Investigating Measurement Scheduling Strategies in Low Resource Networks (Poster)

Taveesh Sharma
University of Cape Town
Cape Town, South Africa
shrtav001@myuct.ac.za

Josiah Chavula
University of Cape Town
Cape Town, South Africa
jchavula@cs.uct.ac.za

ABSTRACT

Community networks have been proposed by many networking experts and researchers as a way to bridge connectivity gaps in rural and remote areas of the world. Many community networks are built with low-capacity computing devices and low-capacity links. Such community networks are examples of low resource networks. The design and implementation of computer networks using limited hardware and software resources has been studied extensively in the past, but scheduling strategies for conducting measurements on these networks remains an important area to be explored. In this study, the design of a Quality of Service monitoring system is proposed, focusing on performance of scheduling of network measurement jobs in a low-resource network. In this paper, we present a testbed for conducting performance evaluation of two measurement scheduling algorithms and present an analysis of trends in their performance with varying experiment profiles.

CCS CONCEPTS

• **Networks** → **Network measurement**.

KEYWORDS

Internet Measurements, QoS, Scheduling Algorithms, Websockets

ACM Reference Format:
Taveesh Sharma and Josiah Chavula. 2021. Investigating Measurement Scheduling Strategies in Low Resource Networks (Poster). In *ACM SIGCAS Conference on Computing and Sustainable Societies (COMPASS) (COMPASS '21), June 28-July 2, 2021, Virtual Event, Australia.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3460112.3472310

1 INTRODUCTION

Community networks are open, free and neutral network infrastructure built and maintained by citizens and organisations who pool their resources and coordinate their efforts [5]. These networks are often run by non-profit organizations. Further, community services like local networking, voice connections and internet access can be developed in cooperation with local stakeholders [6]. ITU statistics [2] for South Africa report that there were around 165.6 mobile-cellular subscriptions per 100 inhabitants as of 2019. Of these 165.6 subscriptions, around 102.2 correspond to mobile broadband subscriptions. Also, about 92.6% of total active internet users worldwide, and 94.7% of internet users in South Africa accessed internet through their mobile phones as of January 2021 [3]. The usage of smartphones as a computing and networking device thus presents an opportunity for constructing mobile crowdsourcing applications [9]. Such applications can be used as a platform to conduct research to improve community networks.

System architectures based on crowdsourcing are generally more complex as compared to systems where design, implementation and execution are centralized [9]. Other challenges arise due to adopting smartphones as an execution unit. It is difficult to precisely capture internet performance data through smartphones because the conditions, like location and bandwidth, are always changing over time, and as the subject moves with the device.

Although the load of executing measurements is shifted to smartphones in a crowdsourcing-based architecture, a centralized server should be able to allocate the measurements intelligently, given the availability of resources like network bandwidth and execution capacity of smartphones. If a large number of measurements are carried out using a limited number of vantage points, the obtained results could suffer from the observer effect [9], i.e a bias in the measurements due to the measurement infrastructure itself. Measurement processes that are executed in common points and links could contend for shared network resources. This contention for resources is also called measurement conflict problem [11]. Thus, scheduling and synchronization of measurements among the smartphones is important to ensure proper resource utilization and accuracy of results.

The costs associated with building large-scale active internet measurement platforms that provide open access to anonymized data to researchers are generally very high [4]. Networks may become overly congested and additional data costs on users' side may be incurred as a result of the injection of probing packets. A major reason behind this phenomena lies in the skewed distribution of measurement jobs towards a few vantage points. We posit that these costs can be reduced by ensuring proper scheduling and distribution of these measurements. Our study thus introduces mobile crowdsourcing in the context of a low cost monitoring system for low-resource networks with a focus on measurement scheduling strategies. In pursuit of finding the best possible design, we present an empirical analysis of alternative techniques for measurement scheduling and synchronization. We study the trends in performance of these strategies with varying parameters and internet measurements. Our study aims to answer the research question, *what design considerations are necessary for a low cost internet measurement system for under-resourced networks?*

COMPASS '21, June 28-July 2, 2021, Virtual Event, Australia

ACM ISBN 978-1-4503-8453-7/21/06.
https://doi.org/10.1145/3460112.3472310

2 RELATED WORK

Scheduling of active internet measurements in classical networks has been studied by few researchers. Some of these works [8, 10] have not given much attention to conflicts between individual measurements. Calyam *et. al* [7] propose a novel enhanced-EDF heuristic that allows concurrent execution of measurements to address the measurement conflict problem. Scheduling algorithms for both periodic and on-demand jobs have been proposed in this work, and have been shown to perform better than Round Robin and EDF through computer simulations as well as in an internet testbed. Qin *et. al* [11] compare the EDF scheme with their novel graph-coloring based approach, and through computer simulations their approach proves to achieve effective contention resolution and low execution delays.

Most of the QoS monitoring research on community networks has been done on the basis of Guifi.net (the largest community network) or other interconnected European community networks. Limited research has been performed on monitoring network characteristics in the context of community networks for developing regions, and measurement scheduling aspect of frameworks that characterize these networks has mostly been overlooked. In classical networks, the design and evaluation of measurement scheduling algorithms is based on computer simulations or synthetic testbeds. Also, these algorithms rely on the assumption that the execution time of measurements is a constant, contrary to a real-world scenario where it could vary because of several factors like signal strength, geolocation and time of the day. In our work, we make an attempt to apply the principles used in literature to develop a QoS measurement platform that can be used and tested in any real-world low resource network.

3 METHODOLOGY

We consider Round Robin (RR) and Earliest Deadline First - Concurrent Execution (EDF-CE) algorithms for evaluation as each of these algorithms can be used in a concurrent context. Also, each of these algorithms accepts the actual conflict relationship between the jobs as input and generates a schedule that tries to avoid these conflicts. For implementation and evaluation of scheduling algorithms, we conduct a preliminary set of tests to obtain an estimate of maximum execution time for each job type. As part of these experiments, we conduct on-demand tests on mobile phones and Raspberry Pis at different times of the day for each job type. We record execution times of these tests in our database and use the maximum value of execution time observed across 15 days in our experiments. The maximum value of execution time is chosen so as to have a safe time window for the next job to execute when the previous job hasn't finished it's execution.

In our experiments, we evaluate implementations of scheduling algorithms (Round Robin and EDF) through the following metrics:

(1) *Platform delay* ($d_{ij}^{platform}$) : Refers to the delay between the actual and expected result display on the platform. This metric captures the effect of multiple factors like poor signal strength at data collection points, scheduling algorithm overhead and network latency. If the job is of type t, then the platform delay is given by:

$$d_{ij}^{platform} = t_p(J_{ij}) - [t_s(J_i) + jp_i] \quad (1)$$

where $t_p(J_{ij})$ is the time at which the result is received at the platform from data collection points for the jth instance of ith job, and $t_s(J_i)$ denotes the start time of the job. Average platform delay for a job can be obtained by averaging the platform delay across all it's instances.

(2) *Average waiting time (AWT)* : Refers to the average duration for which the job stays in the waiting queue before being dispatched to one of the data collection points.

(3) *Node Busy Time Ratio (NBTR)*: Refers to the percentage of time for which a measurement node executes jobs during an experiment. This metric is also critical in determining overall load distribution.

$$NBTR = \frac{Total\ time\ spent\ by\ the\ node\ in\ job\ execution}{Total\ time\ of\ the\ experiment} \quad (2)$$

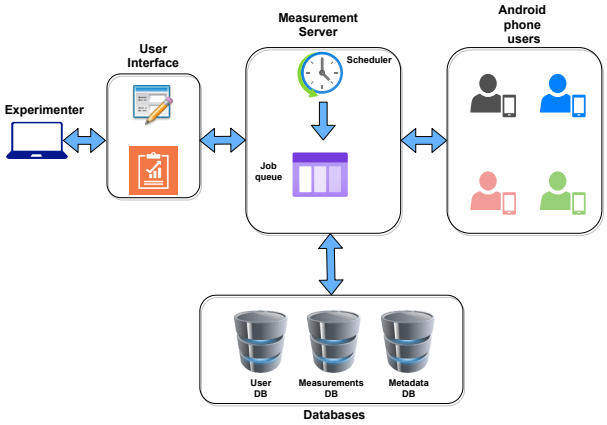

Figure 1: A diagram showing the architecture of QoSMon, the proposed measurement system. A key component in our architecture is a centralized measurement server, which is responsible for orchestrating measurements. Measurements are conducted in data collection points distributed within the community network. Scheduling of measurements is not decentralized to end-user devices because a network-aware central server can be useful in determination of conflicts prior to jobs getting scheduled. This will further help in reducing the congestion in the network and thus lead to results with higher accuracy. Our implementation supports Android phones running a custom version of MobiPerf [1]. Communication between the phones and the measurement server is facilitated through Websockets, whereby the server sends measurement jobs to the phones as and when they are scheduled. In addition to the server, a user interface is developed for community network administrators and researchers to schedule experiments, visualize the collected data and perform QoS analysis. Community network members will be able to visualize application usage through their smartphones.

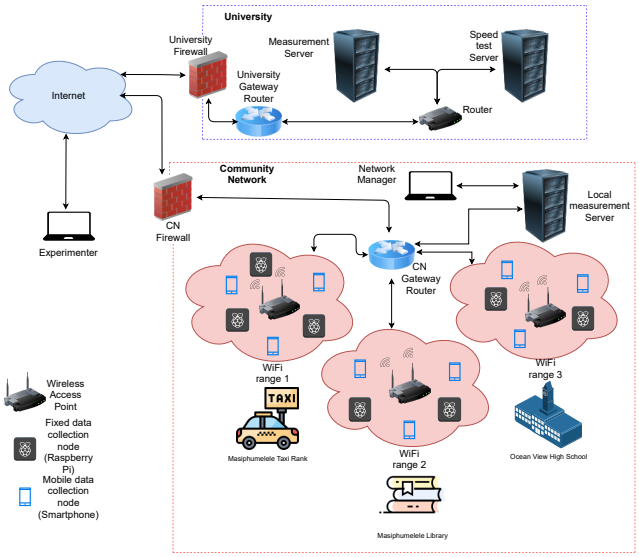

Figure 2: A number of Android phones will be used to run each type of MobiPerf-supported experiment. The phones should have the requisite hardware for running at least Android Marshmallow (API level 23). Multiple Raspberry Pis (4B, 4 GB RAM) will also be used to run the same set of experiments that MobiPerf supports. The Android phones and Raspberry Pis will be connected to the community network through multiple wireless access points. Two measurement servers will be used to schedule the experiments and store the collected data. One of these servers will be used by researchers in the university and the other will be used by community network administrators. The server located in the university will be responsible for running scheduling algorithms while the server located in community network will only be responsible for accepting jobs from network administrators and sending them to the university server. Speed tests will be run against a separate server located in our university's campus in Cape Town.

4 PRELIMINARY RESULTS

Our current implementation of the measurement server supports the concurrent versions of Round Robin and EDF algorithms. In our preliminary experiments on a test network, job types were selected uniformly from the set {ping, dns_lookup, traceroute, http, tcp_speed_test }. The target servers for ping, dns_lookup, traceroute and http were chosen uniformly from 8 popular websites. TCP speed tests were run against a local speed test server running on the same test network. The periods of the jobs were chosen uniformly from the range [5, 10] minutes and the experiments were allowed to run for 2 hours for each scheduling algorithm on 4 Android phones (API level 27). Our measurement server was running on 64-bit Ubuntu 20.04 on Intel Core i5-10210U CPU @ 1.60GHz × 8 processor with 16 GB RAM.

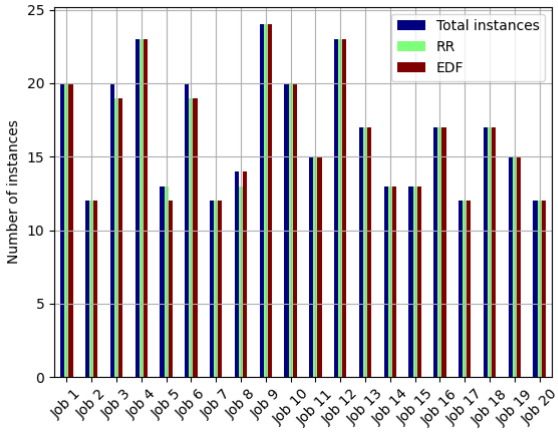

Figure 3: Number of job instances that ran for RR and EDF along with total number of expected instances for RR and EDF algorithms. It can be inferred from this figure that the missed job ratios were an absolute zero for most of the jobs. The reason for jobs being missed in RR and EDF in some cases can be due to the parallel execution of high-duration jobs like traceroutes and TCP throughput tests, causing the replacement of currently pending jobs with their new instances.

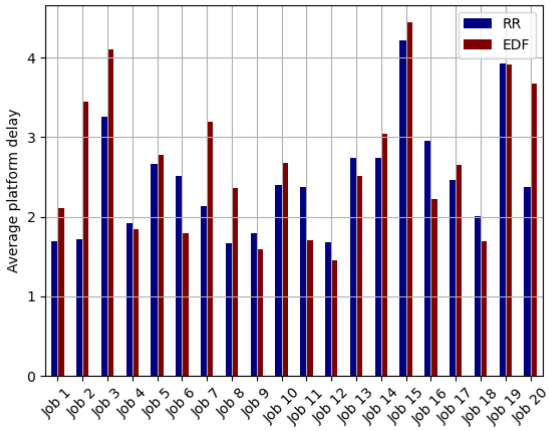

Figure 4: We achieved low average platform delays in minutes for almost all of the jobs. In a future work, these delays can be further reduced by calibrating the system with more accurate estimates of expected values of execution times of different job types. In our current implementation, the expected values for job execution times were chosen as the maximum value observed across our preliminary tests. We used the maximum value so as to have a safe time window for the next job to execute when the previous job has not finished it's execution.

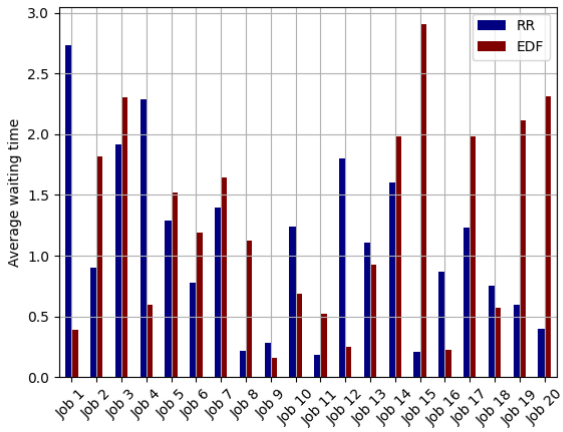

Figure 5: To investigate the delays further, we checked the average waiting time (in minutes) for each of the jobs. It can be inferred that in some cases (Job 15, for example), jobs being stuck in the waiting queue for long was one of the reasons of higher platform delays. Also, most of the scheduled jobs had more waiting time in EDF as compared to RR. This can be attributed to the fact that these jobs had higher periods and an almost similar inter-arrival time compared to other jobs.

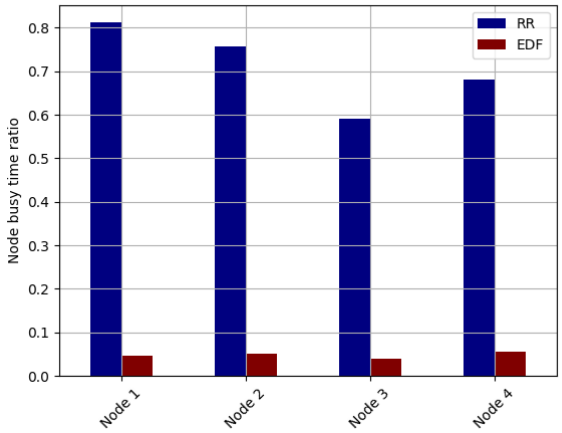

Figure 6: To check the distribution of jobs among the four nodes, we calculated the node busy time ratio. Figure 6 shows that EDF scheme distributes jobs more evenly as compared to Round Robin. A more even distribution of jobs in case of EDF could have led to lesser number of jobs per phone during the 2 hour window, resulting in a lower busy time ratio. This also means that EDF algorithm is more energy efficient than Round Robin from an end user perspective as lower values of node busy time ratios directly translate into less drainage of smartphone batteries.

5 FUTURE WORK

Our current work is the first step towards an optimal low resource network monitoring solution. In future work, more advanced scheduling algorithms like the ones proposed in the work by Qin *et al.* [11] will be implemented and considered for evaluation. A future implementation can also be made to support fixed nodes, like Raspberry Pis. For performance evaluation, dependant variables like number of jobs, number of measurement nodes, time of the day and mobility of the devices etc. will be considered. Another interesting addition to this paper would be the support for on-demand measurements and the adjustment of job schedules in accordance with a custom priority order.

REFERENCES

[1] 2019. *MobiPerf.* Retrieved June 10, 2021 from http://mobilab.eecs.umich.edu/mobiperf.html
[2] 2020. *Statistics.* Retrieved May 27, 2021 from https://www.itu.int/en/ITU-D/Statistics/Documents/statistics/2020/MobileCellularSubscriptions_2000-2019.xlsx
[3] 2021. *Internet users in the world 2021 | Statista.* Retrieved May 27, 2021 from https://www.statista.com/statistics/617136/digital-population-worldwide/
[4] Giuseppe Aceto, Alessio Botta, Walter De Donato, Pietro Marchetta, Antonio Pescapé, and Giorgio Ventre. 2012. Open source platforms for Internet Monitoring and Measurement. In *2012 Eighth International Conference on Signal Image Technology and Internet Based Systems.* IEEE, 563–570.
[5] Roger Baig, Ramon Roca, Leandro Navarro, and Felix Freitag. 2015. guifi. net: A network infrastructure commons. In *Proceedings of the Seventh International Conference on Information and Communication Technologies and Development.* 1–4.
[6] Bart Braem, Chris Blondia, Christoph Barz, Henning Rogge, Felix Freitag, Leandro Navarro, Joseph Bonicioli, Stavros Papathanasiou, Pau Escrich, Roger Baig Viñas, et al. 2013. A case for research with and on community networks.
[7] Prasad Calyam, Chang-Gun Lee, Phani Kumar Arava, and Dima Krymskiy. 2005. Enhanced EDF scheduling algorithms for orchestrating network-wide active measurements. In *26th IEEE International Real-Time Systems Symposium (RTSS'05).* IEEE, 10–pp.
[8] R Les Cottrell, Connie Logg, and I-Heng Mei. 2003. Experiences and results from a new high performance network and application monitoring toolkit. In *Passive and Active Measurement Workshop.*
[9] Adriano Faggiani, Enrico Gregori, Luciano Lenzini, Valerio Luconi, and Alessio Vecchio. 2014. Smartphone-based crowdsourcing for network monitoring: opportunities, challenges, and a case study. *IEEE Communications Magazine* 52, 1 (2014), 106–113.
[10] Matthew Luckie and A McGregor. 2002. IPMP: IP measurement protocol. In *Passive and Active Measurement Workshop.*
[11] Zhen Qin, Roberto Rojas-Cessa, and Nirwan Ansari. 2010. Task-execution scheduling schemes for network measurement and monitoring. *Computer communications* 33, 2 (2010), 124–135.