHz configuration to 100 PRBs for 20 MHz configuration [2].

The scheduler at the base station (eNodeB) uses channel condition information received periodically from the UEs to assign Modulation and Coding Schema (MCS) to the allocated PRBs. This essentially determines the number of bits transmitted using the allocated PRB. Using a higher MCS with poor channel conditions leads to data loss and requires using more PRBs for retransmissions. In good channel conditions, using a lower MCS leads to unnecessarily reduced throughput. RAN scheduling algorithms are optimized to determine the best MCS assignment for each allocated PRB. In the time domain, the schedulers also make decisions regarding how often PRBs are assigned to a specific bearer.

The LTE standard defines scheduling priorities or QCIs to address the different scheduling rules for the different classes of service [3]. Each QCI has its own associated QoS characteristics (e.g., priority, guaranteed (or not) bit rate, packet delay budget, and packet error loss rate). QCIs reserved for GBR service have the scheduler attempt to ensure certain guaranteed bitrate for the bearer. For example, QCI 1 is typically reserved for Voice over IP traffic. QCIs designed for non-GBR traffic typically use weighted or max-min fair share scheduling algorithms for PRB allocation [3]. Max-min fair share algorithms assign users with a small demand the resources that they need and distribute the remaining resources evenly to large users.

#### 3.2 RAN Resource Utilization Metrics

The broker allocates PRBs based on the SLA of each slice. In order to get better insight into provisioning resources, it is important to understand how bearers in that slice are currently utilizing the resources. In addition, it would be insightful to separate PRB usage and user channel conditions. Each have an essential role in the throughput and latency of individual UEs. RAN usage has been studied by several previous research efforts [11, 17, 28–30, 35]. However, they specifically do not focus on the application of RAN slicing. Below are examples of metrics that are not adequate for a broker.

- **Aggregate percent of available PRB utilization per second by the scheduler [17]** - There is no sense of fairness and relation to per bearer SLA. A single greedy application such as FTP can utilize close to 100% of all PRBs in the LTE RAN, if there are no other bearers served by the RAN. Clearly, the RAN serving just a single client is not congested, and if a second FTP client joins, the scheduler would allocate to it roughly half of the available PRBs.

- **Aggregate throughput of all bearers [29, 30, 35]** - The metric is inadequate for the same reasons given above. A single FTP bearer in perfect channel conditions can utilize close to 100% of all PRBs.

- **Metrics based upon latency or throughput of individual or groups of bearers [11, 28]** - These metrics are inadequate due to the reasons below:
  - Bearer throughput depends on channel conditions. Low throughput or high latency of bearer(s) may not result from RAN congestion but could result from poor channel conditions of the respective bearer(s).
  - Low throughput may be a function of applications usage characteristics. Specific applications may not need a lot of network resources (e.g., Voice over IP, low resolution video, and instant messaging).

- **Number of users served by the RAN** - Such a metric does not take into account RAN resource consumption by individual bearers. A RAN serving a large number of VoIP or other low volume bearers is not necessarily congested.

### 4 RAN RESOURCE ESTIMATION

In this section we outline the design objective for a prediction model along with the REVA metric. We then describe the definitions needed for REVA, and the algorithm for its computation.

#### 4.1 Objective

Our objective is to develop a metric, REVA, that can be used by a broker to efficiently measure, predict usage, and provision slice resources. We represent REVA as $y_t$ throughout the remainder of the paper. We assume the broker has a history of $T$ decision-making
Without loss of generality, we assume one or more QCIs per slice. A prediction model \( f \) uses the history \( y_{t-1} \) to predict \( s \) decision intervals ahead: 
\[
y_{t}, y_{t+1}, \ldots, y_{t+s-1} = f(y_{t-1}).
\]
The goal is to develop a prediction model for \( y_t \) that has a minimal prediction error \( \epsilon_t \):

\[
y_t = f(y_{t-1}) + \epsilon_t.
\] (1)

The REVA metric is developed as follows:

- A function of the available resources that is independent of: (i) channel conditions of the bearers; (ii) the application behavior and throughput needs of individual user bearers; (iii) transport protocol (e.g., TCP, QUIC, UDP, or raw IP); (iv) bearer throughput or roundtrip time. This would allow for scheduling slices based on the required PRBs.
- The average number of PRBs used by the bearers that attempt to obtain more than their maximal fair share of PRBs (defined as very active bearer in Definition 4.2). These are the bearers that need to be monitored to ensure SLAs.
- A method for precise and direct computation of available bearer throughput per slice. When combining the amount of average PRBs (\( \bar{PRB}_{b} \)) with individual channel bearer conditions (\( b_i \)), it allows for easy derivation of the wireless throughput available to a very active bearer \( i \). The throughput can be computed as:

\[
R(b) = \bar{PRB}_{b} \cdot C(b_i)
\] (2)

where \( C(b_i) \) is the average number of useful bits per PRB for bearer \( i \) [31]. The forecast of the UEs channel quality can be used to estimate the MCS [24]. Table 1 illustrates the average throughput given a variety of PRB rates along with the user's MCS. For each PRB range, 3 MCS values are provided along with the corresponding resulting max throughput.

### 4.2 Definitions

We introduce the following definitions prior to defining REVA. Without loss of generality, we assume one or more QCIs per slice.

**Definition 4.1. Active bearers** for a non-GBR QCI \( m \) are bearers that use an average \( y \) PRBs per second (e.g., \( y = 30 \))\(^5\). Active bearers can be broken down into two groups: Very Active and Less Active.

**Definition 4.2. Very Active (VA)** bearers for a non-GBR QCI \( m \) are those that continuously attempt to obtain more than a maximal fair share of PRBs that are available from the scheduler for a given duration of time.

**Definition 4.3. Less Active (LA)** bearers for a non-GBR QCI \( m \) are the active bearers that are not VA.

Examples of VA bearers are FTP and HTTP adaptive streaming video. Examples of LA bearers are web browsing and viewing social media. An example of a non-active bearer is a smartphone application that periodically performs keep-alive handshakes and receives push notifications. Bearers for each slice are classified into VA and LA based upon PRB resource consumption.

REVA is now formally defined as the following:

\[^5\] PRBs limits the max throughput of a bearer to ~ 1 KBps with channel conditions appropriate for 16QAM modulation. This parameter can vary based on the service type of the slice.

### Table 1: Throughput for PRB rate along with the UE’s MCS.

<table>
<thead>
<tr>
<th>Average PRBs/sec</th>
<th>MCS</th>
<th>Max Throughput (kb/sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-500</td>
<td>8,19,27</td>
<td>70,180,310</td>
</tr>
<tr>
<td>500-1000</td>
<td>8,19,27</td>
<td>140,360,630</td>
</tr>
<tr>
<td>1000-2000</td>
<td>8,19,27</td>
<td>280,720,1260</td>
</tr>
<tr>
<td>2000-3000</td>
<td>8,19,27</td>
<td>420,1080,1900</td>
</tr>
<tr>
<td>3000-5000</td>
<td>8,19,27</td>
<td>700,1800,3150</td>
</tr>
</tbody>
</table>

### Table 2: Notation.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>( s )</td>
<td>Fraction of control plane PRBs available to a slice over interval ( \Delta t )</td>
</tr>
<tr>
<td>( w_m )</td>
<td>Proportional fair weight for QCI ( m )</td>
</tr>
<tr>
<td>( RPRB_m )</td>
<td>Reserved PRBs for QCI ( m )</td>
</tr>
<tr>
<td>( B_m )</td>
<td>Vector of PRBs of active bearers at QCI ( m )</td>
</tr>
<tr>
<td>( BR_m )</td>
<td>Vector of PRBs of unclassified active bearers at QCI ( m )</td>
</tr>
<tr>
<td>( LAm )</td>
<td>Vector of PRBs of LA bearers at QCI ( m )</td>
</tr>
<tr>
<td>( Nm )</td>
<td>Number of active bearers at QCI ( m )</td>
</tr>
<tr>
<td>( NR_m )</td>
<td>Number of unclassified active bearers at QCI ( m )</td>
</tr>
<tr>
<td>( Um )</td>
<td>Number of PRBs used by LA bearers at QCI ( m )</td>
</tr>
<tr>
<td>( Im )</td>
<td>Fair share of PRBs at QCI ( m )</td>
</tr>
</tbody>
</table>

### Procedure 1 Aggregate available PRB Rate

1. procedure \( s_m \)
2. \( S = S_{\text{total}}(1 - \delta) - \sum_{j \in S_{\text{total}}} \bar{PRB}_{j} \)
3. if Proportional Weighted Share Scheduling then
4. \( S_m = S - \sum_{j \in S_{\text{total}}} \min(\bar{PRB}_{j}, S \cdot w_j) \)
5. if Strict Priority Scheduling then
6. \( S_m = S - \sum_{j \in S_{\text{total}}} S_j - \sum_{j \in S_{\text{total}}} \bar{PRB}_{j} \)

return \( S_m \)

### Definition 4.4. REVA for a slice is defined per QCI and traffic direction (DL or UL) as: available Resource rate (in PRBs/sec) for an ideal ‘Very Active’ bearer.

REVA determines the number of PRBs that a VA bearer at a given QCI can obtain. REVA specifically focuses on non-GBR bearers since, for GBR bearers a guaranteed amount of resources are allocated. The GBR service class guarantees the throughput and therefore the amount of PRBs allocated to a GBR bearer depends upon bearer channel conditions which typically varies in time. Notice that we use 1 second (1,000 TTIs) as the time interval. For low latency slices, one can use a smaller time interval (e.g., 20 to 100 TTIs).

The algorithm to calculate REVA appears in Algorithm 1. Bearers are categorized in an iterative way similar to the max-min fair share algorithm. In each iteration, LA bearers are those that use less than their fair share of the resources remaining. We define \( S_{\text{total}} \) as the total number of PRBs/sec available to a slice. For example, for a slice with 10 MHz bandwidth, \( S_{\text{total}} = 50,000 \) PRBs/sec. The available PRBs for the slice is \( S_{\text{total}}(1 - \delta) \), where \( \delta \) represents a fraction of control plane PRBs.

In this section, we assume that the slice includes multiple QCIs and we compute the REVA metric for each non-GBR QCI. Algorithm 1 consists of two steps:
Algorithm 1 REVA Computation

1: Initialize $\delta$, $s_{total}$
2: for $m = 5:9$ do
3: Initialize $B_m$, $N_m$, $L_{\Lambda \mu}$
4: $BR_m = B_m$, $NR_m = N_m$, $N_{m,p} = 0$, $U_m = 0$
5: Calculate $S_m$ (Procedure 1)
6: while $NR_m \neq N_{m,p}$ do
7: $N_{m,p} = NR_m$
8: $I_m = \frac{S_m-U_m}{\max(1,N_{m,p})}$
9: Update $BR_m$, $NR_m$, $U_m$, $L_{\Lambda \mu}$
10: $REVA(m) = \frac{S_m-U_m}{V \Lambda_m}$

- Compute available PRB rate per QCI of the slice. (line 5 of Algorithm 1, Procedure 1)
- For each QCI, classify the slice bearers into VA and LA based upon their PRB consumption. Then, compute the REVA value. (lines 6-10 of Algorithm 1)

$B_m$ is the vector of PRB rates for all the active bearers at $QCI_m$. Initialization is done by assigning $N_m$ as the total number of active bearers at $QCI_m$. $BR_m$ is the vector of PRB rates for all the active bearers that have not been classified yet, and is initially set to $B_m$. $NR_m$ is denoted as the number of active bearers that have not been classified yet and is initially set to $N_m$. $U_m$ is the number of PRBs used by LA bearers, and is initially 0. $L_{\Lambda \mu}$ is the vector of PRB rates for bearers that are classified as LA, and initially it is empty.

The next step of Algorithm 1 is to estimate the amount of available PRBs for non-GBR $QCI_m$ (Procedure 1).

Procedure 1 first adjusts for control PRBs and then removes the PRB rate for all the GBR bearers in line 2. The next computation performed depends upon the scheduling schema used across QCI classes. If proportional weighted share scheduling is used, $S_m$ is set based on the minimum of the fair share requirement or the amount of traffic required for that QCI level. In the case of priority scheduling, the amount of resources for that QCI level is calculated based on the amount of resources for higher priority QCI levels and the amount of Required PRBs (RPRB) for lower priority QCIs. The minimum number of resources for lower priority QCIs is used to ensure that even lower priority QCIs are not starved.

Algorithm 1 continues by iteratively eliminating LA bearers. In each iteration, the amount of fair share of PRBs, $I_m$, is calculated. LA bearers are those that use less than $I_m$. The amount of PRBs used by LA bearers, $U_m$, is updated accordingly. After eliminating LA bearers, the amount of unclassified active bearers and the vector of PRBs of those bearers are updated, $NR_m$ and $BR_m$, respectively. Iterations continue until either no additional LA bearers are added ($N_{m,p} = NR_m$), or 0 or 1 non-LA bearers remain. Eliminated bearers are LA, and remaining bearers are classified as VA. The resulting REVA level is computed for each QCI $m$. Each slice will have at least 1 VA bearer by definition. Therefore, if 0 non-LA bearers remain, then the LA bearer with the largest number of PRBs becomes VA.

### 4.3 Computation Example

Consider a scenario where 20 UEs are served by a 10 MHz slice (50,000 PRBs/sec) with $\delta = 0.02$ and each UE has a single DL bearer at QCI 9. The PRBs over the past 1 second are assigned as follows (all units in PRBs/sec): UE1-5000, UE2-4900, UE3-4800,..., UE9-4200, UE10-3000, UE11-3000, UE12-20 each have 180 PRBs/sec. For example, in iteration 0 there are initially 20 active bearers, with $S_m = 49,000$. The fair share would be $I_m = 2,450$ PRBs, but UEs 12 to 20 use less than that amount and should be classified as LA. The amount of PRBs used by the LA bearers, $U_m$, would then be set to 1,620 with 11 remaining active bearers for iteration 1. The algorithm would then operate as shown in Table 3. After 6 iterations there is 1 VA UE, resulting in a REVA value of 4,980 PRBs/sec.

There is one UE that is using its maximal share of the PRBs and the other UEs do not require additional resources. This can be converted into throughput based on the UE’s MCS. The slice broker can then update the slice’s PRB allocation accordingly.

### 5 EXPERIMENTAL DATA COLLECTION

Due to the lack of data from service providers, we built an experimental LTE testbed to collect data and calculate the REVA metric. PRB distribution per bearer with $\leq 1$ second granularity is unavailable from deployed eNodeBs. Hence, we designed a lab LTE network with synthetic loads. The collected data is used in Sec. 6 to train forecasting models and in Sec. 7 to evaluate the impact prediction accuracy has on dynamically allocating resources. In this section we describe the experimental setup and the data collection process.

#### 5.1 LTE Testbed

The experiments were performed using the lab testbed configuration depicted in Fig. 3. The LTE eNodeB was configured with a 10 MHz bandwidth using 700 MHz wireless spectrum (LTE band 13). All UE minicomputers were connected to LTE Remote Radio Heads (RRHs) via LTE USB modems using Radio Frequency (RF) cables and splitters. An RF impairment tool with two input and two output ports was used to emulate a variety of radio conditions for two groups of UEs.

Slices were emulated by using different QCIs, with separate Access Point Names (APNs) configured for each slice. For the purpose of this paper we focused on predicting REVA of a non-GBR slice at QCI 9. We built an LTE load generator consisting of 15 UEs configured for QCI 9 and 3 UEs configured for QCI 3 (GBR). The operation of the load generator was controlled via scripts by a load generator controller connected to the UEs over Ethernet LAN. In each experiment, the non-GBR UEs were running FTP download over LTE in a continuous loop. The scripts also controlled GBR UEs to start/stop.
FTP download over LTE to emulate multiple independent patterns for the non-GBR slice.

We enhanced the LTE eNodeB scheduler by adding a thin layer to instantaneously compute REVA. The LTE eNodeB scheduler sends data digests every second in the form of the REVA metric per QCI and per slice to the Central Analytics Engine (CAE) via a dedicated out-of-band connection. The CAE performs instantaneous REVA data smoothing and further processing of the smoothed data. The predicted REVA metric is processed by the CAE policy engine to suggest an optimal policy for resource allocation. The CAE then sends this suggestion to the slice manager for enforcement.

5.2 Data Collection

The REVA metric (Section 4) is calculated at the eNodeB Scheduler. It records per bearer PRB distribution data every TTI (1 millisecond), and aggregates per bearer data into bins of time duration $t$ (e.g., $\Delta t = 1$ second or $\Delta t = 100$ milliseconds). At the end of each $t$ interval, it performs the computation described in Algorithm 1 (Section 4) and sends data digest per QCI level to the analytics engine. Our measurements indicate that executing Algorithm 1 every 1 second with approximately 500 active bearers adds less than 1% to eNodeB CPU utilization. The algorithm’s complexity is $O(k^2)$, where $k$ is the number of users. Performing preprocessing of the data at the agent instead of sending the raw scheduler data every 1 millisecond, reduces the amount of data transfer by a factor of more than 1000.

The original REVA data has a high variance over short windows due to fluctuations in the TCP client behavior. To rectify this issue we smoothed the data to reduce the noise fluctuations. Smoothing is done by looking at a window size of 10 seconds and eliminating the minimum and maximum values to remove potential outliers. The minimum remaining value of REVA is chosen. We focus on the minimum opposed to other metrics (e.g., mean, median, max) to ensure that current slices have enough resources and to prioritize current slices over incoming new slices.

In this paper, we discuss 3 time series sets collected from the testbed. [29] shows that there are temporal patterns of cellular traffic at time scales on an hourly, daily, and weekly basis. We vary

Figure 4: Experimental Data Collected (a) Set 1, (b) Set 1 autocorrelation, (c) Set 2, (d) Set 2 autocorrelation, (e) Set 3, (f) Set 3 autocorrelation.

the number of overlapping temporal patterns from 1 to 3 to emulate similar situations.

The data sets consist of the 15 non-GBR clients using FTP, and were collected for roughly 18 hours. The first data trace contained one periodic GBR client and is referenced as Set 1 and viewed in Fig. 4(a). This resulted in a simple square pattern with small variations and can be used for baseline evaluations of the prediction models. The second data set had two varying GBR clients that used FTP and is referenced as Set 2 and viewed in Fig. 4(c). Three GBR clients periodically used FTP to create a three pattern overlay profile for Set 3 (Fig. 4(e)).

The Autocorrelation Function (ACF) is used to analyze the potential predictability of a time series. ACF is the correlation of a signal with a time lag $l$ version of itself. The results of ACF give insight into how much information from the past can be used to predict future values. Figs. 4(b), 4(d), 4(f) show ACF at a time lag of $l = 0, 1, \ldots, 2000$ for REVA in Sets 1,2, and 3, respectively.

The periodic nature of ACF for Set 1 shows spikes that do not decay. It also reveals that the underlying data should have a high predictability. The ACF of Set 2 shows a strong autocorrelation for the first 100 time lags with additional significant correlation at later time lags. The medium correlation values reveal a time series that should have moderate predictability. The ACF of Set 3 is not smooth like the previous two functions. In addition, there are not as many peaks in the ACF, meaning that this data set should be the most difficult of the three to predict.

6 MACHINE LEARNING MODELS

A precise prediction model allows a broker to utilize the RAN more effectively. Therefore, there is a need for reliable and short-term prediction of the slice resource utilization. Current time series
models are designed for predicting one step into the future and do not perform well when predicting multiple steps at a time. Our design objective is to forecast REVA for the next 30 seconds with prediction intervals of 5 seconds. The forecasting function \( f(y_{t-1}) \), will provide a forecast for \( y_{t+5}, y_{t+10}, y_{t+15}, y_{t+20}, y_{t+25}, y_{t+30} \). The first 60% of each time series is used as training data, the next 20% for validation, and the final 20% for testing. We begin this section with a description of the time series models used as a baseline: ARIMA and LSTM. We then describe the architecture of the X-LSTM model designed to solve this problem.

6.1 ARIMA

One of the most popular statistical methods for time series analysis is the ARIMA model [6]. The ARIMA model assumes that future values are a linear function of previously observed values and random noise. The ARIMA model can be modeled as ARIMA\((p,d,q)\).

The data used to train the ARIMA model was the 5 second averages of the smoothed data. The ARIMA model was implemented in Python using the Statsmodel package. Optimizing the performance of the ARIMA prediction model requires tuning of the parameters \( p, q, \) and \( d \). The optimal model was selected by a grid search while varying the parameters for \( p, q, \) and \( d \) between 0 and 3. The parameter setting that gave the minimum RMSE for the validation data was \( (p,q,d) = (1,0,1) \) for both data sets. To obtain a 30 second prediction, there were 6 steps of prediction done at a time. The ARIMA model parameters are retained for every batch of multistep prediction.

6.2 LSTM

Recurrent Neural Networks (RNNs) differ from traditional Feedforward Neural Networks (FNNs) in that they contain feedback loops to allow information to propagate from previous steps. Feeding information from previous steps into the next step allows for a deep neural network architecture without computing and storing as many parameters. Therefore, these models are ideal neural network architectures for time series prediction. One of the major problems with traditional RNN is that it does not store information for long periods of time due to the vanishing gradient problem [12]. A type of RNN specifically designed to understand long term dependencies is an LSTM [13].

We implemented the LSTM neural network in Python on Keras using the Tensorflow backend. Each LSTM block contains a single recurrent layer with 100 hidden neurons and a dense output layer. For each LSTM block 80 timesteps were used. The learning rate was set to \( \alpha = 0.005 \).

For our experimental data, the first phase is done at a 30 second time level. The first LSTM block makes predictions for the average over the next 30 seconds, \( y_{t+30} \). The goal of the next phase LSTM is to make predictions for the time series of the residuals \( (y_t - y_{t+30}) \) at a higher granularity. For our evaluation, a second phase with a time scale of 5 seconds was used. The residual values for Sets 1, 2 and 3 are seen in Figs. 6(a), 6(c), and 6(e), respectively. Looking at the autocorrelations, in 6(b), 6(d), 6(f), reveals that the residual data has information from previous time lags that can be used to improve the prediction accuracy.

The first step is to make a prediction for the average over the next 30 seconds. The second LSTM phase predicts the residual \( (y_t - y_{t+30}) \) on a granularity of 5 seconds. It is used for multistep prediction in an iterative procedure for 6 steps. The 6 predicted values are then added back to the predicted 30 second average in order to predict REVA at \( y_{t+5}, y_{t+10}, y_{t+15}, y_{t+20}, y_{t+25}, y_{t+30} \).

The number of phases and the time scale of each phase are the additional tuning parameters for X-LSTM neural network. While each of our data sets was only collected for roughly one day, this method can be extended to longer data sets by including additional phases. Typically networks exhibit time patterns due to natural seasonal, weekly, daily, and hourly traffic variations [29]. Data can be reduced by summarizing the data into lower granularity. For example, instead of storing per second data, data can be summarized into hourly, and daily averages. Data can be stored on the granularity.

![Figure 5: An example of X-LSTM machine learning architecture used for the experimentally collected data. This X-LSTM architecture contains two phases, one at a time scale of 30 seconds and the next at 5 seconds.](image)
of seconds for the past day, per hour bases for the past month, and
daily values for the past year. This can reduce the amount of data
needed for prediction by over 99%.

7 EVALUATION

In this section we briefly discuss accuracy measures used for evalua-
tion. We then analyze the prediction accuracy of each proposed
prediction model. We continue with a description of a slice pro-
visioning algorithm and design of a cost function based on the
amount of over-provisioned PRBs and SLA violations. We conclude
by showing how the algorithm exploits the prediction models to
minimize cost for service providers.

7.1 Prediction Results

Accuracy measures used for comparison between the different time
series methods include: Root Mean Square Error (RMSE), Mean
Absolute Error (MAE), and Mean Absolute Percentage Error (MAPE).
The true output value is \( y_t \) and the predicted value is \( \hat{y}_t \) for each
time \( t \). The accuracy of each proposed method is measured by
splitting the experimental data collected into three components:
training, validation, and test. The training data is used to tune the
machine learning parameters for each method, while the validation
data determines the best parameters for each method. The test data is
used to compare the performance of each model.

The prediction results on test data for Sets 1, 2, and 3 can be
seen in Fig. 7, along with the resulting error measures in Fig. 8.
Four different prediction models are used for comparison. The first
model predicts a one step 30 second mean of the time series using
a vanilla LSTM model (referred to as 30 Second LSTM). The second
model is determined by the X-LSTM model. The third prediction
model uses a multistep LSTM to determine the congestion over the
next 30 seconds in a granularity of 5 seconds. The fourth model
uses a multistep ARIMA model to predict the congestion over the
next 30 seconds in a granularity of 5 seconds.

Set 1 follows close to a periodic square wave with numerous time
steps in between. The results show that all the models are able to
follow the square wave relatively accurately. The multistep LSTM
and ARIMA models have a more difficult time and incorporate
errors from earlier steps leading to extrapolation. The prediction of
the 30 Second LSTM is better able to determine the square pattern,
as it has less information to store from previous periods and is only
making a one step prediction. The resulting error measurements
show that even on the simpler square wave pattern, our X-LSTM
model outperforms the other models. It is able to learn meaningful
information from the residual between phase 1 and phase 2.

The second best model, 30 Second LSTM, gives a MAPE of 7.68%
and an RMSE of 312, while the X-LSTM model gives a MAPE of
6.93% and an RMSE of 256, resulting in a 10% and 18% improvement,
respectively.

Set 2 has a more complex time series which has two overlapping
periodic events. All models in this data set have a higher error rate
than in Set 1. The multistep LSTM has a difficult time incorporating
the many variations it learned from the two overlapping patterns.
When creating multistep predictions the errors continue in a posi-
tive or negative direction causing a sawtooth pattern. The multistep
ARIMA model has a higher accuracy for Set 2 than the multistep
LSTM model. Phase 1 of the X-LSTM model is able to track the
model, however it is often slow to update when there are large
changes. The X-LSTM model is able to incorporate the residual
difference and make an improvement on the phase 1 predictions.
The additional benefit of X-LSTM is that when the last phase LSTM
needs to predict multiple steps, the bias of the prediction is reduced.

The second best model is again, 30 Second LSTM, which has
a MAPE of 8.11% and an RMSE of 353. The X-LSTM model has a
MAPE of 6.36% and an RMSE of 304, resulting in a 22% and 14%
improvement.

The most challenging set to model due to its three overlapping
periodic patterns is Set 3. The result is a lower level accuracy across
all models. There is a strong residual component that has high ACF
values. This allows the second phase of X-LSTM to improve upon
the accuracy of the first phase and thus reduce the prediction error.
The multistep ARIMA and LSTM models in this situation perform
significantly worse than X-LSTM.

The MAPE of the multistep LSTM is 13.07% and for the LSTM
predicting 30 second averages it is 13.03%. The X-LSTM is able to
outperform these models by 31% with a MAPE of 8.99%.

7.2 Slice Allocation

We evaluate the impact that forecasting accuracy has on slice re-
source utilization. The REVA metric allows for easy adjustment of
slice provisioning by using the difference between REVA and the
SLA. The value can be used to estimate the amount of additional
resources the slice should be allocated to satisfy the SLA or the
amount of PRBs that should be removed while still satisfying the
SLA.
We assume that the forecasting error $\epsilon_t$ in (1) is normally distributed with a standard deviation of $\sigma$ and a mean of 0, since the exact distribution of the empirical error is unavailable. This gives a Gaussian prior for the prediction model:

$$y_t \sim N(\hat{y}_t, \sigma^2)$$ (3)

We assume that assigning too few resources results in an SLA violation and incurs a penalty with cost $k$. Conversely, when forecasting a higher REVA value than predicted results in extra PRBs available to VA users that can otherwise be allocated to other slices. We use a one sided prediction interval $h$ that determines the bound that should be used when assigning resources for forecast model $f$. We define the cost function, $\Gamma$ as:

$$\Gamma(y_t) = \begin{cases} 
  k, & \text{if } \hat{y}_t + h > y_t \\
  y_t - h - \hat{y}_t, & \text{if } \hat{y}_t + h \leq y_t 
\end{cases}$$}

Accordingly, we formulate the following optimization problem to obtain the optimal $h$ such that the system cost is minimized, given $\sigma, \hat{y}_t$, and $k$:

$$\min_h \quad k(\hat{y}_t + h - y_t)^+ + (y_t - \hat{y}_t - h)(y_t - \hat{y}_t - h)^+$$ (4)

Taking the expected value of the cost metric simplifies the optimization problem to:

$$\min_h \quad (k + \hat{y}_t)(\Phi(\frac{h}{\sigma})) - h(1 - \Phi(\frac{h}{\sigma})) + \sigma(\frac{\phi(\frac{h}{\sigma})}{1 - \Phi(\frac{h}{\sigma})})$$ (5)

Where $\Phi(z)$ is the CDF of the standard normal random variable $Z$, and $\phi(z)$ is the PDF of the standard normal random variable $Z$.

### 7.3 Slice Allocation Efficiency

For each forecasting model, the optimal $h$ is used for each $\hat{y}_t$ and the cost $\Gamma(y_t)$ is calculated. Fig. 9 shows the average $\Gamma$ per forecasting point versus the SLA violation cost $k$. As $k$ increases the optimization function weighs SLA violations more. Therefore, the number of standard deviations away the prediction interval $h$ is increases. The optimization function causes the probability of missing the SLA to decrease from 10% to 1% (depending on the forecasting algorithm’s $\sigma, \hat{y}_t$, and the cost $k$).

In data set 1, with $k = 0$, Multistep LSTM slightly outperforms X-LSTM by less than 1%. For every other SLA cost $k$ the X-LSTM outperforms the other forecasting algorithms. For $k \geq 5000$, X-LSTM provides more than 15%, 40%, and 15% reduction in average system cost over 30 Second LSTM, Multistep ARIMA, and Multistep...
We thank the anonymous reviewers for helpful comments and X-LSTM outperformed the other time series models by X-LSTM, for REVA. We showed that X-LSTM provides a higher accuracy of X-LSTM on real world traces, and develop slice sets, the improvement of X-LSTM over ARIMA and LSTM increases. semi-periodic components increases from one to three in the test sets, respectively. As the number of independent conditions information. An experimental LTE testbed was developed to collect REVA for time series analysis. This data was then used to develop and evaluate the accuracy of time series prediction models.

We proposed and evaluated a new time series prediction model, X-LSTM, for REVA. We showed that X-LSTM provides a higher degree of prediction accuracy with a MAPE of 7.68%, 6.36%, 8.99% for Sets 1, 2, and 3, respectively. As the number of independent semi-periodic components increases from one to three in the test sets, the improvement of X-LSTM over ARIMA and LSTM increases. X-LSTM outperformed the other time series models by 10%, 22%, and 31%, respectively. In addition, X-LSTM results in more than 10% cost reduction per slice. In future work, we will further test the accuracy of X-LSTM on real world traces, and develop slice admission control algorithms for the broker, based on REVA.

8 CONCLUSION

In order to effectively provision resources and make admission control decisions, the 5G RAN slice broker requires an accurate prediction of the required resources in a slice. We defined a new metric, REVA, that precisely measures the amount of PRBs that the RAN scheduler can allocate to VA bearers. This allows for direct derivation of wireless link throughput when coupled with channel conditions information. An experimental LTE testbed was developed to collect REVA for time series analysis. This data was then used to develop and evaluate the accuracy of time series prediction models.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for helpful comments and suggestions. This work was supported in part by NSF grants CNS-1650685, and DGE 16-44869.

REFERENCES


[29] Huadong Wang, Fengli Xu, Yong Li, Fengyu Zhang, and Deperg Juin. 2015. Understanding mobile traffic patterns of large scale cellular towers in urban environment. In Proc. of ACM IMC.
[33] Xiaofeng Xie, Xinyu Zhang, Swaran Kumar, and Li Erran Li. 2015. psStream: Physical layer informed adaptive video streaming over LTE. In Proc. of ACM Mobicom.