Pathfinder: End-to-End Automation of Coverage Mapping of 4G/5G Networks at Street Level

A. Almazi Bipon, Ahmed Osman, Md Shantanu Islam, A. Taufiq Asyhari and Raouf Abozariba
College of Computing, Birmingham City University, United Kingdom
{md.almazibipon, ahmed.osman, mdshantanu.islam, taufiq.asyhari, raouf.abozariba}@bcu.ac.uk

Abstract—**Despite 5 revolutionary generations and 18 releases, automated measurement tools for cellular networks offer limited programmability and their integrated APIs remain difficult to reproduce. We demonstrate the challenges and solutions related to building web and mobile-based applications for continuous cellular networks data collection and analysis. This is shown in the form of an integrated suite of reliable, low-cost cloud-based data processing, querying and analysis software functions that domain experts and laypeople users can utilize to assess cellular networks' quality of service at street level.**

Index Terms—**Drive Test, Network Analysis, 4G/5G**

I. Introduction

As mobile network operators (MNO) roll out 5G networks, there is an increasing need for tools and techniques to enable effective coverage mapping and data analysis. While cellular networks have tremendously evolved at all network layers, the same has not happened from the monitoring and evaluation perspective. Industry standardised techniques such as the immediate and logged Minimization of Drive Tests (MDT), which were introduced in 3GPP Release 8 and enhanced in Release 11, suffer from positioning/quantisation errors and scarcity of user equipment reports, where only less than 10% of users enable MDT through subscription, limiting its data quality [1], [2]. The growing demand for reliable tools and techniques for the use of public and private companies has generated interest from the industry. There are now several available solutions in the market offering different measurements and analysis but with limited customisation options and do not enable access to APIs [3], [4], making them difficult to reproduce. On the other hand, tools developed by the research community such as [5]–[8], are tailored to small-scale studies to assess the performance of specific technologies such as MIMO, beamforming, and mmWave. We demonstrate Pathfinder: an autonomous, end-to-end coverage mapping tool enhanced with state-of-the-art data and statistic visualisation capabilities and based on control plane (C-Plane) information. Pathfinder provides on-the-fly network statistics such as the number of vertical and horizontal handover events, bandwidth, and signal strength in an area, allowing users to precisely locate coverage holes at street level, among other vital metrics. By incorporating spatial area selection for 4G/5G coverage mapping, our solution offers insights into network coverage and performance per selected region. In this short demo paper, we specifically discuss the key underlying technologies we leveraged to build a successful mapping tool using only open-source software and off-the-shelf mobile devices.

II. Passive network data acquisition

We designed an Android application to collect C-Plane data, including signal strength and ASU level, SNR, handover events, Physical Cell ID (PCI) etc. To monitor four networks (Vodefone, Three, O2 and EE) simultaneously, we 3D printed a unit capable of holding four Android-based smartphones (we used Samsung Galaxy S21/5G). The application then utilizes local storage and Firebase (cloud) to record timestamped data. This application's main technical developments are discussed next.

LTE/NR-NSA discrimination: The Android application employs the *TelephonyManager* class, a built-in feature in the Android system, to retrieve network information from the mobile device [9]. The application's primary objective is to gather network data on both the NSA and SA connections, as well as LTE since these RATs will coexist in the foreseeable future. However, the LTE subclass in the Android system represents both 5G NR-NSA and 4G RAT and does not distinguish between the two. To address this issue, we leverage the *dataNetworkRegistrationInfo* flag, which provides information on the NR-NSA connectivity, to discern 5G NSA and LTE. It is also worth noting that it is considered within Android App-based studies that Reference Signal Received Power (RSRP) measurements of commercial smartphones are accurate and correct without confirmations from author-based spectrum analyzer measurements and power level calibration.

Synchronizing GPS with network measurement: The application also captures geolocation coordinates, which are crucial to associate network measurements. Geo tags are obtained from the Android system's *locationManager* class, which operates asynchronously with the network data collection process. We use the network update time and the GPS timestamps to sync the data. The significance of the temporal disparity between the network update time and GPS update time varies with the speed of drive tests. In the case of a stationary device, the temporal disparity bears no significance. As we solicit updates from the network and GPS every 2 milliseconds, we determined that a maximum temporal disparity of 10 milliseconds is the optimal value to facilitate synchronization between the device location and network parameters.

Foreground services: In order to improve the data collection process, our application incorporates an autonomous operating mode, enabling uninterrupted data acquisition for an extended period. This was achieved by binding data acquisition with an Android foreground service that remains active even when the device is in idle mode, minimizing the need for human intervention. We also created two separate foreground services for network data acquisition and GPS data acquisition to enhance the application's automatic functionality further. This approach is critical because relying on manual assistance for a long period can be costly, inefficient, and prone to human error.

Data collection: To collect the data and to test our system, we deployed the data logging devices in refuse vehicles, operated by county councils to generate saturated drive test data. Due to its small size and light weight, the data logging devices can be used for general drive or walk test. The collected data is stored locally on the mobile device's memory and in cloud servers, providing a backup source. The local database also enables the application to transmit all the collected data captured in areas with no network coverage at a later time when a connection becomes available, ensuring that all data is accurately captured and processed, regardless of network coverage/congestion limitations inherent in cellular networks.

System optimization: During one year of field deployments under varied scenarios and monitoring four MNOs, we updated the mobile application several times to enhance the system's reliability. During these updates, we noticed that the Firebase SDK has a maximum allowed JSON file size of 16 MB for a single transmission. This situation emerged when a handset roamed in areas with no network coverage for a long period, resulting in a substantial number of instances of offline data – the data collected while the device had no network coverage was not sent to the cloud database. The significant volume of offline data caused the application to crash, interrupting services. To address this issue, we implemented a maximum limit of 100 instances to be transmitted to Firebase in a single transmission. If there is more offline data, it is segregated into batches of up to 100 instances before transmission to ensure that the transmission size remains within the Firebase SDK allowed limit.

We also encountered a problem with the log file size in the long-running application. We started by monitoring all errors and warnings for future updates and bug fixing. However, within just a few weeks of data collection, the application crashed when attempting to write a new entry due to limited memory caused by the large size of the warnings log file, which had grown to 1-2 Gigabytes. To resolve the problem, we only track errors in the log files ignoring the warning logs.

III. Real-time monitoring

As noted above, the data collection method is a mobile application that runs on Android phones and collects information about carrier signals and coordinates, generating time series datasets with geospatial properties. This data is then transmitted to Firebase through a wireless cellular connection. The carrier information is saved in the root of the database in a node named after the carrier (e.g., Vodafone and O2). Each node has several child nodes named after the time the data was collected, containing values about the carrier data. To enable real-time device health check and survey progress on maps, we use a Zabbix server with several functions, including battery status, device location and coverage map. The design of the server is straightforward, but here we discuss two methods, which enable efficient monitoring, specific to the application.

Back-end fetching: To handle large volumes of data in real-time, we utilised Zabbix agents to read the data, but this method proved to be ineffective as the data volume increased, resulting in long loading times and high cloud processing costs. We developed a Python script to perform data segmentation, reducing processing complexity. The method utilises the *requests* library to execute HTTP GET requests to Firebase and store the data in a local database. *startAt* parameter, a Firebase query, is set to the previously recorded timestamp, to get the next set of data. The fetched data is saved in JSON objects named after the carrier, and relevant information is extracted and saved in the respective files. We then use the *L.glify.lines* [10] a web graphic language renderer plugin for leaflet in typescript to display latitude and longitude keys, size, and colour of the lines.

Front-end fetching: When reading small amounts of data, we deploy front-end fetching. This method is useful specifically when only a few child-nodes need to be accessed, and the data does not need to be saved locally. For instance, we used front-end fetching to monitor the battery status on each phone, which is essential for maintaining operations, where only four nodes of information requires retrieving. We use Firebase JavaScript SDK [11] and leverage *ChildEventListeners* to listen for events of a specified database reference.

IV. Visualized Analyses of Network Parameters

To build the web application for data analysis we used Laravel, an open-source PHP framework that offers a variety of tools to simplify development, such as caching and CRUD. We also utilized several JavaScript libraries, including Open Street Maps and leaflet-area-selection, to add functionality and interactivity to the web application. A Percona Server was deployed to build a MySQL database. In this section, we highlight the underlying technologies we used to integrate polygon-based filtering to simplify coverage mapping.

Heatmap/contour representation: We use geolocation data and signal strength to generate a colour-coded gradient line to produce coverage heatmaps. We define coverage as the 4G/5G signal strength in RSRP on a given road/street. Areas without 4G or 5G connectivity are not colour coded, indicating the presence of coverage holes. Gradient lines span a range from Green (representing strong signal strength) to Yellow (moderate signal strength) to Red (indicating weak or absent signal strength), corresponding to RSRP values ranging from -65dBm to -120dBm, a widely recognised range for evaluating network signal strength. We used 'Open Street Map' to visualise coverage on a map [12]. To enhance the usability of the heatmap, we made the gradient-coloured line transparent, enabling street names to remain visible. Considering vehicular speed, if the distance between two subsequent points exceeds 500 meters on the map, we break the line sequence. This value can be adjusted according to the data collection system. In addition, when analysing repeated drive test routes, we superimposed the gradient-coloured lines to produce a composite representation.

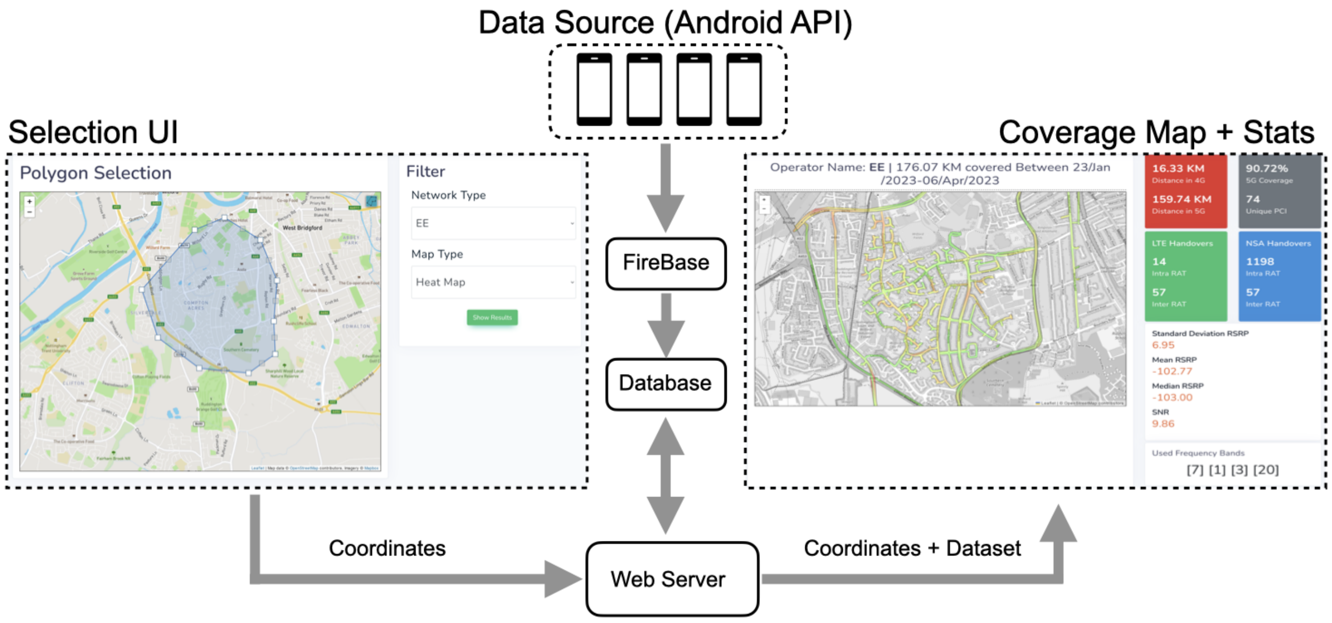

Fig. 1. The architecture of Pathfinder with illustration of Selection UI and Coverage Map.

It is also worth noting that the dataset contained a moderate number of stale data points, which can add unnecessary delay in presenting data. To address this problem we created a script to compare the distance between any given two subsequent data points, and if the distance is less than 3 meters, we discard it.

Polygon-based filtering: The execution time of data visualisation grows fast as the size of the data points increase. In addition, processing of entire data sets generate out-of-context statistics, which can not be used to form specific region-based analysis. We leverage a polygon-based method for the first time in this domain, enabling a user to draw a custom outline on a map, specifying only the area where users intend to construct the coverage map and generate a set of key stats. We used a Javascript plugin 'leaflet-area-selection', to capture the coordinates of the selected polygons to achieve the 'Selection UI' functionality [13] which is then used to querying the database. We use the same methodology used for generating global heatmaps but with significantly smaller datasets to gain a faster and more interactive response.

V. Conclusion

We demonstrate the enabling technologies for developing tools for drive tests intending to generate street-level coverage heatmap, providing bench-marking between mobile network operators and vital statistics, such as signal strength and handover events. We also discussed the inherent challenges in modern cloud platforms and mobile/web applications to channel data to remote servers. In addition, the technologies discussed are essential for building live monitoring graphical user interfaces to control real-time data collection and optimizing system operations to enable autonomous drive tests, which meet vehicle's routines. For the first time, we presented how a polygon-based approach is leveraged to focus on areas of interest instead of generic global coverage maps and non-localized statistics. Figure 1, captures the end-to-end workflow of the coverage map representation along with data capture methodology. There are various directions to extend this work. An exciting approach is to utilize the power of machine learning models to predict coverage in areas where mapping using drive tests is not feasible, such as in no-entry and hard-to-reach regions. This work was partially sponsored by the Local Government Association (LGA), UK, through Nottinghamshire County Council (NCC).

References

[1] W. A. Hapsari *et al.*, "Minimization of drive tests solution in 3GPP," *IEEE Communications Magazine*, vol. 50, no. 6, pp. 28–36, 2012.
[2] C. K. Anjinappa *et al.*, "Coverage hole detection for mmWave networks: An unsupervised learning approach," *IEEE Comms. Letters*, vol. 25, no. 11, pp. 3580–3584, 2021.
[3] rantcell, "rantcell," https://www.rantcell.com/index.html, accessed: June 15, 2023.
[4] rootmetrics, "rootmetrics," https://www.rootmetrics.com/en-US/home, accessed: June 15, 2023.
[5] C. Hausl, *et al.*, "Mobile network testing of 5G NR FR1 and FR2 networks: Challenges and solutions," in *Proceedings of EuCAP*. IEEE, 2022, pp. 1–5.
[6] K. Kousias *et al.*, "Implications of handover events in commercial 5G non-standalone deployments in Rome," in *Proceedings of SIGCOMM*, 2022, pp. 22–27.
[7] A. Narayanan *et al.*, "A variegated look at 5G in the wild: performance, power, and QoE implications," in *Proceedings of SIGCOMM*, 2021, pp. 610–625.
[8] Rohde and Schwarz®, "TSME6 Ultracompact Drive test Scanner, Product Brochure," Version 11.00.
[9] Android, "TelephonyManager," https://developer.android.com/reference/android/telephony/TelephonyManager, accessed: March 01, 2023.
[10] Robert Lee Plummer Jr., "Leaflet.glify," https://github.com/robertleeplummerjr/Leaflet.glify, accessed: February 24, 2023.
[11] Google, "Firebase JavaScript SDK," https://firebase.google.com/docs/reference/js/v8/firebase.database, accessed: February 24, 2023.
[12] Iosphere GmbH, "leaflet hotline," https://github.com/iosphere/Leaflet.hotline, accessed: March 25, 2023.
[13] B-Open, "leaflet area selection," https://github.com/bopen/leaflet-area-selection, accessed: March 25, 2023.