UNIVERSITY OF CAPE TOWN

DEPARTMENT OF COMPUTER SCIENCE

CS/IT Honours Final Paper 2019

Title: Quality of Service Monitoring Tool

Author: Meluleki Dube

Project Abbreviation: QOSMON

Supervisor(s): Josiah Chavula

Category	*Min*	*Max*	Chosen
Requirement Analysis and Design	*0*	*20*	20
Theoretical Analysis	*0*	*25*	0
Experiment Design and Execution	*0*	*20*	0
System Development and Implementation	*0*	*20*	20
Results, Findings and Conclusion	*10*	*20*	10
Aim Formulation and Background Work	*10*	*15*	10
Quality of Paper Writing and Presentation	*10*		10
Quality of Deliverables	*10*		10
Overall General Project Evaluation (*this section allowed only with motivation letter from supervisor*)	*0*	*10*	
Total marks		**80**	**80**

Quality of Service Monitoring System for Community Networks

Meluleki Dube

Department of Computer Science
University of Cape Town
South Africa

June 2018

Abstract

This paper presents the development of a Quality of Service Monitoring System for Community networks. This works focuses mostly on the development of the application, Inethi Perf that will collect the different measurement results as well as a place for storing these measurements, so they can be easily accessible. The application will collect different network metrics and send the to the server for storage. Additionally, it provides a way for users to track their data usage within the community network. The works in this paper lay down groundwork for building an actual Quality of Service Monitoring tool as well as a testbed for researchers and system administrators to run measurements on.

Keywords Community Networks, Quality of Service

1. Introduction

Community networks are large-scale decentralized networks that are built and operated by the citizens[20]. The networks are at times run by non-profit organizations who can cooperate with stakeholders to develop community services, which include the internet and local networking[14]. These networks are becoming more and more popular for providing Internet access to remote areas. As these networks are mostly managed by the users in the network it is good to have a monitoring system that users, and the administrators of the network could have used to look at how the network is performing. The performance of the network will be based on network metrics. The networks metrics to be used are time for HTTP response to come back after sending an HTTP request, a DNS lookup speed test for finding our how long it took for a DNS lookup to finish, a TCP throughput speed test, which measures the Download speed and Upload Speed, ping test, which measures the time taken for pinging a chosen target and lastly trace-route values give the different servers that the system hits when sending packets and time taken to reach any of them.

This paper then focuses on building a Monitoring tool for the community networks with Inethi as a case study. There is more emphasis on creating the application for actually collecting the data from the network, and the storage of the data that is collected. The paper also describes an agile process of building a system and different techniques for testing an application. The rest of the paper describes the requirements from the stake holders' perspective, then describes the design and implementation of the system, lastly we describe the testing and results of the application performance.

2. Background

Quality of Service(QoS) is defined to be the ability of the network to provide a service at an assured service level[22]. QoS has relations to the term Quality of Experience(QoE), which explains how users experience a particular service[8]. For example if we have the network throughput to be low, and we were watching a YouTube video, we will not have a good experience of this service. In this case the raw throughput measured will be the Quality of Service measurement while the experience that the users have with streaming the YouTube videos will be Quality of Experience. The above example is illustrates that QoS affects the users QoE of services. Quality of Service includes metrics like throughput and delays being experienced in the network. For QoS metrics to be meaningful and effective each data point needs to be be tied to a particular location, device that recorded the data, and possibly the time the data was collected.

The aim of this work is to build a software tool that would allow different stakeholders in the network to monitor these metrics in order to estimate the Quality of Service that can be expected from the network.

2.1 Community Networks

Community networks are defined to be large-scale self-organized distributed and decentralized networks which are built and managed by the citizens for citizens [14]. The architecture of community networks is usually built with low cost hardware with the nodes often running an open source software[14]. Nodes in a community network are devices that relay data within the network or devices that provide other nodes network access[9]. The links between the nodes can be wireless or wired[9]. Community Networks are part of the subset of networks characterized as: (i)open because everyone has a right to know how they are built, (ii)free because networks access is not based on discriminatory principles (iii)neutral as the network can be used to transmit data of any kind and by any participant[13].

The setup of community networks make them more appealing for providing internet access to unrepresented areas. Some characteristics that allow for this are, decentralization of the network[20] which removes the chances of having a single point of failure within the network. Community networks are also self-managing[14], which means that they will not be so costly as they will not require a body to manage the network since the networks are managed by the users of the network. Examples of community networks include, **Inethi Network** whose goal is to extend community network models by enabling sharing and authoring of local digital resources and services[1].**Zenzele.net** which is a community network that provides cheap telephone and internet access to the community members of the[3]. **guifi.net** that operates in Spain[17], which by 2015 was the world's largest community networks in terms of number of nodes and coverage with over 27000 operational nodes[4].

2.2 Related Work

2.3 Monitoring systems

Monitoring systems are essential to network systems as they give an overall view and many metrics for a network. Monitoring of networks is crucial not only for the end users of the network who need to know the performance metrics of the network, but also for the policy-makers and the network regulators[5]. Three stakeholders in network performance measurements are the end-users, the ISPs and the regulators[6]. All these groups of people all have different use cases for the measurement data. For instance, ISPs can use the data to identify, and isolate network faults and to have an understanding of user's end to end service experience[16]. Data collected from monitoring networks by running certain measurements is not only important to improve user experience on the network, but also important for providing feed back on the design of other upcoming technologies[5].

Monitoring of networks consists of being able run passive and active measurements on the networks. Different measurement types include ping measurement where we measure the time taken to ping a target and if there is any packet loss during the ping process, TCP throughput, this is in two forms, the download speed and the upload speed of the network, DNS lookup, which tells the average time it takes to be able to resolve a name of a host to its ip address, lastly we have http measurements telling us the time it takes to send and get a http response to a known http server. Different systems have been built thus far to take networks measurements. Some of these are accessible to end-users like by going to their website, like OOKLA Speedtest[5]. Tools like OOKLA Speedtest have been criticized for not being scalable and not repeatable[5]. Repeatability has to do with the fact that we cannot continuously take measurements on the network. However, there have been more internet measurement platforms that have been introduced through research. Internet measurements platforms refers to infrastructure that has been dedicated to periodically run network measurement test on the internet[6]. These platforms are built on wifi-enabled micro controllers, which are always actively taking measurements.

2.3.1 Measurement Tools

There have been different network measurement tools designed for collecting network measurements for community networks in particular. Some of these studies where mentioned and discussed by Braem and team[13] and include:

- RIPE ATLAS, which is a measurement infrastructure consisting of thousands of hardware probes and servers all around the globe.[6, 12]. While hardware probes obtain active measurements to determine network connectivity and global reachability, anchors serve as dedicated servers that can act as sources and sinks for the network measurement traffic[12]
- Another tool is the BISMark tool that seeks to measure home broadband performance by means of custom gateway firmware[13]
- Perfsonar-ps, a suit of measurement tools, which can be deployed freely[13].

2.3.2 Network Measurement Platforms

2.3.3 Using Smart phones for network data collections

One approach to taking network measurements is to make use of the users mobile phones constantly take network measurements and then sending these to a central database server for analysis. This can be via an application installed on mobile phones that will collect

measurements on fixed time intervals. This approach has problems, which include privacy issues, power demands of measurements that will affect the mobile phone's battery life to mention but a few. LiveLab, is a solution to address some challenges that arise from using mobile phones in taking network measurements[21]. Livelab is method for measuring real-world smart phone usage and wireless networks with a re-programmable in device logger, which is built to overcome privacy and power challenges[23]. Livelab leverages hashing of the data recorded from the phone to ensure overcome the privacy issues[21]. For power saving, Livelab uses 4 different logging methods, which are[21], Interrupt depended on logging as opposed to polling method. Another method is, Piggy-backing, which is to save all the data on the phone and then collecting them when the phone is connected to power and idle Optimizing logging intervals for periodically logged items is also a method used. Lastly having the logger hitchhike on other system application and services waking up.

2.3.4 MONROE PLATFORM

MONROE platform is a multi-home open access hardware-based platform for running measurements and experiments in operational mobile broadband networks[5]. As of date MONROE platform is said to consist of 250 nodes, both mobile and stationary[10]. Each node runs 3 software, the management software to ensure that the node always remains operational and to enable remote updates to all the other software components. Another component is the maintenance software that monitors the operational status, and the last software is an experiment enabler[10].

Monroe can handle experiments that range from continuous latency measurements to real-time flows together with meta-data like location of data type of device for measuring data and so on[5]. The platform comes with a website for viewing the measurements being collected in the network, Nodes and software for orchestrating the collection of the measurements[5].

2.3.5 Measurement-Lab and Community Lab

Measurement lab(M-Lab) is "an open, distributed server platform for researchers to deploy active internet measurements tools" [15, 13] with the data collected released to the public domain[15, 13]. M-Lab only runs active measurement tests and, in addition conducts measurements between the user/client, and the M-Lab servers to examine the end-end performance along the path[15]. Some tools provided by M-Labs currently include:

- **Network Diagnostic Tool (NDT)** which is a tool that measures the through put between the client and host in terms of download and upload speeds[15, 13]. The tools also tries to determine the causes of slow speeds as well as checks for proxies, NAT devices or middle boxes between the machine running the tests and the M-Lab server collecting the tests thereby providing several objective indicator of user experience of an internet connection[13].
- **Network Path and Application Diagnosis (NPAD)** which uses TCP to measure end-to-end throughput and information about the switch/route queues along the path[15].

Community-lab is said to be an open infrastructure that provides to researchers and experimenters a testbed to carry out experiments within wireless community networks[19]. It comes with a testbed that support experimentally-driven research on community based networks[14]. The testbed provided by Community-Lab was inspired by Planet lab[6, 13]. Each node in Community-Lab consists of two to three devices, and these include the community device, the research device and an optional recovery device connected together via a local network[19]. Community devices are wireless routers, research devices are low powered systems running OpenWRT distribution that makes it possible to allow simultaneous virtual containers[19].

3. Requirement Analysis and Design

This paper focuses mostly on designing the tool for collecting measurements from the phones and storage from these measurements, there will be limited talk about users hence the model diagrams are more concerned with the system and not user interaction with the system as there is limited interaction with the users from collection of data and storage of data point of view.

3.1 Requirements Gathering

To ensure that a software system is useful to all those involved there is need to gather requirements from the different types of users of the system. In our study we gathered these requirements through few techniques.

3.1.1 Communication with Project Supervisor

The initial requirements were given by the project supervisor. Throughout the project we consistently checked in with the supervisor via emails and meetings to ensure that the system being built was per requirements and to gather any ad hoc requirements that might have come up.

3.1.2 Communication with the Inethi Network Stakeholders

The main stakeholders for the software system are, Network administrators of Inethi network, potential researchers in the community and the users of the Inethi network. The stakeholders we interacted with included the directors of the community network, the

administrators of the network and the normal users of the network. For directors of inethi we had face to face encounter with them were we briefed them about the project, and we asked their opinions on it. This meeting enabled us to establish the requirements from the point of view of the directors of the network. Their requirements where adding functionality to be able to see the mostly requested pages. Another request was being able to see the top ip addresses in terms of the ip address downloading and uploading the most.

We also met with the potential users of the network and administrators of the network. This was done through a workshop organized for users to interact with some mock-ups we had designed. For the workshop, we interacted with 3 users of the Inethi network and 1 person having an admin role within the network. We also interviewed the participants about the idea of the project and asked for their thoughts and if there were any features that they needed from the application. The requirements that came from the potential users and network administrators included adding functionality for users to be able to monitor how their apps use the network, allowing parents to monitor the website visited by their children also allowing the administrators to filter the results shown in the graph based on different parameters.

3.1.3 Surveys

We also asked users of the network to answer some survey question at the end of the workshop. This also gave us more feedback that the users might have not mentioned during the sessions. The answers to these surveys were anonymous.

3.2 Requirements Analysis

Using the above techniques we managed to gather a list of requirements and have split these into two sets, the one set is a list of functional requirements, and the other set of is that of non-functional requirements. The list of functional requirements includes:

- Be able to analyze IP addresses sending the most request, and most visited ip addresses
- Users to be able to track app data usage from their phones.
- Users to be able to track data usage from their children's accounts
- A way of showing users how much data is used on inethi

The list of non-functional requirements includes:

- Lightweight(in terms of used resources) application to run on users phones to run data collection.
- Secure ways of storing user data to avoid leaks of collected data. Secure speak to two things, data should not be accessible from outside the system and if it happens, the data should be stored such that the identity of users is not compromised.

One of the resources was left out after discussing with the users that it was a potential security risk. This resource is that of parents being able to monitor their children's usage of the community network. The security issue at risk is that at the end it would be hard to maintain a list of who is a guardian for who and who has permission to look over whose internet usage. This would have ended up leading to anyone in the network being able to see what anyone in the network is using the network for which goes against the privacy requirements. For these reasons, this requirement was left out in the design of the application.

3.3 Design

3.3.1 Package Diagrams

From the high level system design and the requirements we got form the users of the system we were able come up with models of the system, which later on where used to drive development of the system. One is a package diagram and shows the arrangement and organizations of model elements. Figure 1 shows the package diagram of our system. The server in our system contains 5 packages that all represent separate services. The server package can be hosted within the Inethi network server and will then be accessible by the other packages like the web interface and the mobile devices. All the other packages use one, or many of the services offered by the server. An example is the mobile phones, which use the Database Request Handler service to save the recorded data to the database. The reasons for prohibiting direct access to databases from mobile phones is so that we can analyze the data being written to the database to ensure that all data is safe. Also, this allows for us to change the underlying databases without needing to change or update any functionality on the mobile phones.

3.3.2 State and State Transitioning of the System

This section details the different states that the mobile application can be in and how the application will alternate between these statuses. Figure 2 shows the state chart of the mobile application that is to be built. The phone has 3 main states shown in the Live container, and these are ídle, ćhecking_inánd ŕunning measurements. The ídleśtate is the first state the phone enters upon launching the application. In this state the phone keeps a timer, which triggers the checking in, and if necessary running of measurements. During checking in, the mobile application contacts the server to find out

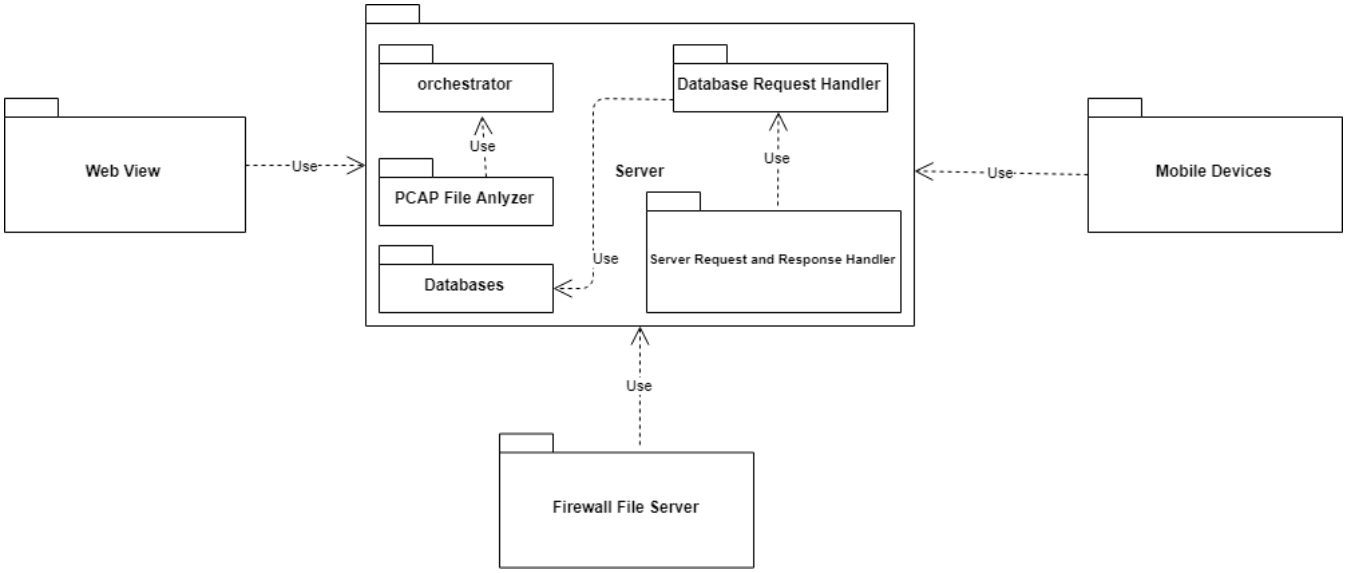

Figure 1. Showing UML package diagrams of the system.

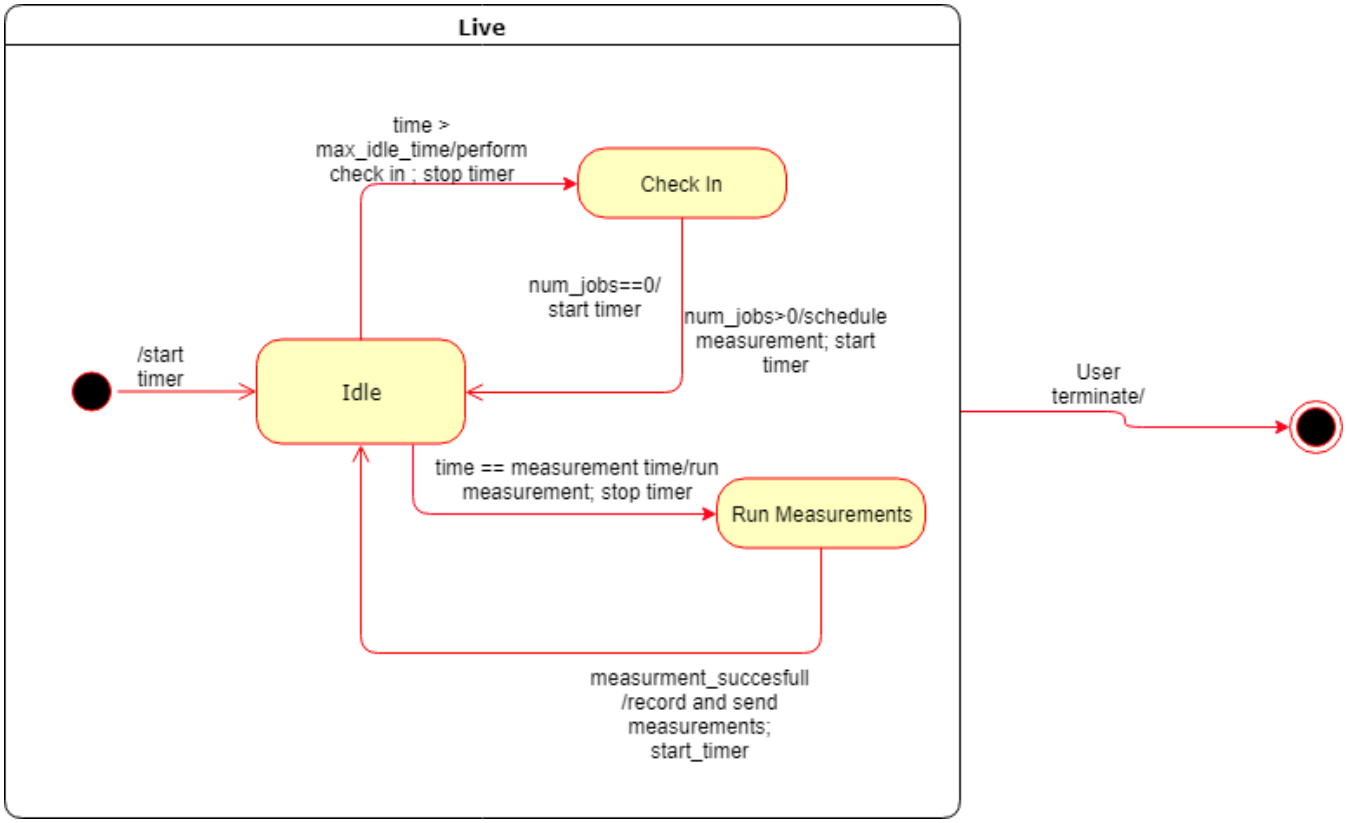

Figure 2. Showing State Machine Chart for the mobile application that we will be building

if there are any experiment jobs that the phone can run. If the server responds with a job, the phone will then schedule these measurements and return to idle state while starting the timer.

If time comes for a particular measurement to run the phone will leave the idle state and move to 'run measurement' state. During the 'run measurement' the scheduled measurements are run, and the results are built and sent immediately to the server as required. The phone then starts the timer and goes back to being idle. Every time the phone leaves the idle state the timer will be stopped and resumed when it comes back. The phone will forever be in this loop unless the user terminates the application directly by which the application will then stop.

3.4 System Architecture

Figure3 shows the architecture of the system after analysing the given requirements with the different components that make up the system. The main components of the system as from the design are:

- **Orchestrator:** used by researchers, system administrator and the system itself to schedule measurements to be run on the data collection points. Admins and researchers will access the collector from a web interface.
- **Collector:** aggregates measurements collected from different collection points and sends these to the server for some analysis and storage.
- **Web Interface:** tool used stakeholders to interact with the system as a whole by doing tasks as, scheduling measurements viewing specific metrics in the network. The scheduled measurement are submitted to the orchestrator.
- **Data storage:** which consists of 3 databases that are going to be used with the system. The measurements database and the metadata databases are adopted from the architecture of the with MONROE platform[5]. We added the user database to be able to store user accounts so that users can be able to have their network usage data stored for later viewing.

3.4.1 Storing of Network Measurement Data

Storage is one important issue that comes up when researchers collect data and try to analyze the data. The storage of data affects how easy it is going to be to extract the data for analysis and to what extent one can be able to run machine learning algorithms on the data. In this project, data will be collected from different devices and different applications(phone application and computer application). As such, there is need to have a good design for the data storage.

One idea to consider is taken from MONROE[5], where the system consisted of a Remote repository and Data Importer, which will remotely collect experimental data from each node after every experiment. It also consists of Database System, which consists of two tables one for experiment measurements, and the other for meta-data like location device type. The database that is used in MONROE is the Cassandra database. Casas and team also used the Cassandra database in their paper for storage of data for the reasons that Cassandra is a fully distributed database with no single point of failure[7].

3.4.2 Databases

Coming up with the system required number of decision to be made. The three databases we have in the system have different use cases and this allowed us to have different databases for each of those.

- The user databases needed to hold user Profile information in a way that it was going to be easy to query user details based on a unique user id or username. We needed a database type that is simple to design while allowing fast queries of a specific user profile. We therefore, decided that the database type to use for this database is going to be a document store database. A document store is a database type where all the entries are stored in documents with the documents addressed using unique keys[18]. These databases offer great performance and horizontally scalability options[18]. In this case they are attractive for their great performance as we needed to be doing fast queries. We settled for Mongo Database as it has a lot of support and a very well documented Java library.
- The Metadata Database is a database that is used to store experiment requests by researchers on the network. When a researcher requests particular experiments to be done on the system, we needed to be able to store this request. The experiment request stored will rarely be queried but when done the queries need to be as fast as possible. Experiment results are given a unique key, which we will use to run queries. For the above reasons we decided to use a document store database as it fits well with the requirements. For this, we also chose the Mongo Database for consistency and for the reasons already mentioned.
- Lastly we have the results' database, which is a database that we will be storing all recorded measurements results. This database stores data indexed by timestamps as we do not need any primary keys to reference measurement. Therefore, to satisfy this requirement a time series database is going t be

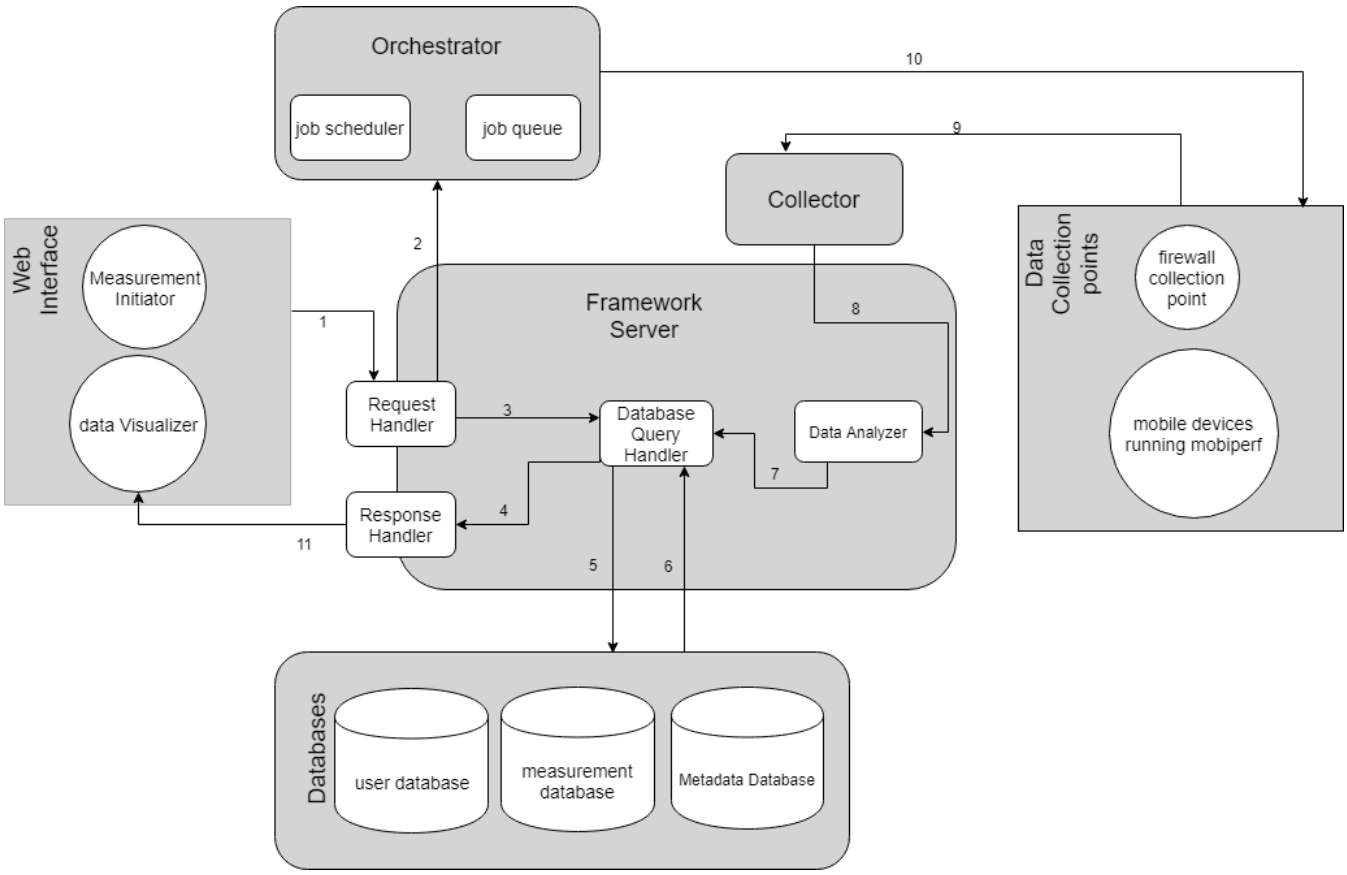

Figure 3. Showing the system overview, different components making the system and the communication directions between these components in the systems.

more suitable. We then decided to go with Influx database as its one of the most well-supported time series database with a good documentation. Influx also allows us to tag some other parameters in every measurement result that we stored so that when queried there can be fast queries for these parameters. An example is that, sometimes, users will want to see measurement results from a particular location. To ensure that this operation is going to be fast and done in Constant Time Big O notation, that is number of comparisons done is not depended on the number of items in the database, we made location parameter a tag. This alone will change the performance of searching for measurements in particular locations. Another tag is the job ID tags for results that corresponded to measurements jobs submitted by researchers

Since all the databases used in this case are noSQL databases we did not have to design the schemas of the databases. Table1 table view of the users' database. It highlights the properties from users that are going to be stored, which include their email address, the

Table 1. Table showing a table view of the Users database, highlighting the properties stored in the database.

Email Address	Hash of Email Address	User Type
String	String	String

MD5 hash of the email address. The rationale for this design is because we do not need to be authenticating the users ourselves. Since users are going to use their email addresses, we can use the authentication services by google and others to do the authentication for us. This will provide more security for users as we don't get to store their passwords, and as seen above, simplifies the design of the system. As mentioned the system will be recording usage data, and to maintain anonymity, we will be storing user data with the hash value of the email address and not their email address to ensure that the data is anonymous and cannot be traced back to the users. For retrieving user data, the users will provide their email address, and in turn the system will check within the database to get their hash value and then

use this hash value to search for any data that belongs to them.

Listing 1. JSON structure showing how the experiment metada documemts stored in the database will look like

```
{
"request_type":string
"job_description":{
"measurement_description":{
"type":string
"key":string
"start_time":string
"end_time":string
"interval_sec":int
"count":long
"priority":long
"parameters":{
//depends on the measurement
}
}
"node_count":int
"job_interval":int
}
"user_id":string
}
```

Listing 1 shows the structure of an individual document that will be stored in the experiment metadata database. The parameter fields will have different structure for different measurement types that are being scheduled. The different possible fields are shown for each measurement type in the appendix. Each job will have an ID field, which is randomly generated and unique for all the jobs scheduled. The user id filed is needed to know, which jobs belong to which users. Here, we will also hash this value with the email address of the researcher who scheduled the job.

Figure 4 shows us the UML Diagrams for the different measurements that can be stored and how they are associated to each other. As can be seen all measurements types inherit from the Measurement class, which has the properties task_key, time, username, and whether a measurement is an experiment measurement result or not. Since we are using Influx Database for storing our data, all measurement data stored needs to be referenced by the time as Influx is a time series database. We are storing the username so that we can know from the database how the number of different users/nodes we are getting the results from. The username is hashed at all times to also maintain anonymity. The property for whether a measurement is related to an experiment stems form the fact that we also record some manual measurements that mobile phone users are going to trigger from the phones. These can also be the realtime measurements that we are constantly collecting to keep track of the quality of the networks. The experiment measurements are those that belong to a particular researcher and thus are as a result of a researcher scheduling measurements on the system. The task_key of the measurements is because we want to be able to monitor all jobs that are run and be able to tie them back to the people who scheduled the measurements so when they query them we can give them the measurements back. The task key is a unique for all the measurements.

4. System Development

4.1 Implementation and UI Design

The project required two separate User Interfaces. The first interface was for the Android application that would allow users to measure differed metrics on the network. The second interface is the front-end web application that would allow users to view network performance. The design of the front-end web interface was done with users in codesign workshop.

4.2 Development Framework and Methodology

4.2.1 Extending Mobiperf

As we required to create an application to perform measurements in the network, we tried not to reinvent the wheel and looked at what has been done already. One tool that closely implemented most required features is Mobiperf[2]. This is an application developed by MLab group, which is an open source android application that can be used for measuring network performance on mobile platforms[2]. The tool besides being outdated as it used many deprecated classes and methods also missed some of the features we needed. One of these is the ability to collect personal data usage from users. We had to add functional and nonfunctional features to Mobiperf for it to work as we wanted. The functional features added included:

- Keep track of daily data usage from users, which is one of the requirements that the users asked for during the decision workshop. This will allow users to budget the amount of data they need more accurately.
- Included username tag to measurement, allowing users to also record measurement anonymously. The username is hashed just before the data is sent to the server for storage, so we cannot link data back to the users.
- Data recorded is sent into a server of our choosing for storage.

Nonfunctional features added to mobiperf include:

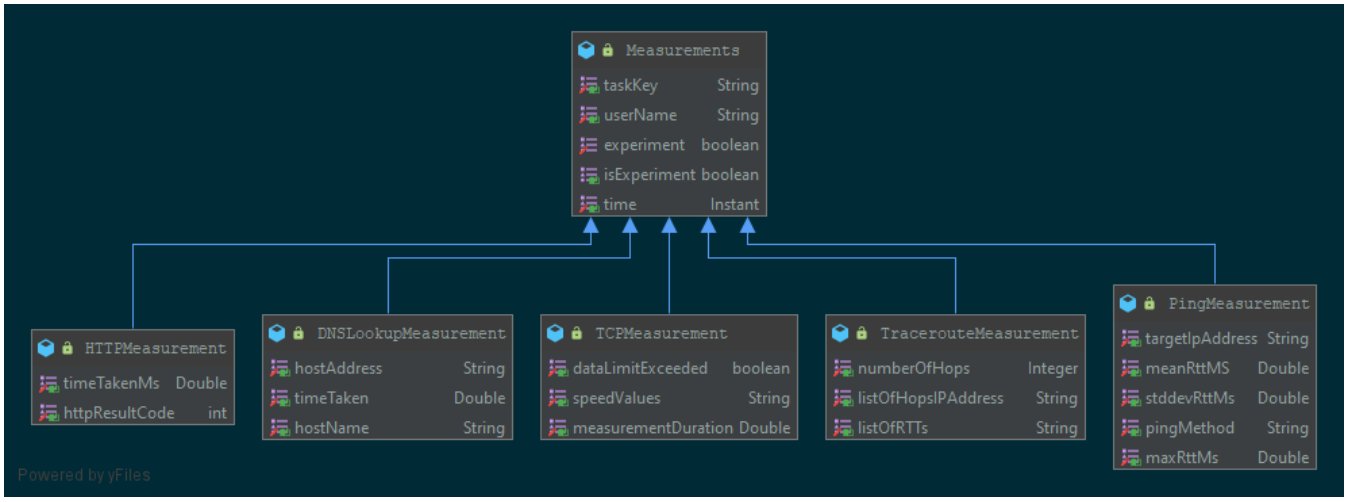

Figure 4. Showing UML class diagrams of the different types of measurements and their associations.

- Most of the implementation was based on the Android API 14 implementation. Many of methods and classes have been deprecated and were not going to work well with the new versions of Android. We had to redesign and implement of most user interface and features to ensure that they were using latest currently supported classes. An example of these are permission asking where old android application required the user to put the list of permissions required in the manifest file and on installation the application would request these permissions, and these would be kept for the duration of the application lifetime on the phone. This needed changing so that an application asks for permissions at run time as required. Before using any feature that requires those permissions you first need to check if you still have permissions and proceed appropriately. Other changes in this regard consist of replacing deprecated classes like TabActivity PreferenceActivity into using the new Android Fragments and FragmentActivity.
- Changed the build system from Ant to the widely used Gradle system.

4.2.2 Programming Language and Framework

The main language used for building the backend of the system is Java with Maven. This allowed us to concentrate on writing the application and not worry much about dependencies needed, and the build cycle for the application. For the android application, the Android Software Development Kit was used to enhance and build the application. For the frontend website, we used React a Javascript framework for building responsive interfaces. We used git and github for version control and collaborations needed. The result of the build of the application is a jar file, which can then be copied to any machine running with Java.

4.2.3 Java Libraries

Where possible, we looked to reuse open source java libraries to avoid reinventing the wheel. The notable open source libraries used are:

- Influx and Mongo Db drivers for java. These where used for performing the queries to the databases.
- Google's gson and the Json library for parsing Json data and converting Plain Old Java Object1 to Json.
- For parsing the pcap files we used the pkts library, which is a pure java library for reading and parsing pcap files.
- Apache VFS library for virtual file systems. This was to allow the users of the network to serve the pcap files in any file system they wish to and the library will able to still access these in a seamless manner.

4.3 Software Development Methodology

The system was built in iterative and Agile software development approach. The process consisted of 3 iterations in total.

4.3.1 Agile Development Methodology

Agile methods are Software Development Methodologies which[11] claims that are about feedback and change. This means that these methodologies are about building software with constant feedback from the target users and building the software in a way that its easier for it to change with change of user requirements. The feedback we got was from the supervisor, members of the Ocean's view community and the masters stu-

dents. All the above represent the stakeholders in the system.

4.3.2 Iterative Development and User Centred Design

As mentioned we had 3 iterations with the different stakeholders involved. The first iteration involved was feasibility study, which we ran with the members of Oceans's view. During this session we presented interactive mockups to the users. During this we had users tell us what they were doing with the mockup to get feedback on the affordance of the features we had built. The results of this iteration were used to drive the developers process.

4.3.3 User Evaluation

We then had sessions with post graduate students who are potential users of the experiment scheduling feature of the system. During this we presented the version of the application we had and asked for their feedback on the work done till this point. This evaluation was done as a group session and not one-to-one. The feedback from this session where used to go through another iteration of design and implementation. We repeated user evaluation process one more time after the initial feedback.

4.3.4 Testing, Documentation and Maintainability

To ensure the credibility of the system, we required testing of the system. Testing was done as a combination of unit testing and manual testing. By manual testing we refer to the fact that the system was left in environment that simulated the production environment. We scheduled a number of different measurements types, this included reoccurring measurements and one time schedules. We then had phones running system left for few days. The system was then used as if it were in the production process and errors and bugs caught were During this one main bug was found. This bug happened to come from the influx java driver, which missed functionality to add column fields from super classes for inherited measurements. As can be seen from Figure 4 this feature was needed to ensure that all measurement types have the fields from the super class. The Influxdb Java driver is open source, so the code was modified to add this feature. The modified code was also submitted to the Influxdb Java driver repository to ensure that future people who need this feature can have it come with Influx driver.

Most functionality is documented to allow for code to be easily maintained by future developers. The system was also built to allow for easy reconfigurations at all fronts. The user applications consists of preferences page to allow users to change work flows. The backend system has a JSON config files that the administrators of the system can use to make the system behave as they want.

5. Results and Findings

This sections seeks to speak into the evaluation the application. One of the non-functional requirements was to design a lightweight application in terms of number of resources needed by the application. The resources that were focused on include storage space taken on the phone, battery used during operation, and the data used in terms of internet. To figure these out the application was left running o the phones over three days. After the three days we then checked to see the data that the application was used, the battery used, and the amount of extra storage that was used.

The application on its own without any extra data takes up 6.7MBs of storage. Initial battery level before running the experiments was 100Initial data used by the application was at 0 mbs.

The settings on the application are as follows:

- Checking in time was 2 minutes. This means that every two minutes the phone will check up with the server to get new measurement jobs to run. The phone will thus have 3 new measurement jobs to run every two minutes.
- The maximum data to be used capacity was set to 50 MBs. That means that the application could use only upto 50MBs after which it will be paused.
- Application sent user personalized data every 2 minutes.

The reasons for the above conditions is because stress testing was what was being aimed for. The server had 6 jobs and every time the phone checked in it got these 6 jobs. The results are presented below.

5.1 Results

Table 2. Data showing how much results were used by the phone.

Resource	Result
Storage Taken	6.8MBs
Data used	6 MBs
Battery Capacity	80%

Table 2 showing the results from running the application for the 3 days with the above mentioned settings. As can be seen, the application is not really resource intensive. We managed to use 6 MBs in data usage. This 6 MBs used includes the data used for running measurement tests, sending personalized user data also checking for extra data. Therefore, we can see from that in terms

of internet usage the application is lightweight as was one of the non-functional requirements. The application is also efficient in terms of storage resource needed. This conclusion is reached as the application required extra 0.1 MBs for storage. The 0.1 MBs used was for probably for storing the user preferences. Aside the preferences the application does not store any more details. Lastly the application also uses the battery efficiently. After being left to run for 3 days as the only main application on the phone, the application used 20% battery. The daily averages for the resource used are estimated to be as follows, for storage usage 0.3MBs, for data Usage 2MBBs and for battery used 6%. Therefore, it is fair to conclude that the non-functional requirement relating to the application being lightweight was met as not much resources are being utilized by the application.

6. ETHICAL, PROFESSIONAL, AND LEGAL ISSUES

This tool involves the collection of user data, which is going to be recorded and stored in a database for viewing by the stakeholders in the application in one way or another. Therefore, we needed to ensure that we obtain consent for recording user data from users before doing so and after getting their consent, we have to ensure that the data is safely stored and cannot be traced back to the users. To ensure privacy of users is guaranteed, two main things where done:

1. For measurement data collection, we hashed the user names associated with the data we were recording. Therefore, in the measurements' database instead of user names we stored the hashed username values.
2. We, also, instead of storing the exact location that data is recorded we stored the coarse location, which is an approximate location. This ensures that the data recorded cannot also be tied to a specific location, which would increase chances of identifying users.

The above measures all where put to reduce the chances of users accessing data that is not meant for them and only allowing owners of the data to be able to view their data.

7. Conclusion

This work has defined a lightweight application, IntethiPerf for collecting network measurements in a community network. Since the application is lightweight in terms of data usage, this means that the network won't be clogged up with a lot of measurements data. We saw how to mitigate the legal issues that come with recording personal user data through hashing and not using exact user location. We also saw in this work how building a system using an agile approach can be helpful in dealing with changes that come during the development process.

8. Future Work

This work just showed the beginning of a network monitoring system, using mobile phones as collectors of network measurements. More work still needs to be put, and the collection of measurement should in the future be done by dedicated Wi-Fi enabled microcontrollers and not be left for the phones. Adding Wi-Fi enabled hardware to run measurements and not rely on phones will make realtime data collection possibly. This will also ensure that we will also always have devices to collect measurements just in case one of the stakeholders want to run or schedule immediate measurements. We can also implement machine learning algorithms to help automatically monitor the network and take precautions as needed. These can be put in the analyzer component of the system and will analyze all recorded data so to detect any anomalies within the network.

References

[1] inethi project.

[2] The m-lab mobiperf data set.

[3] Zenzeleni.net.

[4] A technological overview of the guifi.net community network. *Comput. Netw. 93*, P2 (Dec. 2015), 260–278.

[5] Alay , ., Lutu , A., García , R., Peón-Quirós , M., Mancuso , V., Hirsch , T., Dely , T., Werme , J., Evensen , K., Hansen , A., Alfredsson , S., Karlsson , J., Brunstrom , A., Khatouni , A. S., Mellia , M., Marsan, M. A., Monno , R., and Lonsethagen, H. Measuring and assessing mobile broadband networks with monroe. In *2016 IEEE 17th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)* (June 2016), pp. 1–3.

[6] Bajpai , V., and Schönwälder , J. A survey on internet performance measurement platforms and related standardization efforts. *IEEE Communications Surveys Tutorials 17*, 3 (thirdquarter 2015), 1313–1341.

[7] Casas , P., Vanerio , J., and Fukuda , K. Gml learning, a generic machine learning model for network measurements analysis. In *2017 13th International Conference on Network and Service Management (CNSM)* (Nov 2017), pp. 1–9.

[8] Kist , A. A., and Brodie , L. Quality of service, quality of experience and online learning. In *2012 Frontiers in Education Conference Proceedings* (Oct 2012), pp. 1–6.

[9] Micholia , P., Karaliopoulos , M., Koutsopoulos , I., Navarro , L., Baig Vias , R., Boucas , D., Michalis , M., and Antoniadis , P. Community networks and sustainability: A survey of percep-

tions, practices, and proposed solutions. *IEEE Communications Surveys Tutorials 20*, 4 (Fourthquarter 2018), 3581–3606.

[10] Schwind , A., Seufert , M., Alay , ., Casas , P., Tran-Gia , P., and Wamser , F. Concept and implementation of video qoe measurements in a mobile broadband testbed. In *2017 Network Traffic Measurement and Analysis Conference (TMA)* (June 2017), pp. 1–6.

[11] Williams , L., and Cockburn , A. Agile software development: it's about feedback and change. *Computer 36*, 6 (June 2003), 39–43.

[12] Bajpai, V., Eravuchira, S. J., and Schö nwä lder, J. u. Lessons learned from using the ripe atlas platform for measurement research. *SIGCOMM Comput. Commun. Rev. 45*, 3 (July 2015), 35–42.

[13] Braem, B., Bergs, J., Blondia, C., Navarro, L., and Wittevrongel, S. Analysis of end-user qoe in community networks. In *Proceedings of the 2015 Annual Symposium on Computing for Development* (New York, NY, USA, 2015), DEV '15, ACM, pp. 159–166.

[14] Braem, B., Blondia, C., Barz, C., Rogge, H., Freitag, F., Navarro, L., Bonicioli, J., Papathanasiou, S., Escrich, P., Baig Viñ as, R., Kaplan, A. L., Neumann, A., Vilata i Balaguer, I., Tatum, B., and Matson, M. A case for research with and on community networks. *SIGCOMM Comput. Commun. Rev. 43*, 3 (July 2013), 68–73.

[15] Dovrolis, C., Gummadi, K., Kuzmanovic, A., and Meinrath, S. D. Measurement lab: Overview and an invitation to the research community. *SIGCOMM Comput. Commun. Rev. 40*, 3 (June 2010), 53–56.

[16] Ford, M., and Eggert, L. Report on the workshop on research and applications of internet measurements (raim). *SIGCOMM Comput. Commun. Rev. 46*, 3 (July 2018), 10:1–10:4.

[17] lluis.dalmau. What is guifi.net?

[18] Nayak, A., Poriya, A., and Poojary, D. Article: Type of nosql databases and its comparison with relational databases. *International Journal of Applied Information Systems 5* (01 2013), 16–19.

[19] Rameshan, N., Navarro, L., and Tsalouchidou, I. A monitoring system for community-lab. In *Proceedings of the 11th ACM International Symposium on Mobility Management and Wireless Access* (New York, NY, USA, 2013), MobiWac '13, ACM, pp. 33–36.

[20] Selimi, M., and Freitag, F. Towards application deployment in community network clouds. In *Proceedings of the 14th International Conference on Computational Science and Its Applications — ICCSA 2014 - Volume 8584* (Berlin, Heidelberg, 2014), Springer-Verlag, pp. 614–627.

[21] Shepard, C., Rahmati, A., Tossell, C., Zhong, L., and Kortum, P. Livelab: Measuring wireless networks and smartphone users in the field. *SIGMETRICS Perform. Eval. Rev. 38*, 3 (Jan. 2011), 15–20.

[22] Soldani, D., Li, M., and Cuny, R. *QoS and QoE Management in UMTS Cellular Systems*. 11 2006.

[23] Wang, T.-Y., Jin, H., and Nahrstedt, K. mauditor: Mobile auditing framework for mhealth applications. In *Proceedings of the 2015 Workshop on Pervasive Wireless Healthcare* (New York, NY, USA, 2015), MobileHealth '15, ACM, pp. 7–12.

Appendices

Listing 2. JSON structure showing how the parameters of HTTP will look like

```
{
    "parameters":
    {
        "url":"www.google.com"
    }
}
```

Listing 3. JSON structure showing how the parameters of DNS will look like

```
{
    "parameters":
    {
        "target":"www.google.com"
        "server" : "null" //put a null
            in their for now
    }
}
```

Listing 4. JSON structure showing how the parameters of Ping will look like

```
{
    "parameters":
    {
        "target":"www.vula.uct.ac.za"
    }
}
```