Proactive Clustering of Base Stations in 5G C-RAN using Cellular Traffic Prediction

Mehul Sharma, *Student Member, IEEE,* Ujjwal Pawar, *Student Member, IEEE,*
Antony Franklin A., *Senior Member, IEEE,* Bheemarjuna Reddy Tamma, *Senior Member, IEEE*

***Abstract*—The rapid growth in mobile network traffic and dynamic user mobility patterns have propelled network operators toward the Cloud-Radio Access Network (C-RAN) to reduce operational costs and improve service quality. C-RAN handles the traffic and mobility issues in a centralized manner by segregating the central units (CUs) from the distributed units (DUs) in a shared CU pool. The ability of C-RAN to map multiple DUs to the same CU allows optimal coverage with high multiplexing gains, using the least number of CUs. However, dynamically mapping DUs to CUs is not trivial since the network traffic and mobility patterns are difficult to predict. This paper presents a two-phase framework for an optimal city-wide C-RAN network. In the first phase, we propose to use the *ConvLSTM* model, which simultaneously learns the hidden spatial and temporal dependencies in a real-world dataset and makes accurate traffic forecasts for a future duration of time. In the second phase, we use the predicted traffic from the first phase to develop a proactive optimal DU-CU clustering scheme that is cost-effective and meets quality objectives. We first formulate an optimization problem, and later, to reduce the computational complexity of the optimization, we propose a lightweight heuristic algorithm. Finally, we evaluate the performance of our prediction model and the mapping scheme using a two-month real-world mobile network dataset of Milan, Italy. Based on simulation results of phase one, we observe the *ConvLSTM* model, when deployed in a C-RAN architecture, outperforms existing state-of-the-art prediction models with up to 26% better RMSE (Root Mean Square Error) and up to 36% better MAPE (Mean Absolute Percentage Error) values. Similarly, in phase two, our simulation results show that compared to reactive threshold-based clustering, proactive clustering can reduce the average number of active CU servers by up to 18% every 10 minutes without overloading.**

***Index Terms*—C-RAN, Traffic Prediction, CU-DU Mapping**

I. Introduction

According to Cisco's predictions, total Internet traffic has experienced dramatic growth in the past two decades. Global IP traffic is expected to reach 4.8 *Zettabytes* (ZB) per year by 2022 [1]. To cope with ever-increasing growth in Internet traffic and fast-growing mobile subscribers, network operators are adding extra processing power to their existing base stations or deploying additional base stations to expand their network coverage. Consequently, the capital expenditure (CAPEX) and the operational expenditure (OPEX) are also increasing significantly. Meanwhile, due to many co-located base stations, network performance and Quality of Service (QoS) are challenging to ensure due to several factors such as interference and handovers. Therefore, operators must adapt to new and promising wireless technologies to provide high network performance and reduce the CAPEX and OPEX.

Cloud Radio Access Networks (C-RAN) emerges as a competent and novel architecture to address the previously mentioned challenges. In C-RAN, to cut down the maintenance and operational costs, the existing base station is decoupled into remote Distributed Units (DUs) and computationally heavy baseband processing units called Central Units (CUs). The DUs are responsible for radio communications and are present at the cell sites, while CUs are responsible for signal and data processing and are moved to centralized cloud/data centers called CU pools. Both DU and CU communicate via a high-speed interface called a fronthaul interface. The recent developments in cloud computing and virtualization techniques can further improve the CU pool's resource utilization and energy efficiency. Additionally, the isolation flexibility of the virtual clouds allows us to map multiple DUs to a single CU. A *m:1* mapping of DUs to CUs provides higher energy savings compared to a *1:1* mapping as several Virtual Machines (VMs) can be switched off/on based on the instantaneous load experienced by the DUs. To maximize the benefits of C-RAN architecture, it is imperative to develop a DU-CU mapping scheme that minimizes deployment and operational costs while maximizing service quality. The mapping scheme that partitions DUs into separate clusters and assigns CUs to each cluster will be deemed cost-effective and QoS aware if it achieves the following objectives [2].

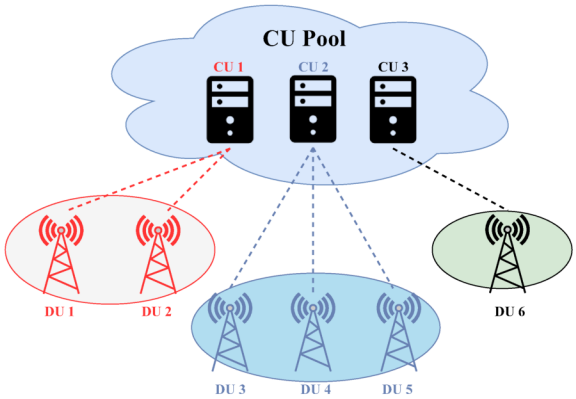

Fig. 1. An illustrative example of a C-RAN architecture.

1) The number of active clusters is as few as possible.
2) For each cluster, to minimize the fragmentation loss, the aggregated capacity of the load generated by the DUs should be almost equal to the maximum rated capacity of the CU serving that cluster.
3) Change in traffic may sometimes overload or underutilize the VMs, and this causes the remapping of CU-DU which takes both computational and radio resources. Frequent

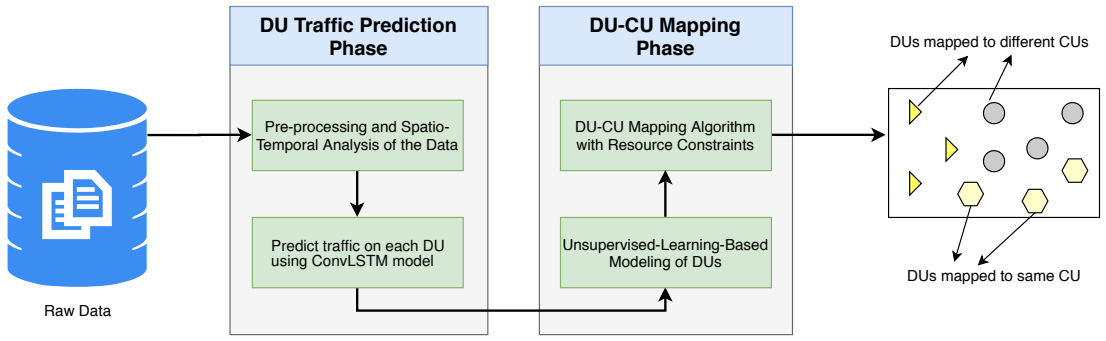

Fig. 2. Framework Overview

remapping of CU-DU must be minimized and should happen in a predefined mapping interval.

Objectives 1 and 2 can be achieved by formulating an efficient DU-CU mapping scheme. But for Objective 3, accurate modeling and prediction of DU traffic are required. By knowing the future traffic information, the mapping scheme can map the DUs to CU by considering current and future traffic requirements, minimizing the frequent remapping. Another way to achieve Objective 3 is reserving the resources, so that server assigned to clusters never gets overloaded. In this paper, we propose a predictive clustering framework that uses traffic prediction for clustering, due to which the servers never get overloaded during the assigned remapping interval without any additional resource reservation. This reduces the number of active servers at the same time and ensures that the assigned servers are not overloaded.

II. Overview of Proactive Clustering Framework

In this section, we present the overview of our proposed proactive clustering framework, as shown in Fig. 2. The first phase is *Traffic Prediction Phase*, where we process the raw data and analyze the spatial and temporal correlations using big-data techniques, which includes studying different types of cell sites based on locality and day-night traffic differences. We use the deep-learning-based *ConvLSTM* model to accurately predict the DU traffic by learning the spatial and temporal dependencies simultaneously. The second phase is *DU-CU Mapping Phase*, where we take the predicted traffic from $Phase$ 1 as an input and then formulate an unsupervised-learning-based mapping algorithm to accurately map DUs to CUs, considering all the global CU pool and resource constraints in mind. We represent the location of DUs as a set of data points on a graph, marking instantaneous DU load as the weight of each graph node. The final output of the framework is a set of clusters represented by different colors and markers, as shown in Fig. 2, where each cluster is served by its CU. The proposed framework ensures low operational costs and high service quality by minimizing the total number of clusters and mapping DUs with high handover probabilities to the same CU, respectively. In sections III and IV, we discuss each phase in more detail.

III. DU Traffic Prediction Phase

The objective of this phase is to predict the traffic on each DU by using deep learning-based algorithms. Traffic prediction is not trivial due to the presence of high temporal and spatial dependencies among the base stations. Traditional prediction methods use time series models to predict future traffic, which fails to capture the temporal and spatial correlations in the data, thus affecting the accurate prediction of network traffic. Therefore, in this section, we propose to use a deep learning-based *ConvLSTM* model in C-RAN architecture to capture the hidden temporal and spatial dependencies using a real-world dataset and predict future traffic accurately.

1) ConvLSTM Model: ConvLSTM model is a combination of both - CNN (Convolutional Neural Networks) and LSTM (Long Short Term Memory Networks). CNN is a deep learning architecture, most commonly used in visual computing, such as image processing, to capture the spatial dependencies [3]. However, it fails to completely learn the temporal dependencies in the data due to a lack of memory states that can remember previous time intervals. LSTM on the other hand is a Recurrent Neural Networks (RNN) based architecture capable of learning long-term temporal dependencies [4]. LSTMs can connect previous cell information to the present cell using memory states in the form of feedback loops, thus allowing data to persist. The default behavior of LSTM to remember the previous time intervals makes it well suited for mobile traffic forecasting problems to learn the temporal behavior of the data. LSTMs can determine the temporal correlation in the data, but they fail to learn the spatial correlation with neighboring cells, which is a key characteristic of multi-cell traffic data. The *ConvLSTM* is a variant of LSTM containing a convolution operation inside the LSTM architecture. In such an architecture, LSTM network learns the temporal features, while CNN learns the spatial features, resulting in a better prediction accuracy. The architecture of *ConvLSTM* model is shown in Fig. 3.

2) Working of ConvLSTM Model: The experimental data is divided into two parts, training data and test data. The training data is used to train the model and the test data is used to evaluate the accuracy of the trained model. Out of a two-month dataset available to us, seven weeks of data is used for training the *ConvLSTM* model and last week's data is used to test the performance of the trained model. The input is in the form of a 5D tensor with shape (*samples, timesteps, channels, rows, cols*) where *samples* represents the total count of input samples sent to the LSTM network, *timesteps* represents the total num-

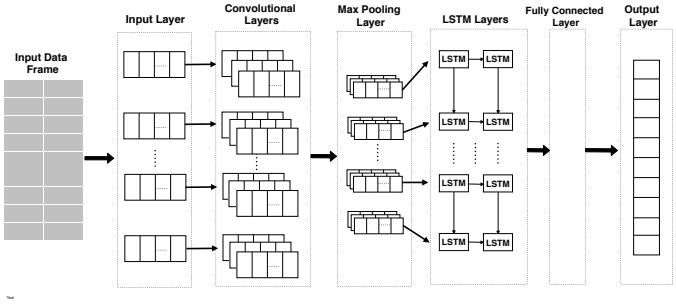

Fig. 3. Architecture of *ConvLSTM* Model in this paper

ber of previous time intervals based on which prediction will be made, and *features* represents the number of dimensions, i.e., the number of DUs in our case, *rows, columns* represents the height and width of the convolutional input matrix, and *channels* represents the depth of each sample point. The original input matrix D is restructured to 5D tensor shape and fed to the *ConvLSTM* model using the *input layer*. The data then passes through *convolutional layers*, where multiple filters (or kernels) of different heights and widths slide over the input data to create new feature maps. The obtained feature maps further pass through a *max pooling layer*, where we select the maximum element from a kernel region such that the feature maps contain only the most prominent features. By this stage, the model has already captured the spatial dependencies in the data using CNN feature maps. There onwards, the output obtained from the pooling layers is given as input to LSTM networks where multiple hidden *LSTM layers* learn the long-term temporal dependencies from the sequence. The output from LSTM cells is passed through a *fully connected layer* to reduce the dimensions and flatten the sequence data. Finally, the *output layer* receives the final predicted values from the model.

IV. DU-CU Mapping phase

The objective of this phase is to design an efficient DU-CU mapping scheme that minimizes the number of active CUs and improves the service quality, given the predicted traffic and pool constraints. We formulate the problem as a *bin-packing problem* [5], where the objective is to pack items (DU loads) with different weights into a finite set of bins (CUs) such that the minimum number of bins are used. In computational complexity, the *bin-packing* problem is already proven to be NP-hard [6]. Hence, to reduce the computational complexity of the optimization problem, we devise a near-optimal heuristic approach efficiently to map DUs to CUs which is inspired by our previous work [2]. We use an *unsupervised-learning* based approach to divide the set of DUs into several clusters based on a *assignment_score*, where each cluster is served by its own CU, considering the CU pool resource constraints.

A. Optimization Model

Consider $\mathcal{D} = \{d_1, d_2, \ldots, d_n\}$ as the finite set of DUs and $\mathcal{C} = \{c_1, c_2, \ldots, c_m\}$ as the finite set of CUs in CU pool. Since we can map multiple DUs to a same CU, we have $\mid \mathcal{D} \mid \geq \mid \mathcal{C} \mid$. Let, $C^{'}$ be the finite capacity of all the CUs. The load on each DU $d \in \mathcal{D}$ is denoted by l_d. Further, the association between DU $d \in \mathcal{D}$ and CU $c \in \mathcal{C}$ can be represented as a binary variable.

$$x_{dc} = \begin{cases} 1, & \text{if DU } d \text{ is mapped to CU } c \\ 0, & \text{otherwise} \end{cases}$$

Since all the CUs may not be active at the same time, we define a binary variable.

$$y_c = \begin{cases} 1, & \text{if CU } c \text{ is active} \\ 0, & \text{otherwise} \end{cases}$$

1) Handover Probability: Let, $\mathcal{G}$ be a distance matrix where each entry $\mathcal{G}_{ij}$ represents the euclidean distance between DU d_i and DU d_j. We define the normalized distance $\mathcal{G}'(i)$ for each DU d_i from all other DUs d_j s.t. j$\in \mathcal{G}_{i,*}$,

$$\mathcal{G}'(i) = (max(\mathcal{G}_{i,*}) + 1) - \mathcal{G}_{ij} \tag{1}$$

The handover probability P_{ij} of a user moving from the coverage of DU d_i to the coverage of d_j s.t. j$\in \mathcal{G}_{i,*}$ can be obtained as,

$$P_{ij} = \frac{\mathcal{G}'(i)}{\sum_{j=1}^{\mathcal{D}} \mathcal{G}'(i)} \tag{2}$$

To ensure better service quality, we map the DUs with higher handover probabilities to the same CU. Let, at timestamp t, UE1 and UE2 are served by DU d_1 and DU d_2, respectively. Both d_1 and d_2 are mapped to the same CU, *i.e.,* CU1. Further, we assume that handover probability $P_{12} > P_{13}$ and $P_{23} > P_{21}$. Therefore, at timestamp $t+1$, UE1 will move from the coverage of DU d_1 to the coverage of DU d_2. Similarly, UE2 will move to the coverage of DU d_3. In such a scenario, UE1 will not encounter a handover as both d_1 and d_2 are mapped to the same CU, whereas UE2 will encounter a handover as it is moving to DU d_3 which is mapped to a different CU. However, in a traditional mobile network, both UE1 and UE2 will encounter the handover.

2) Problem Formulation: The notation used in formulation of optimization are shown in Table I.

Objective Function:

$$\min_{y,z} : \underbrace{\sum_{c=1}^{\mathcal{C}} y_c}_{\text{(A)}} \times \underbrace{\frac{1}{\sum_{\forall i,j \in c} P_{ij}}}_{\text{(B)}}) \tag{3}$$

Constraints:

$$\sum_{d=1}^{\mathcal{D}} (x_{dc} \times l_d) \leq (C^{'} \times y_c), \quad \forall c \in \mathcal{C} \tag{4}$$

$$\sum_{c=1}^{\mathcal{C}} x_{dc} = 1, \quad \forall d \in \mathcal{D} \tag{5}$$

$$x_{dc} \in \{0,1\}, \quad \forall d \in \mathcal{D}, \forall c \in \mathcal{C} \tag{6}$$

$$y_c \in \{0,1\}, \quad \forall c \in \mathcal{C} \tag{7}$$

Term A minimizes the total number of active CUs in the CU pool. Term B ensures the service quality by mapping DUs with higher handover probability to the same CU. Eqn. (4) ensures that the sum of DU loads mapped to a CU should not exceed the CU's total capacity. Eqn. (5) ensures that a DU is assigned to only one CU. Eqns. (6-7) indicate that x_{dc} and y_c are boolean variables.

B. Proposed Heuristic Algorithm

We model the DU-CU mapping problem as an *unsupervised learning* problem, where we represent the geographical location of DUs as a set of data points on a graph. We then divide the data points into multiple clusters of DUs, where each cluster is served by a dedicated CU, considering the resource constraints in the CU pool.

The description of the heuristic algorithm is as follows:
Let M be the matrix containing coordinates of the DUs, P be the matrix representing handover probability between DUs, and L be a vector containing predicted load on all the DUs. For simplicity, we assume that the handover probability P_{ij} is already computed by the mobile operator based on the geographical location of DUs. The notation used in designing the heuristic algorithm is listed in Table II. The key steps involved in the heuristic algorithm are listed below.

1) Calculate the minimum number of active servers required (Line 1). Let, L_i be the predicted load on DU i and C' be the finite server capacity. We define K as the minimum number of CUs required,

$$K = \frac{\sum_{i=1}^{D} L_i}{C'} \tag{8}$$

TABLE I
NOTATION USED IN THE OPTIMIZATION MODEL

Notation	Definition
l_d	Compute load at DU $d \in \mathcal{D}$
$\mathcal{D}$	Finite set of DUs
$\mathcal{C}$	Finite set of CUs
C'	Capacity of CU $c \in \mathcal{C}$
y_c	1 if CU $c \in \mathcal{C}$ is active; otherwise 0
x_{dc}	1 if DU d is mapped to CU c; otherwise 0
P_{ij}	Handover Probability of a user moving from DU d_i to d_j

TABLE II
NOTATION USED IN THE HEURISTIC ALGORITHM

Notation	Definition
M	Matrix with coordinates of all the DUs
L	Vector of predicted load on all the DUs
C'	Capacity of CU $c \in \mathcal{C}$
$\mathcal{D}$	Finite set of DUs
$\mathcal{C}$	Finite set of CUs
P_{ij}	Probability of a user moving from DU d_i to d_j
K	Number of CUs
$assignment\_score_{ij}$	Assignment score of a DU i with a centroid j
X_i, Y_i	Centroid coordinates of cluster i

2) After selecting the minimum number of servers required, we identify the K initial cluster centroids based on decreasing order of load (Line 2). A cluster centroid is defined as the average of all the points mapped to that cluster.
3) Sort the DU loads in decreasing order so that the larger loads are assigned to the CUs first, and the DUs with smaller loads can be packed easily in other CUs (Line 3). If smaller DU loads are mapped first, the number of active servers may not be optimal because the larger DU loads may lead to the formation of additional servers.
4) For each unassigned DU, we calculate the *assignment_score* of that DU with all the cluster centroids (Lines 6-8). The *assignment_score* between a DU $i \in \mathcal{D}$ and a cluster centroid j can be calculated as,

$$assignment\_score_{ij} = L_i * P_{ij} \tag{9}$$

where L_i is the load on DU i and P_ij is the handover probability between DU i and cluster j.
The *assignment_score* is a combination of both load and handover probability. If DUs are mapped based on the load alone, ensuring service quality is challenging as we are not mapping DUs with a higher handover probability to the same CU.
5) Assign DU to a cluster with highest *assignment_score* (Lines 9-11). If it cannot be assigned, map DU to cluster with next nearest *assignment_score* (Lines 12-15).
6) If none of clusters can can accommodate the DU, initialize a new cluster and assign the DU to that cluster (Lines 16-19).
7) Once all the DU loads are mapped to the CUs, we will calculate the new centroids for all the clusters (Line 21). Let, $(x_1, y_1), (x_2, y_2) \ldots (x_n, y_n)$ be the coordinates of DUs mapped to a cluster. For each cluster i, we calculate the centroid (X_i, Y_i) as,

$$X_i = \frac{\sum_{i=1}^{n} x_i}{|n|} \quad \text{and} \quad Y_i = \frac{\sum_{i=1}^{n} y_i}{|n|} \tag{10}$$

8) Repeat steps 1 to 7 until centroids and clusters are changing.

1) Time Complexity Analysis of the Heuristic Algorithm: Consider $|C|$ as the number of CUs and $|D|$ as the number of DU loads that we want to assign. For simplicity, we assume the handover probabilities are pre-computed by the operators. We can calculate the total number of initial K clusters required in $O(1)$ time using Eqn. 8. To minimize the number of active clusters formed, we need to sort the DU in decreasing order of load. This can be done in $O(|D|log|D|)$ in the worst case. In each iteration, we calculate the *assignment_score* of all the DUs with K clusters (where $K << D$) in $O(|DK|)$ time. For each DU, we need to find the cluster with maximum *assignment_score*. This requires an additional time of $O(|DK|)$. Lastly, the centroids can be updated in $O(1)$ using Eqn. 10. Combining all the times, the overall time complexity of the

Algorithm 1: Proposed DU-CU Mapping Algorithm.

Input : Coordinate Matrix (M), Handover Probability matrix (P), Predicted Load (L), Server capacity (C')

Output: Best possible DU-CU mapping at time epoch $t+1$.

$K = \frac{\sum_{i=1}^{D} L_i}{C'}$;
Initialise K cluster centroids from M in decreasing order of load and store in a list T;
Sort the DUs in decreasing order of load L_i;
while *not converged* **do**
  for *each DU* $i \in \mathcal{D}$ **do**
    for *each centroid* j *in* T **do**
      $assignment\_score_{ij} = L_i * P_{ij}$;
    end
    if *DU* i *can be assigned to cluster* j *with max(assignment_score)* **then**
      Assign DU i to cluster j;
      $C'_j = C'_j - L_i$;
    else
      Assign DU i to next best cluster j ;
      $C'_j = C'_j - L_i$;
    end
    if *DU* i *cannot be assigned to any cluster* j **then**
      Create a new cluster n and assign i to n;
      Add n to T;
    end
  end
  Update the centroids;
end

proposed heuristic algorithm in the worst case is $O(|D|log|D|)$ and thus can be solved in polynomial time.

V. Performance Evaluation

A. Evaluation of Prediction Phase

1) Dataset Description: In this work, we use one of the richest telecom datasets ever released by Telecom Italia in 2015 [7]. Originally, the dataset was released as a part of *"Big Data Challenge"* organized by the Telecom Italia. Later, the dataset was made publically available to the research communities. The dataset contains the time series data of cellular traffic observed after every 10-minute intervals, for the city of Milan, collected for two months, *i.e.*, 1^{st} November 2014 - 31^{st} December 2014. The city of Milan is divided into a grid of 100x100 squares (235m x 235m each), and each square is referred to as a "cell." The dataset is anonymized and is aggregated in terms of Call Detail Records (CDRs). Whenever a user engages in a call, SMS, or internet, a new CDR is generated. For each cell in the dataset, we know,

- *Square id:* unique identification of each cell.
- *Country code:* code of country whose data is collected.

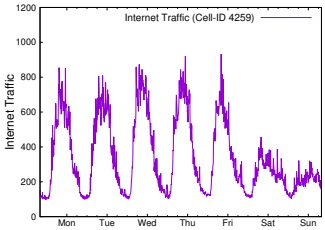

Fig. 4. Variation in traffic with time for Cell id 4259

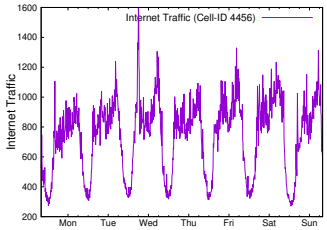

Fig. 5. Variation in traffic with time for Cell id 4456

- *Time Interval:* start interval time in milliseconds.
- *SMS-in and SMS-out:* number of SMS received or sent from the cell during the time interval.
- *Call-in and Call-out:* number of calls received or made from the cell during the time interval.
- *Internet traffic:* number of internet traffic CDRs generated inside a cell during the time interval.

As a large portion of data is composed of only Internet traffic, we focus only on the Internet traffic prediction in this work.

2) Key Observations: Temporal Domain: Fig. 4 shows the temporal variation of Internet traffic throughout a week for one of the famous universities in the city of Milan, *i.e.*, *Bocconi University*, denoted by square id 4259. It can be seen from the figure that the traffic follows a strong diurnal pattern; e.g., the traffic volume during weekdays is much higher than on the weekends. Similarly, the traffic generated during office hours is higher than in evening hours when people start returning to their homes from the university.

Spatial Domain: To understand the spatial behavior of data, we compare the traffic of two contrasting areas in the city of Milan. The first one is the same university mentioned above, *i.e.*, *Bocconi University* with cell id 4259 and the second one is the famous nightlife places in the city, *Navigli district*, denoted by square id 4456. Figs. 4 and 5 shows that both the areas observe spatially different traffic patterns. As expected, traffic volume starts to increase at *Navigli District* during the evening as compared to *Boconni*, where traffic starts to drop off during the evening. Also, the total traffic generated during a day is much higher at *Navigli District* as it is located in the center of the city. It can also be observed that traffic at *Navigli District* remains the same throughout the week, whereas the traffic at *Bocconi* drops during weekends. Such a spatio-temporal behavior among different cells located at different places makes it very challenging to predict future traffic correctly. Therefore, efficient techniques must be devised to capture the spatial and temporal behavior of traffic and enhance the prediction engine.

3) Data Pre-processing: In the dataset provided by Telecom Italia, the data of each day was stored in a separate file. Each file contained traffic information (square id, country code, SMS-in/out, calls-in/out, and Internet) for all the square ids on that particular day in intervals of 10 minutes. As mentioned earlier in §V-A, we consider only Internet traffic prediction in this work. Hence, all the files were parsed, and the Internet traffic of all the cells is extracted and stored

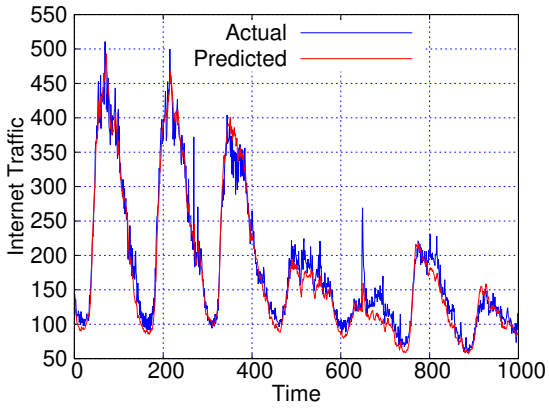

Fig. 6. Predicted Traffic for cell id 4259

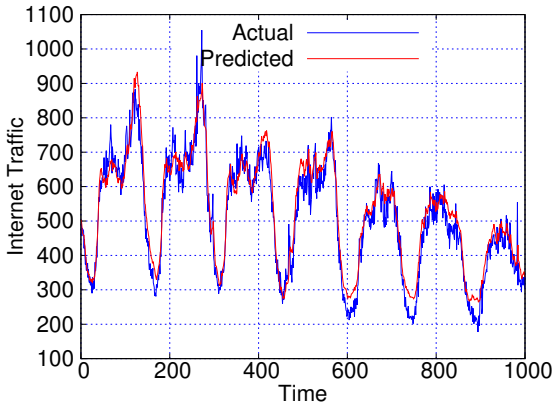

Fig. 7. Predicted Traffic for cell id 4456

in a single data frame *D* using *pandas* library of python. Data frame is a matrix representation of the data with *T*C* dimensions where *t* represents the total time steps separated by 10-minute intervals for two months, and *C* represents the total number of DUs, *i.e.*, cells. Each entry D_{tc} represents the internet traffic of c^{th} cell at t^{th} timestep.

4) Experimental Setup : The experimental data is divided into two parts, training data and testing data. Training data is used to train the model and testing data is used to evaluate the accuracy of the trained model. Out of a two-month dataset available to us, seven weeks of data is used for training the *ConvLSTM* model and last week's data is used to test the performance of the trained model. We convert the input data frame *D* into a 5D tensor of shape *(samples, time_steps, channels, rows, cols)*. For training the model, we use 6626 *samples* where each sample contains 144 previous *time_steps*. The value of *channels* and *rows* is considered as one since we convolve each sample point independently. The total number of DUs or the number of *cols* in our model is 100.

5) Parameters and Model Training: Below we discuss all the key parameters used in our *ConvLSTM* model.

- *Filters:* The number of filters used in the convolution process.
- *Kernel Size:* It defines the height and width of the convolution window.
- *Activation Function:* It is a function that is used for non-linear transformation of the input data.
- *Optimizer:* It changes the weights and learning rate of a neural network such that the loss functional is minimized.

We use the famous *tensflow* library to train all the three models for 100 *epochs* using sequential *ConvLSTM* model with 100 filters and a kernel size of *(1*2)*. To transform the input sequence to a non-linear space, we use *Relu* as the activation function. Further, to optimize the loss function we use the famous *adam* optimizer.

6) Baseline Methods: We compare the performance of the *ConvLSTM* model with the following two baseline models,

1) Convolutional Neural Networks (CNNs)
2) Long Short Term Memory Networks (LSTMs)

7) Evaluation Metrics: We evaluate the performance of cellular traffic prediction based on two metrics: Mean Absolute Percentage Error (MAPE) and Root Mean Square Error (RMSE).

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{A_i - P_i}{A_i}\right| \tag{11}$$

$$RMSE = \sqrt{\frac{1}{n}\Sigma_{i=1}^{n}\Big(A_i - P_i\Big)^2} \tag{12}$$

where n is the total number of sample points, A_i is the actual value, and P_i is the predicted value.

8) Performance Analysis: The *ConvLSTM* model achieves an RMSE of 50.56 which is 10% and 26% less than the baseline methods LSTM and CNN, respectively. Further, when compared using MAPE, the *ConvLSTM* model achieves a prediction error of 9.86 which again outperforms the baseline LSTM and CNN by 4% and 36%, respectively. The main focus of the LSTM model is to remember the previous information by its design and hence it fails to capture the spatial dependency while CNN fails to learn the temporal dependencies due to a lack of memory states that can remember previous time intervals. In the case of *ConvLSTM* model, we use a combination of both CNN and RNN which simultaneously learns both temporal and spatial dependency, and hence a better prediction accuracy is achieved.

Figs. 6 and 7 shows the forecasting results for the cell IDs 4259 and 4456, respectively. The selected areas show very different behavioral patterns. As expected, Navigli district has high Internet traffic during the night and Bocconi has less Internet traffic during the weekends. The results clearly show that the *ConvLSTM* model can capture the hidden correlations from the cells with different behavioral patterns and predict future traffic with high accuracy.

B. Evaluation of Mapping Phase

In this section, we evaluate the performance of our proposed heuristic algorithm with the optimization model.

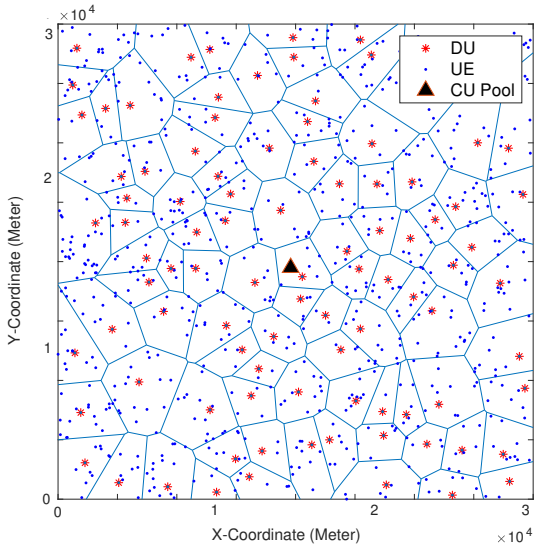

Fig. 8. Spatial distribution of 500 UEs and 100 DUs as per MHCPP II.

1) Experimental Setup: At time instance t, the predicted traffic on all the DUs is given as input to *mapping phase* so that an efficient mapping of DUs can be obtained at the next time interval. The predicted load is normalized in a range of [0 1] and the fixed capacity of each CU is considered as 1. We represent DU coordinates as a set of data points over a specific geographical region, maintaining their spatial dependency. Since the location of DUs is not available in the dataset, we use tools from point process models and stochastic geometry to geographically distribute the DUs. As an example, we highlight the distribution of 100 DUs and 500 UEs over a $30km * 30km$ region in Fig. 8. We use the Matern Hard Core Point Process type II (MHCPP II) and Poisson Point Process (PPP) model for the deployment of DUs and UEs, respectively. Further, to highlight the coverage regions of a DU, we use Voronoi Tessellation. We assume there is a *1:1* mapping between the DUs considered in the prediction phase and the DUs generated here using MHCPP II. In real-world scenarios, however, operators will have the exact location of the DUs in their dataset (unlike the dataset used in this paper). In a simulation duration of 24 hours, we consider a total of 144 iterations separated in intervals of 10 minutes. To implement the optimization model, we use the ILP solver called "Gurobi," executing four concurrent threads in an Intel Xeon E5620, 3.4 GHz machine running GAMS Version 24 and 64-bit Windows 7.

2) Evaluation Metrics: For comparing the performance of our optimization model and heuristic algorithm, the following metrics are considered,

(i) *Total number of active CUs* - In each iteration, we count the total number of CUs required to map all the DUs efficiently.

(ii) *Total fragmentation loss* - We evaluate the packing efficiency of the proposed heuristic algorithm in terms of total fragmentation (space wastage) occurring in each cluster. Let K be the total number of clusters formed, C' be the fixed capacity of each cluster, and U_i be the total capacity used by the mapped DUs in cluster i. We define, fragmentation F as,

$$F = \frac{K * C' - \sum_{i=1}^{K} U_i}{C'} \tag{13}$$

(iii) *Total number of handover* - In each iteration, we calculate the total number of handover events occurring for all the 100 DUs.

3) Performance Analysis : In Fig. 9, we show the total number of CUs used in each iteration by the heuristic algorithm and the optimization model. We observe that the heuristic performs closely with the optimization model. However, due to its greedy nature, the heuristic slightly overestimates the total number of CUs by 2.5% over all the 144 iterations.

In Fig. 10, we compare the performance based on the packing efficiency of each cluster. Compared to the optimization model, the heuristic algorithm has slightly higher fragmentation. More specifically, the proposed heuristic has 28% high fragmentation than the optimization over all the 144 iterations. Due to the more number of active CUs in the heuristic algorithm, some of the clusters are not tightly packed and hence we observe a slightly high fragmentation loss.

In Fig. 11, we show the total number of handovers occurring in each iteration for all the 100 DUs. We observe that the heuristic algorithm has a 13% higher number of handovers than the optimization model.

VI. Reactive vs Proactive clustering Techniques

In the previous section, we showed how a deep learning-based model is used to get an accurate traffic forecast of 100 $cells$. We also proposed a DU-CU mapping algorithm that can cluster the DU's inside a CU pool, such that the number of active CUs is reduced without much fragmentation loss. In this section, we present how the traffic prediction can be further used to reduce the number of active servers, by comparing the proposed proactive learning-based mapping technique with the standard reactive learning-based approach.

A. Reactive clustering

Since internet traffic is dynamic and bursty, the VMs hosting peak DU loads can overburden the CU resources, which could easily lead to user throughput reduction. Such issues could easily be avoided with a reactive approach. In Reactive clustering, a new server (or a VM) could be instantiated based on certain policies. One such example of a policy is CPU load, where a new VM is initiated as soon as the CPU load reaches a certain threshold. Thus, the servers will never get overloaded, preventing the loss of QoS for the UEs. However, the reactive approach could lead to inefficient usage of resources because of resource reservations. The threshold value of each server determines how many resources are to be reserved.

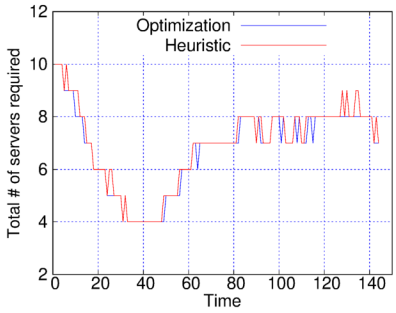

Fig. 9. Total number of active CUs vs Time

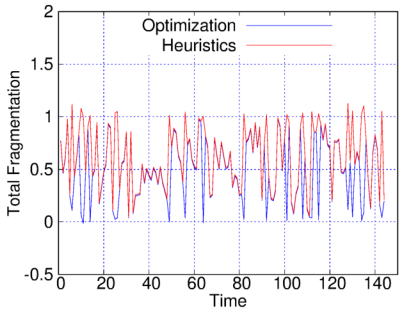

Fig. 10. Total Fragmentation vs Time

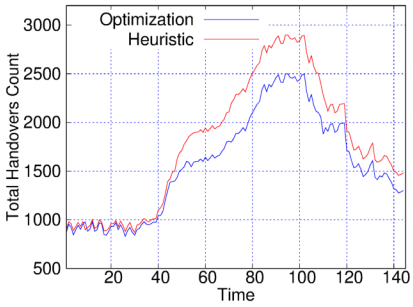

Fig. 11. Handovers Count vs Time for 100 DUs

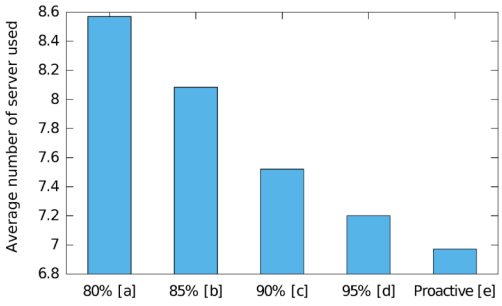

Fig. 12. Average number of servers used every 10 minutes for reactive clustering with different threshold [a-d] and Proactive clustering [e]

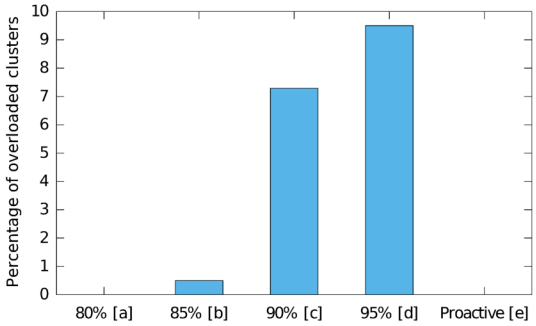

Fig. 13. QoS loss for reactive clustering with different threshold [a-d] and Proactive clustering [e]

B. Proactive clustering

In proactive clustering, a traffic prediction model is used to forecast the expected future traffic. The predicted load on each server can be used to proactively assign the DUs to the CU server without reserving the compute resources. If the expected load on each server is known beforehand, the resources can be allocated by considering both current and upcoming traffic.

C. Performance comparison between Proactive and Reactive clustering techniques

In this section, we compare proactive clustering with reactive clustering at different threshold levels for the number of active servers required and the number of overloaded servers.

1) Average number of servers used every 15 minutes: This measure the average number of active server while doing the clustering every 10 minutes for both reactive and proactive clustering. For reactive clustering new VMs are created when threshold CPU usage is reached. For proactive clustering traffic prediction results are used. Maximum current traffic and predicted traffic are used to cluster the DUs. This allows the CPU to reach 100% utilization. Fig. 12 compares the number of servers required for proactive and reactive clustering for the different threshold values. For reactive clustering, the average number of servers used increases with a reduction in the threshold. We can notice that 80% threshold has a higher average server requirement compared with using a higher threshold.

2) Percentage of overloaded clusters: This result gives the percentage of clusters where the CPU utilization exceeded the total available compute capacity. Fig. 13 shows the percentage of clusters that exceeded the total compute capacity. For the reactive clustering with 80% threshold, total capacity is never exceeded. This is because in reactive clustering with 80% threshold more resources are reserved and hence the servers never get overloaded. But, for the 85% threshold, some clusters are getting overloaded. For 90% and 95% threshold values, around 7% and 9% of clusters exceeded their capacity, respectively. In proactive clustering, as it considers future traffic while creating DU clusters, the total capacity is never exceeded.

VII. Related Works

A. DU-CU Mapping in C-RAN

In C-RAN architecture, to reduce the capital expenditures (CAPEX) and the operational expenditures (OPEX), multiple DUs can be mapped to the same CU to minimize the number of active CUs [8]. Therefore, designing an optimal DU-CU clustering scheme becomes a key problem in the C-RAN architectures. Such a problem has been identified as *bin-packing problem* [5] and is already proven to be NP-Hard. Therefore, to reduce the complexity, several heuristic approaches are proposed in the literature that map DUs to CUs in a time-efficient manner. To this end, authors in [9], propose a particle swarm-based optimization to find an optimal mapping, but the model has non-linear time complexity. In [10], the authors proposed semi-static and adaptive DU-CU switching schemes for C-RAN. Authors in [11] introduce the QoS-aware joint DU-CU scheme to minimize the power consumption in C-RAN. There are numerous such schemes present in the

literature, but most of them require prior knowledge of DU traffic profiles.

B. Mobile Traffic Prediction

1) Time Series Based Prediction: To understand and predict network traffic, several time series models have been proposed in the literature. Prediction techniques such as Auto-Regressive Integrated Moving Average (ARIMA) [12] and Exponential Smoothing [13] use linear regression techniques to predict the time series data. Fengli *et al.* used the ARIMA model to predict the data traffic for 9000 base stations in the state of Shanghai [14]. Similarly, authors in [15] use an exponential smoothing scheme for traffic prediction in GSM/GPRS networks. Authors in [16] improve the time series prediction accuracy by identifying strongly correlated regions and feeding them together to ARIMA models. Such regression and moving average techniques have been widely used for decades. However, they can only be used to model a single DU and fail to capture the spatial correlations [17]. When dealing with C-RAN architecture, the model should predict the traffic for a whole city-wide network simultaneously. Further, with the emergence of 5G technology, cellular networks are expected to become more complex with highly dynamic mobility patterns. Therefore, such linear statistical models are not suitable for accurate traffic prediction in mobile networks.

2) Deep Learning-Based Prediction: With recent advances in Artificial Intelligence (AI), deep learning-based models have been widely adopted for cellular traffic prediction, capturing the complex and non-linear spatio-temporal correlations in the traffic data [18]. Authors of [19] use Recurrent Neural Networks (RNNs), such as LSTM, to predict the cellular traffic data. They show that deep learning-based models outperform the classical statistical models in traffic prediction. RNNs have demonstrated a remarkable ability to predict time series data, but they fail to capture the spatial dependencies in the data accurately.

VIII. Conclusions and Future Work

By mapping multiple DUs to the same CU, the C-RAN can optimize coverage with the minimal number of CUs while gaining high multiplexing gains. However, dynamically mapping DUs to CUs isn't trivial due to the unpredictable nature of network traffic and mobility patterns. Our paper presented a methodology for an optimal city-wide C-RAN network that utilizes a two-phase approach. In the first phase, we propose to use the *ConvLSTM* model, which combines the existing Convolutional Neural Networks (CNN) and Long Short Term Memory Networks (LSTMs). Based on the predicted traffic from the first phase, we develop an optimal DU-CU mapping scheme that accounts for cost and quality objectives. Simulated results show that the *ConvLSTM* model outperforms existing state-of-the-art models with a 26% lower RMSE and a 36% lower MAPE. Compared to reactive threshold-based clustering, proactive clustering can reduce the number of active CU servers by up to 18% every 10 minutes without overloading.

In the future, we would like to extend the proposed framework to include the concept of dynamic adaptation of the RAN functional split. We will also study and quantify the impact of frequent re-assignments of DUs to CUs on ongoing connections.

Acknowledgment

This work has been supported by the project "CCRAN: Energy Efficiency in Converged Cloud Radio Next Generation Access Network" funded by Intel India.

References

[1] Cisco, "White Paper on Visual Networking Index, Global Mobile Data Traffic Forecast Update." https://tinyurl.com/y8kuucvk, 2019.

[2] H. Gupta, M. Sharma, A. Franklin A., and B. R. Tamma, "Apt-ran: A flexible split-based 5g ran to minimize energy consumption and handovers," *IEEE Transactions on Network and Service Management*, vol. 17, no. 1, pp. 473–487, 2020.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 1097–1105, 2012.

[4] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–80, 12 1997.

[5] B. Korte and J. Vygen, *Combinatorial Optimization: Theory and Algorithms*. Springer Publishing Company, Incorporated, 5th ed., 2012.

[6] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences)*. W. H. Freeman, 1979.

[7] G. Barlacchi *et al.*, "A multi-source dataset of urban life in the city of milan and the province of trentino," *Scientific Data*, vol. 2, 04 2015.

[8] A. Checko, H. L. Christiansen, Y. Yan, L. Scolari, G. Kardaras, M. S. Berger, and L. Dittmann, "Cloud ran for mobile networks—a technology overview," *IEEE Communications Surveys Tutorials*, vol. 17, no. 1, pp. 405–426, 2015.

[9] K. Thaalbi, M. T. Missaoui, and N. Tabbane, "Performance analysis of clustering algorithm in a c-ran architecture," in *International Wireless Communications and Mobile Computing Conference (IWCMC)*, pp. 1717–1722, 2017.

[10] S. Namba, T. Warabino, and S. Kaneko, "Bbu-rrh switching schemes for centralized ran," in *7th International Conference on Communications and Networking in China*, pp. 762–766, 2012.

[11] J. Yao and N. Ansari, "Qos-aware joint bbu-rrh mapping and user association in cloud-rans," *IEEE Transactions on Green Communications and Networking*, vol. 2, no. 4, pp. 881–889, 2018.

[12] G. E. P. Box and G. Jenkins, *Time Series Analysis, Forecasting and Control*. USA: Holden-Day, Inc., 1990.

[13] D. Tikunov and T. Nishimura, "Traffic prediction for mobile network using holt-winter's exponential smoothing," pp. 1 – 5, 10 2007.

[14] F. Xu, Y. Lin, J. Huang, D. Wu, H. Shi, J. Song, and Y. Li, "Big data driven mobile traffic understanding and forecasting: A time series approach," *IEEE Transactions on Services Computing*, vol. 9, no. 5, pp. 796–805, 2016.

[15] P. R. Winters, "Forecasting sales by exponentially weighted moving averages," *Manage. Sci.*, vol. 6, p. 324–342, Apr. 1960.

[16] S. Ntalampiras and M. Fiore, "Forecasting mobile service demands for anticipatory mec," in *2018 IEEE 19th International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)*, pp. 14–19, 2018.

[17] R. Li, Z. Zhao, X. Zhou, J. Palicot, and H. Zhang, "The prediction analysis of cellular radio access network traffic: From entropy theory to networking practice," *IEEE Communications Magazine*, vol. 52, no. 6, pp. 234–240, 2014.

[18] J. Wang, J. Tang, Z. Xu, Y. Wang, G. Xue, X. Zhang, and D. Yang, "Spatiotemporal modeling and prediction in cellular networks: A big data enabled deep learning approach," in *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, pp. 1–9, 2017.

[19] H. D. Trinh, L. Giupponi, and P. Dini, "Mobile traffic prediction from raw data using lstm networks," in *2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, pp. 1827–1832, 2018.