Network Selection Algorithms for Multi-Homed mobile Terminals in a Heterogeneous Network Using Utility-based MADM and Mobile Terminal Movement Prediction

by Jiamo Liu

Prepared for O. Falowo
Department of Electrical Engineering
University of Cape Town

Submitted to the Department of Electrical Engineering at the University of Cape Town in partial fulfilment of the academic requirements for a Bachelor of Science degree in Electrical and Computer Engineering.

November 2016

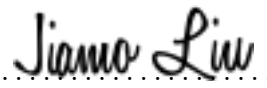
Declaration

1. I know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.

2. I have used the IEEE convention for citation and referencing. Each contribution to, and quotation in, this report from the work(s) of other people has been attributed, and has been cited and referenced.

3. This report is my own work.

4. I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as their own work or part thereof.

Signature: 

Jiamo Liu

Date: 13/11/2016

Abstract

As the number of network subscribers increases, it has become a very challenging task to satisfy the ever-increasing demand of bandwidth by mobile terminals using current infrastructures. Multi-homed terminals are proposed as one of the possible solutions because of its bandwidth-aggregation feature, thus this paper proposes a network selection algorithm for multi-homed mobile terminals using movement prediction and MADM.

In this paper, network and movement models are establish in order to facilitate the simulation. The algorithm predicts the positions of the user with Kalman Filter, and then establish a list of RATs that are qualified as candidates using information of predictions, finally the algorithm iterates through all combinations of the candidates and evaluate the suitability of each combination based on three attributes, namely monetary cost, power dissipation and bandwidth utility. Once the most suitable set of interfaces are found, the user will switch his connections to the most suitable combination of interfaces if the dead time has passed, in order to alleviate the ping-pong effect.

In addition to making logical network interface selection decisions, the simulation of the algorithm has also suggested that the prediction mechanism is able to reduce the number of hand-overs caused by ping-pong effect and ensure the ubiquitous connection within the prediction window period if the predictions are reasonably accurate. Furthermore it is also observed that prediction mechanism enhances the rationality of the decision making process in certain scenarios.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background to the study

Mobile terminals have been one of the focal point of technological development with revolutionary ideas such as smart phones in the past decade. The increase in network subscribers and advancement of mobile terminals led to an increasing demand of bandwidth to satisfy services such as video streaming, therefore it is of great interest to study how to satisfy these demands with the current infrastructure and data traffic. It is shown that most mobile terminals (MT) today has the ability to connect to multiple different radio access technology (RAT) either alternatively or simultaneously [6], i.e. multi-modal or multi-homed MT. A multi-modal MT experiences a very low call blocking probability compared to that of a single modal mobile terminal [7], and the benefits of multi-homing include, but not limited to, permanent and ubiquitous access, reliability, load sharing, bandwidth aggregation [4], and etc. The study focuses on multi-homing since bandwidth aggregation is one of the potential solutions to the ever-increasing demand of bandwidth.

The current RATs in operation are 2G, 3G, 4G, WIFI, and so on. Due to the increase in demand of bandwidth by users, 2G is no longer an effective or efficient option, therefore only 3G,4G,5G and WIFI are considered in the study. It should be noted that each RAT, except for 5G, has an official set of standards with regards to its bandwidth, mobility support, and etc. A big percentage of mobile networks is heterogeneous wireless network (HWN), which is defined by having more than one RAT co-existing in an area, therefore a decision has to be made

about which set of RATs, in a multi-homed situation, the user should connect to and how the workload should be distributed amongst the different networks, in order to maximise the benefits for the user.

1.2 Objectives of this study

1.2.1 Problems to be investigated

The problem under investigation in this study is a RAT selection algorithm that performs the optimal network selection when the current and predicted locations of a multi-homed mobile terminals are given.

1.2.2 Purpose of the study

The study is to establish a reasonably accurate movement model and network model for the purpose of simulation. A location-predicting algorithm is designed to select the optimal set of interfaces based on attributes of the networks. Furthermore the study should address or contain the following aspects:

- Substantiating the logic behind the network selection decision
- Reducing the number of hand-overs when possible
- Alleviating ping-pong effect caused by moving in and out of the network in quick succession

Finally the algorithm is simulated to prove its validity.

1.3 Scope and limitations

The scope and limitations are:

- Lack of official standards of 5G network
- Signal strength model is linear instead of logarithmic
- The speed of movement is not considered when the algorithm forms candidate list

- The states of movement model are assumed to be linear
- Only three attributes are compared during the comparison process
- Bandwidth re-scaling of elastic and adaptive services is not considered

1.4 Plan of development

The plan of development of this report is as follows:

- Chapter 2 - Presents relevant studies that will facilitate the design of the network selection algorithm
- Chapter 3 - Defines and validates all models involved in the simulation
- Chapter 4 - Presents details of the implementation of the network selection algorithm and models, as well as an overview of the interaction of different components of the network selection algorithm
- Chapter 5 - Presents and validates results from the simulation
- Chapter 6 - Draws conclusions based on the results and discussion
- Chapter 7 - Suggests future work to improve on the current models and algorithm

Chapter 2

Literature Review

2.1 Development of wireless network

The development of new generations of mobile wireless networks usually requires to modify the fundamental attributes of the current mobile networks (including new frequency band, new transmission technology, etc.), and such changes are usually non-backward compatible, which means the new generation technology can not be implemented using the existing operating equipments [5]. The evolution of mobile network technology leads to improvements in quality of service (QoS) provided by the network. The minimum QoS a specific type of network must be able to provide is labelled as the QoS standards of that network. The current mobile network technology under research is 5G which aims to bring significant improvement to 4G in terms of bandwidth and mobility.

2.1.1 Zero Generation (0G)

0G is the start of wireless telephone service, and it can be dated back to the end of World War II. Concept of cell had not yet been realised and implemented in 0G, thus the mobile operator had to set up the calls. The capacity of the network was low as only a very limited number of channels were available. The technology usually took the form of radio telephones in car and only basic voice communication service could be realised [8]. As restricted as the technology appears to be now, it still played a vital role in long distance communication back then.

2.1.2 First Generation (1G)

At this point, the cell concept was introduced by having a transceiver station to cover a relatively small area, and all mobile terminals within the area connected through the same transceiver. 1G mobile technology used analogue radio signals to transmit voice messages. The voice call was modulated to about 800MHZ [5] and above, and Frequency-Division Multiple Access (FDMA) was used to allocate radio resources to subscribers within a cell [8].

In comparison to the future mobile network generations , 1G technology ha a lot of shortcomings such as low capacity, unreliable hand-off, poor voice links, and especially lack of privacy. The lack of privacy is due to the fact that voice calls were played back in the radio tower which makes it vulnerable to unwanted eavesdropping. 1G is however better at transmitting from a location that is far away from the radio tower in comparison to 2G. This is attribute to the nature of analogue signals in 1G, the analogue signals suffer a gradual degradation in performance whereas the digital signals would completely fail once the signal is weaker than a certain threshold. [8].

2.1.3 Second Generation (2G - 2.75G)

2G was introduced in the end of 1980s, this marked the beginning of the digital era of wireless communication network. GSM (Global System for Mobile Communication) quickly became the most widely adopted standard (used in 212 countries) after its launch in Finland in 1991, as a result, international roaming was common amongst mobile operators all over the world, thus subscribers could use their devices in many parts of the world. The multiple access technologies employed in 2G were TDMA (time division multiple access), CDMA (code division multiple access),etc. 2G technology also utilised compression and decompression algorithm, so that more calls could be accommodated per bandwidth unit, which led to better capacity [8].

The major advantage of 2G over 1G was that digital signals required less power than that of analogue signals, therefore the device became smaller (transmission power is related to the size of device) and battery also lasted longer [5]. The main drawback of 2G was that it was designed with very limited data transmission capabilities, therefore it was not suitable for web browsing, video streaming, and

etc.

The increasing demand of data services led to the introduction of 2.5G technology which is also known as GPRS (General Packet Radio Service). Besides removing restriction of number of characters allowed in SMS, 2.5G also significantly improved the data rate from 9.6Kbps (GSM) to a maximum of 115Kbps (GPRS) [5], and this improvement led to the introduction of new mobile services such as WAP (Wireless Application Protocol), MMS(Multimedia Messaging Service), mobile games, and etc. [8].

The last modification of 2G before its successor was EDGE (Enhanced Data Rates for GSM Evolution), and this technology was also labelled as the 2.75G [8]. As the name suggests, the data rate of EDGE is up to 384Kbps [5], which is a considerable amount of improvement in comparison with GPRS's 115Kbps. The reason why EDGE truly gained popularity was that no additional hardware or software was required to exploit the service [8]. Even though EDGE fulfils the requirement of 3G, but its technology is different from that of 3G. The packet transfer on the air interface behaves similar to a circuit-switched call, therefore it is less efficient than 3G because some packet connection efficiency is lost during the circuit-switching process [5].

2.1.4 Third Generation (3G -3.75G)

3G was developed to satisfy the requirements of multimedia application, however it did not take long before internet became the most popular application of 3G, subsequently various improvements were introduced to adapt 3G to a more internet-centric user base [8]. Mobility impacts on the performance of the network, 3G should provide at least 144Kbps for users in a car or a train, 384Kbps for pedestrians and 2Mbps for stationary users [5]. Furthermore it also has greater spectral utilisation efficiency, which in turn leads to greater capacity [8].

In order to accommodate the tremendous amount of subscribers worldwide, cells were made smaller to permit frequency reuse, however such approach is limited, and this constraint had led to the development of the newer technologies [5].

The International Telecommunication Union (ITM) published a family of standards of 3G, which was collectively called IMT-2000. The mobile systems that

adhere to the requirements of IMT-2000 are UMTS (Universal Terrestrial Mobile System) and CDMA2000. WCDMA (Wideband Code Division Multiple Access) was used as the air interface of 3G [9]. 3G is a very flexible network standard as it is able to operate under numerous multiple access technologies, such as CDMA, TDMA, etc. HSDPA (High-Speed Downlink Packet Access) and HSUPA (High-Speed Uplink Packet Access) were marketed as an evolution to the 3G network, and they were labelled as 3.5G and 3.75G respectively. The main feature of these technologies were enhanced speed in uploading and downloading [8].

2.1.5 Fourth Generation (4G)

4G is the successor of 3G network with more bandwidth and services offered. The data bandwidth of 4G amounts to an staggering 1Gbps, which is 500 times faster than its predecessor 3G. It should be noted that 4G had completely moved away from the circuit-switching technology, instead all services were operated in conjunction with IP (Internet Protocol). The reason of the migration to packet based network was to provide a platform where all services and technologies could function together, thus achieving network convergence and simplification of network infrastructure. 4G network aims to provide users with abundant choices of services at an affordable price and with reasonable QoS any time, anywhere [9], and this is achieved via terminal mobility. Terminal mobility enables the user to have access to network services based on personal identification in a new geographic area.

There are two major challenges with terminal mobility, namely location management and hand-off management. Location management means that the network should be able to track the terminals and connect roaming terminals to network based on its information, such as authentication, local cells, etc. Hand-off management refers to the technique that keeps the user connected while changing network from one to another. In order to solve these problems, Mobile IPv6 was proposed as the standard mobility protocol, IPv6 protocol assigns each terminal with a home address, and when the terminal is out of the local network, its home address then becomes a invalid, and a new IPv6 address (care-of address) would be issued by the network it switched to [5].

2.1.6 Fifth Generation (5G)

5G is the next major network technologies that succeed 4G, however there are currently no agreed standards published for 5G network, therefore all standards in the section are proposed by different papers.

5G technology should enable user to use their devices with an extremely high bandwidth capacity. Users can expect to get access to complete wireless communication without any limitation, and such concept is named World Wide Wireless Web (WWWW) [5]. The aggregate data rate measured in bits per unit area refers to the total amount of data the network is able to offer. This quantity is expected to be 1000 times of that of 4G network. Edge rate indicates the worst reasonable data rate a user can expect, and it is expected to be 100Mbps which is 100 times of that of 4G. Besides the massive enhancement in terms of data rate, 5G also has a very low latency of 1ms, which is a noticeable improvement from 4G's 15ms, such improvements could possibly allow new exciting services such as virtual or enhanced reality [10].

The energy consumption 5G network would ideally be less than that of its predecessors, however this is actually a quite demanding task. The data rate offered in 5G is envisioned to be 100 times faster, which means that energy per bit would have to fall by 100 times in order to keep the same energy consumption [10]. An overview of different generations of mobile network is shown in Table I

Table I: Specifications of different mobile networks [5]

Technology ⇨ **Feature ⇩**	**1G**	**2G**	**3G**	**4G**	**5G**
Start/ Deployment	1970 – 1980	1990 – 2004	2004-2010	Now	Soon (probably 2020)
Data Bandwidth	2kbps	64kbps	2Mbps	1 Gbps	Higher than 1Gbps
Technology	Analog Cellular Technology	Digital Cellular Technology	CDMA 2000 (1xRTT, EVDO) UMTS, EDGE	Wi-Max LTE Wi-Fi	WWWW(coming soon)
Service	Mobile Telephony (Voice)	Digital voice, SMS, Higher capacity packetized data	Integrated high quality audio, video and data	Dynamic Information access, Wearable devices	Dynamic Information access, Wearable devices with AI Capabilities
Multiplexing	FDMA	TDMA, CDMA	CDMA	CDMA	CDMA
Switching	Circuit	Circuit, Packet	Packet	All Packet	All Packet
Core Network	PSTN	PSTN	Packet N/W	Internet	Internet

2.2 Motion Prediction Algorithms

2.2.1 ARMA

ARMA stands for auto-regressive moving average model. It is a time domain approach to predict the values at a future time instant according to current or past observations of parameters [11]. In the ARMA model, the errors are assumed to be Gaussian White Noise, therefore there is no autocorrelation between the errors [12]. While ARMA model is essentially the combination of auto-regressive and moving average model, the two models can however be used independently, thus the models are discussed separately.

AR Mathematical Model

The auto-regressive model states that the current output, y_t, is a linear combination of its past values. p^{th} order Auto-regressive model is written as [11]:

$$y_t = a_1 y_{t-1} + a_2 y_{t-2} + \ldots + a_p y_{t-p} + \epsilon_t \tag{2.1}$$

$$y_t = \epsilon_t + \sum_{x=1}^{p} a_x y_{t-x} \tag{2.2}$$

Where ϵ_t is Gaussian White Noise.

MA Mathematical Model

The q^{th} order moving average model relates noise to the current output as the following [11]:

$$y_t = \epsilon_t + b_1 \epsilon_{t-1} + b_2 \epsilon_{t-2} + \ldots + + b_q \epsilon_{t-q} \tag{2.3}$$

$$y_t = \epsilon_t + \sum_{x=1}^{q} b_x \epsilon_{t-x} \tag{2.4}$$

Where ϵ_t is Gaussian White Noise.

ARMA Mathematical Model

ARMA(p,q) combines the two models illustrated above, yielding [11]:

$$y_t = \epsilon_t + \sum_{x=1}^{p} a_x y_{t-x} + \sum_{m=1}^{q} b_m \epsilon_{t-m} \tag{2.5}$$

Where ϵ_t is Gaussian White Noise.

Forecast

If the above mathematical model is assumed to be linear, then the following can be obtained [11]:

$$
\begin{aligned}
y_t &= \epsilon_t + \sum_{x=1}^{p} a_x y_{t-x} + \sum_{m=1}^{q} b_m \epsilon_{t-m} \\
y_{t+1} &= \epsilon_{t+1} + \sum_{x=1}^{p} a_x y_{t-x+1} + \sum_{m=1}^{q} b_m \epsilon_{t-m+1} \\
&\vdots \\
y_{t+h} &= \epsilon_{t+h} + \sum_{x=1}^{p} a_x y_{t-x+h} + \sum_{m=1}^{q} b_m \epsilon_{t-m+h}
\end{aligned}
$$

It is observed that each equation in future can be expressed in terms of the past y and future ϵ values, as a result the equation in future can be re-written in the form of current y values and future ϵ values.

Model Co-efficient Calculation

It is evident from above observations that such mathematical model is able to predict linearly about the future value, however co-efficient a_x and b_x still need to be determined to complete the model.

Ordinary least squares (OLS) is used to calculate the co-efficients of the model. The technique aims to find a linear best-fit line to represent the trend of a collection of scattered data. The idea can be expressed in the following equations. [13] [1]

$$
\begin{aligned}
y_i &= \alpha + \sum_{m=0}^{n} \beta_m x_{i-m} + \epsilon_i \\
\epsilon_i &= y_i - (\alpha + \sum_{m=0}^{n} \beta_m x_{i-m}) = y_i - \hat{y}_i
\end{aligned}
$$

Where y_i denotes the actual value of i^{th} data in the model, $\hat{y}$ is the estimated i^{th} value, α is the y-intercept of the line fitted, β is the gradient of the fitted line, and ϵ_i is the vertical distance between the fitted line and the actual data at i. All unknown parameters can be obtained using the following condition:

$$\min \sum_{i=1}^{n} (\epsilon_i)^2$$

A graphical representation is included in 2.1.

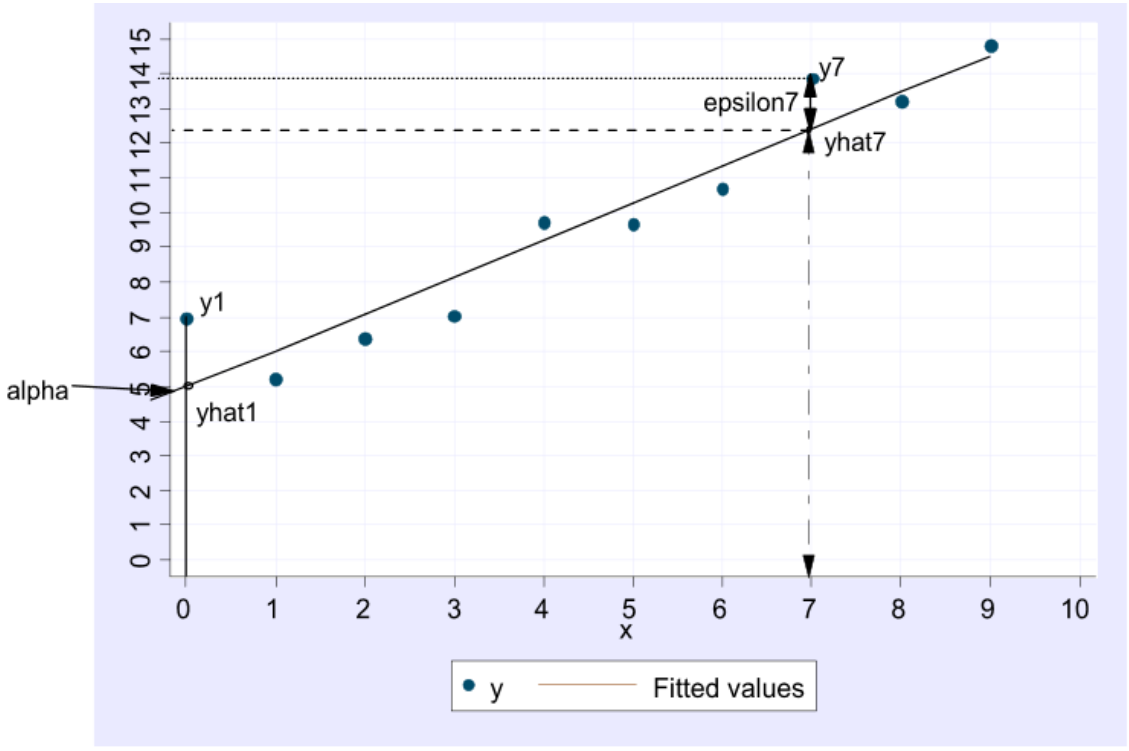


Figure 2.1: Graphical representation of OLS method [1]

2.2.2 Kalman Filter

Kalman Filter is a recursive predictive filter that operates on a linear state space model. It is used to estimate the current and future state of a state space model when there are noises present. The Kalman filter can be classified into two distinct phases, namely Prediction and Correction. The filter firstly predicts the state of

the state space model and then correct the prediction in the second phase according to the observation model. The process is repeated to minimise the error covariance of the estimator [14] [15].

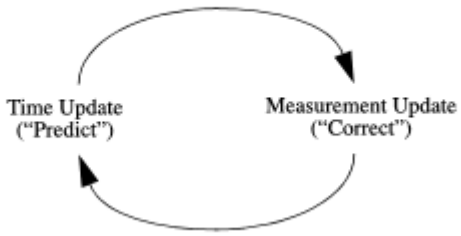


Figure 2.2: Graphical representation of process of Kalman Filter [2]

State Space Model

A linear discrete state space model is a mathematical model that relates the future states with the current states, input, and other externalities. The states are the physical attributes of interest in a dynamic system model, for example, velocity, position and etc. The state space model is expressed as [2]:

$$x_{k|k-1} = Ax_{k-1|k-1} + Bu_{k-1|k-1} + w_{k-1|k-1} \tag{2.6}$$

The measurement of states is then:

$$z_{k|k-1} = Hx_{k-1|k-1} + v_{k-1|k-1} \tag{2.7}$$

Where x denotes all states of interest in matrix form, A is the relationship between current states and the next state without externalities, u is a known input vector, B is the control input model that maps inputs to states, H is the observation model that maps true state space to observed state space, and w, v are Gaussian white noises with co-variances Q and R respectively.

$$P(w) \sim N(0, Q) \tag{2.8}$$

$$P(v) \sim N(0, R) \tag{2.9}$$

Prediction

The prediction phase can be described with the following equations [2]:

$$\hat{x_k^-} = A\hat{x}_{k-1} + Bu_{k-1} \tag{2.10}$$

$$P_k^- = AP_{k-1}A^T + Q \tag{2.11}$$

Where $\hat{x}^-$ denotes the *a priori* estimate, $\hat{x}$ is the *a posteriori* estimate, P^- is the *a priori* estimate error covariance, P is the *a posteriori* estimate error covariance. It should be noted that the relationship between $\hat{x}^-$ and $\hat{x}$ is expressed as [2]:

$$\hat{x}_k = \hat{x}_k^- + K(z_k - H\hat{x}_k^-) \tag{2.12}$$

It can be observed that $(z_k - H\hat{x}_k^-)$ represents the difference between the predicted measurement $H\hat{x}_k^-$ and the true measurement z_k, and this is called the residual. K is the Kalman gain that minimises the *a posteriori* error covariance P. This is evident in the Correction phase.

Correction

The Correction phase is governed by the following equations [2]:

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1} \tag{2.13}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-) \tag{2.14}$$

$$P_k = (I - K_k H)P_k^- \tag{2.15}$$

It can be observed that the correction phase use the *a priori* estimate and covariance matrix to calculate the *a posteriori* estimate and covariance matrix, which would then be used to calculate the *a priori* estimate and covariance matrix of the next state. Such recursive approach can be used to predict the future states if it is assumed that the current state can be linearly projected to the next state. The overall process can then be illustrated in 2.3.

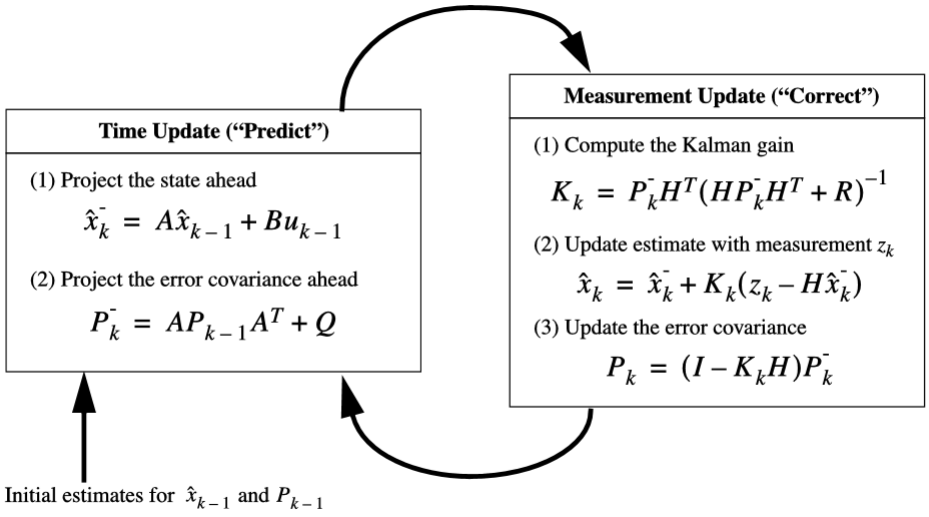


Figure 2.3: Detailed Graphical representation of process of Kalman Filter [2]

2.3 RAT selection for multi-homed mobile terminals

While it is beneficial for a mobile terminal to be capable of connecting to multiple network interfaces at the same time, it also greatly increases the complexity of selecting the optimal sets of networks.

There are numerous parameters (signal strength, bandwidth,etc.) that could potentially influence the user satisfaction. Choosing the "best" network usually involves trade-off amongst the different parameters, and potential solutions are usually plenty, therefore it is of great interest to establish methods to quantify the importance of parameters so that a digital processing unit is able to make a optimal network selection. Different approaches of selecting the optimal networks are investigated in the section.

2.3.1 MADM

MADM stands for Multiple Attribute Decision Making, which describes a systematic process of making a decision based on the importance of multiple attributes or parameters.

A MADM process can be formulated as the following [16]:

$$S = \{s_i, i = 1, 2, 3, \ldots n\} \tag{2.16}$$

Where S represents a finite set of potential solutions for network selection;

$$A = \{a_j, j = 1, 2, 3, \ldots m\} \tag{2.17}$$

Where A represents a finite set of attributes concerned in the MADM process. The attributes involved in a network selection scenario could include power consumption, bandwidth, cost, network congestion, etc.

$$W = \{w_j, j = 1, 2, 3, \ldots m\} \tag{2.18}$$

Where W represents a set of normalised weights as proposed in 2.20, which indicates the relative importance of each attribute.

$$X = \{x_{ij}, i = 1, 2, 3, \ldots n, j = 1, 2, 3, \ldots m\} \tag{2.19}$$

X represents the set of score or utility of an attribute of a potential solution. x_{ij} indicates the score of attribute a_j of potential solution s_i. All information can be collected into Table II.

Table II: MADM Matrix

	a_1	a_2	.	a_m
s_1	x_{11}	x_{12}	.	x_{1m}
s_2	x_{21}	x_{22}	.	x_{2m}
.	.	.	.	.
s_n	x_{n1}	x_{n2}	.	x_{nm}

As the problem is formulated above, the fundamental idea of MADM is to find an aggregate score of all attributes of a potential solution, and then rank all possible solutions according to the aggregate score obtained, finally the alternative with highest score would be deemed as the optimal solution. It should be noted that all scores must be normalised to be comparable to each other. Different approaches of calculating the aggregate score, normalisation of scores and weight are discussed in the following sections.

Normalisation

Although attributes or parameters are measured in different units, however it is possible to compare one attribute to another through the use of normalisation. The normalisation formulas modified from [17] are tabulated in Table III.

Table III: Different normalisation methods

Type \ Method	Simple	Linear	Vector
Benefit x_j	$r_{ij} = \frac{x_{ij}}{x_j^*},\ x_{ij} > 0$	$r_{ij} = 1 - \frac{x_j^* - x_{ij}}{x_j^* - x_j^-},\ x_j^* \neq x_j^-$	$r_{ij} = \frac{x_{ij}}{\sqrt{\sum_{i=1}^{n} x_{ij}^2}}, x_j^* > 0$
Cost x_j^c	$r_{ij}^c = \frac{x_j^-}{x_{ij}}, x_{ij} > 0$	$r_{ij}^c = 1 - \frac{x_{ij} - x_j^-}{x_j^* - x_j^-}, x_j^* \neq x_j^-$	$r_{ij}^c = \frac{1}{x_{ij}\sqrt{\sum_{i=1}^{n} (\frac{1}{x_{ij}})^2}}, \max_i \frac{1}{x_{ij}} > 0$

where $x_j^* = \max_i x_{ij}$, which is the maximum value of j^{th} attribute in all potential solutions. $x_j^- = \min_i x_{ij}$, which is the minimum value of j^{th} attribute in all potential solutions.

It should be noted that $r_{ij} \in [0, 1]$, with 0 being the lowest possible score and 1 being the highest possible score. The score boundary are obtained by assuming that maximising x_{ij} for benefit and cost would result in the highest possible score and lowest possible score respectively, and vice versa. This can be observed by the following relationships.

For Benefit x_j:

For all methods:

$$x_{ij} = x_j^*, n = 1 \Rightarrow r_{ij} = 1$$

For Simple method:

$$x_{ij} = x_j^-, \lim_{x_{ij} \to 0} r_{ij} \to 0$$

For linear and vector method:

$$x_{ij} = x_j^- = 0 \Rightarrow r_{ij} = 0$$

For cost x_j^c:

For all methods:

$$x_{ij} = x_j^-, n = 1 \Rightarrow r_{ij}^c = 1$$

For simple method:

$$x_{ij} = x_j^* \Rightarrow \lim_{x_{ij} \to \infty} r_{ij}^c \to 0$$

For linear method:

$$x_{ij} = x_j^* \Rightarrow r_{ij}^c = 0$$

For vector method:

$$x_{ij} = x_j^*, n \neq 1 \Rightarrow \lim_{x_{ij} \to \infty} r_{ij}^c \to 0$$

To normalise the weight vector, the following is used [18]:

$$w_j = \frac{w_j^a}{\sum_{x=1}^{m} w_x^a}, j = 1, 2, ...m \tag{2.20}$$

where w_j^a represents the actual weight for j^{th} attribute before normalisation, and w_j denotes the normalised weight of j^{th} attribute.

SAW

SAW stands for Simple Additive Weighting, this is probably the most intuitive method. It calculates the weighted sum of the scores of all attributes in a potential solution, and then select the solution with the highest aggregate score. The method to calculate the aggregate score of a potential solution can therefore be expressed as the following [16]:

$$S_{SAW} = \max_i \sum_{j=1}^{m} x_{ij} \times w_j \tag{2.21}$$

Where S_{SAW} is the solution investigated.

WP

WP stands for weighted product, this approach has similar idea to that of SAW. It calculates the product of exponentially weighted scores of attributes instead of sum of weighted scores of attributes. The mathematical expression is as follows [16]:

$$S_{WP} = \max_i \prod_{j=1}^{m} x_{ij}^{w_j} \tag{2.22}$$

Where S_{Wp} is the solution selected.

TOPSIS

TOPSIS stands for Technique for Order of Preference by Similarity to Ideal Solution, which is probably the most widely used algorithm for interface selection at the moment. As the name suggests, it selects the solution which is closest to the positive ideal solution and farthest from the negative ideal solution. The algorithm, based on [16], is implemented in the following steps:

1. Construct a normalised score matrix R for benefit and cost attributes according to the vector normalisation method provided in Table III.

2. Construct a weighted normalised score matrix V, where each entry in V is the product of the normalised weight and normalised score.

$$V = \begin{bmatrix} v_{11} & v_{12} & . & v_{1m} \\ v_{21} & v_{22} & . & v_{2m} \\ . & . & . & . \\ v_{n1} & v_{n2} & . & v_{nm} \end{bmatrix}$$

$$V = \begin{bmatrix} r_{11} \times w_1 & r_{12} \times w_2 & . & r_{1m} \times w_m \\ r_{21} \times w_1 & r_{22} \times w_2 & . & r_{2m} \times w_m \\ . & . & . & . \\ r_{n1} \times w_1 & r_{n2} \times w_2 & . & r_{nm} \times w_m \end{bmatrix}$$

3. Establish the positive ideal and negative ideal solutions. The attributes with the highest score in its category form the positive ideal solution, and

the attributes with the lowest score in its category form the negative ideal solution.

$$A^+ = \max_j[v_{ij}] = [v_1^+, v_2^+, \ldots, v_m^+] \tag{2.23}$$

$$A^- = \min_j[v_{ij}] = [v_1^-, v_2^-, \ldots, v_m^-] \tag{2.24}$$

Where A^+ denotes a set of maximum weighted score for each attribute, forming the ideal solution, A^- denotes a set of minimum weighted score for each attribute, forming the negative ideal solution.

4. Calculate the m-dimensional Euclidean distance from the potential solution to the positive/negative ideal scores as the following:

$$S_j^+ = \sqrt{\sum_{j=1}^{m}(v_{ij} - v_j^+)^2} \tag{2.25}$$

Where S_j^+ denotes the m-dimensional Euclidean distance between the alternative and the positive ideal solution.

$$S_j^- = \sqrt{\sum_{j=1}^{m}(v_{ij} - v_j^-)^2} \tag{2.26}$$

Where S_j^- denotes the m-dimensional Euclidean distance between the alternative and the negative ideal solution.

5. Calculate the closeness index C_j:

$$C_j = \frac{S_j^-}{S_j^- + S_j^+} \tag{2.27}$$

It should be noted that $C_j \in [0, 1]$, and C_j approaches to 1 as the alternative moves further way from the negative ideal solution, and closer to the positive ideal solution. It can therefore be established that better alternatives result in a bigger C_j.

6. Rank all potential solutions according to the descending order of C_j, the alternative with the biggest C_j will be the optimal solution.

Drawbacks

In [16], Nguyen *et al.* had pointed out that each method discussed above have shortcomings. The simulation showed that TOPSIS is prone to "ranking abnormality" while SAW and WP are less accurate in identifying the most suitable candidate compared to that of TOPSIS.

Ranking abnormality refers to the change of order of rankings when a low ranking alternative is removed or replaced by another low ranking candidate. This is not desirable as this introduces inconsistency to the algorithm [16].

It was shown in simulations that WP and SAW produced very similar scores for different alternatives, which leads to the difficulty of claiming one alternative to be the clear optimal solution of a situation.

2.3.2 Policy-based

In [3], a dynamic policy based interface selection algorithm is proposed. The decisions are based on a set of policies or rules defined by the user. [3] has proposed five basic components to a policy-based interface selection:

1. Entities: are those who carry out actions, for example, a user, peer node, etc.

2. Actions: are defined by the entities, and are controlled by the system, it indirectly suggests the type of interface that should be connected in order to meet the requirement of the action.

3. Policy: governs the action of an entity. Although the entities can define multiple policies for multiple scenarios, however only one action can take place in a policy.

4. Credential: Used to authorise the actions.

5. Mechanism: Evaluates the information about the connection and select the optimal interface according to the connection information.

The overview of interface mechanism is in 2.4. The obvious drawback of this approach is that a policy must be defined for each action, which could become very laborious and complicated once the problem size grows.

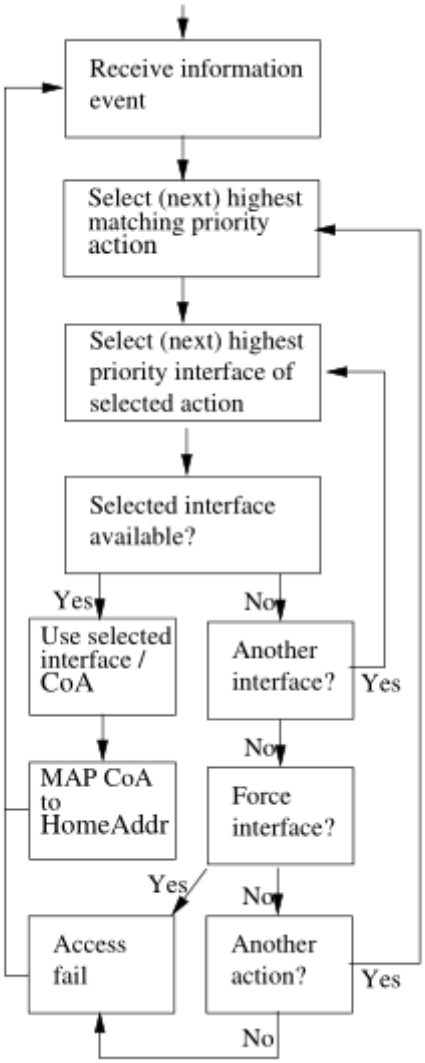


Figure 2.4: Policy based Interface Selection Mechanism [3]

Chapter 3

System Modelling

The entire process of network selection is modelled using the principals of discrete event simulation and queueing theory. It should be noted that all quantities are dimensionless unless otherwise stated.

3.1 Arrival Rate Modelling

The call arrival rate of the network in a time interval is modelled with a Poisson distribution that is expressed as the following:

$$P(k) = \frac{\lambda^k e^{-\lambda}}{k!} \tag{3.1}$$

where:

- λ is the average number of events per time interval
- k is the number of events that occur during the time interval, $k \in \mathbb{W}$

It should be noted that the inter-event time of Poisson process follows an exponential distribution.

3.2 Usage Time Modelling

The usage time of each call is modelled with an exponential distribution whose probability density function is given by the following formula:

$$P(x) = \begin{cases} \lambda e^{-\lambda x} & x \geq 0 \\ 0 & x < 0 \end{cases} \tag{3.2}$$

Where:

- λ is the rate parameter
- x is the duration of event

3.3 Network Infrastructure Modelling

The modelling of RAT can be divided into the following categories.

3.3.1 Physical attributes of the RAT

Each RAT has the following attributes modelled to facilitate the decision making process:

- Power dissipation per bbu
- Monetary cost per bbu for subscriber
- Total capacity of the network in bbu
- Maximum amount of bandwidth that can be allocated to one user
- Guard capacity of CAC in bbu
- Coverage radius
- Position of RAT

3.3.2 Coverage of Network

The coverage of a network is assumed to be a circle with radius r, signal strength is assumed to decrease linearly as the user is further away from the tower.

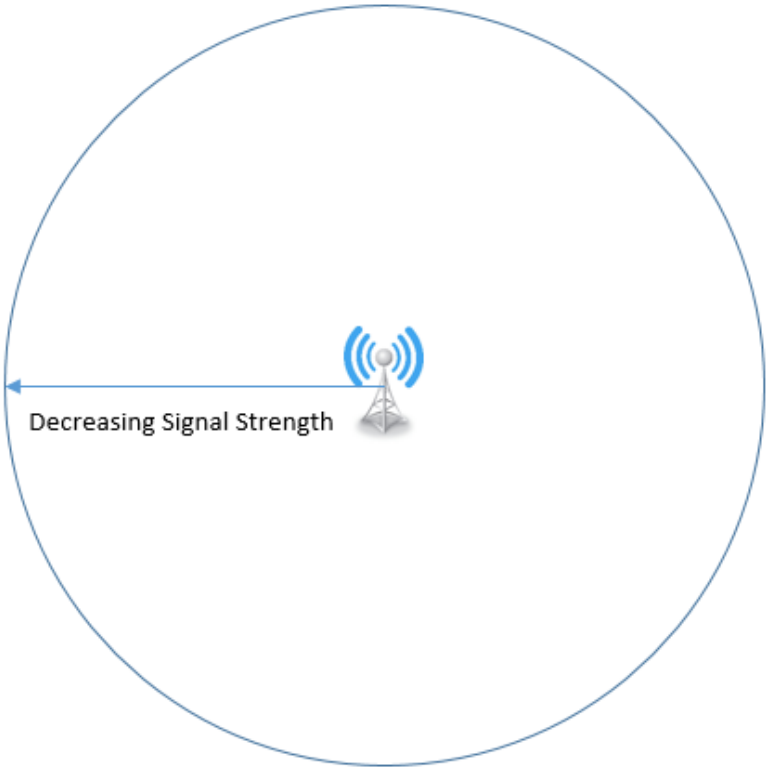


Figure 3.1: Coverage of a transceiver

3.3.3 Capacity and CAC

The total capacity of a RAT is expressed in terms of base band units (bbu), and the RAT can only provide a limited amount of bbu to a single user due to physical limitation of the technology. CAC stands for call admission control, which determines whether or not a call would be accepted by the network. In this model, RAT only accepts hand-off calls after a certain number of bbu are in use. The concept is illustrated in 3.2

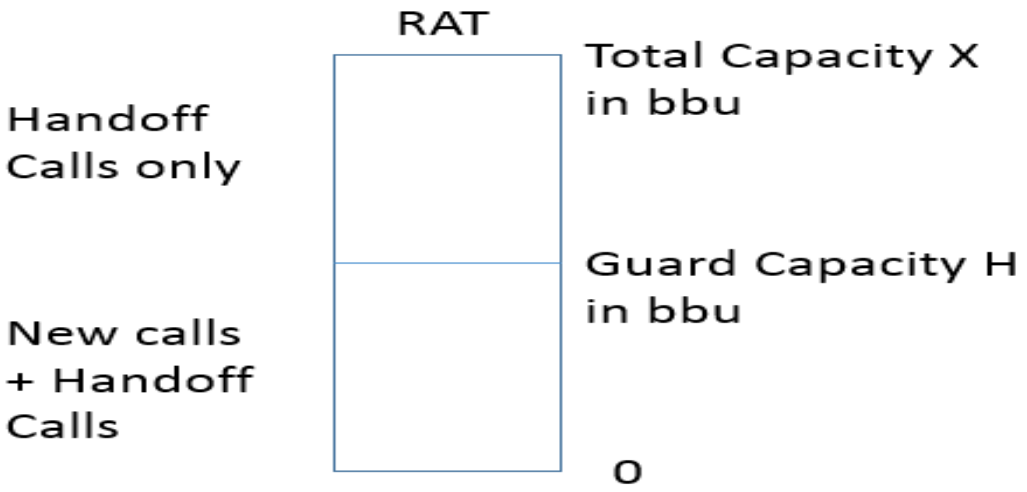


Figure 3.2: CAC policy

The decision making aspect of the CAC algorithm can be represented by the following flow diagram:

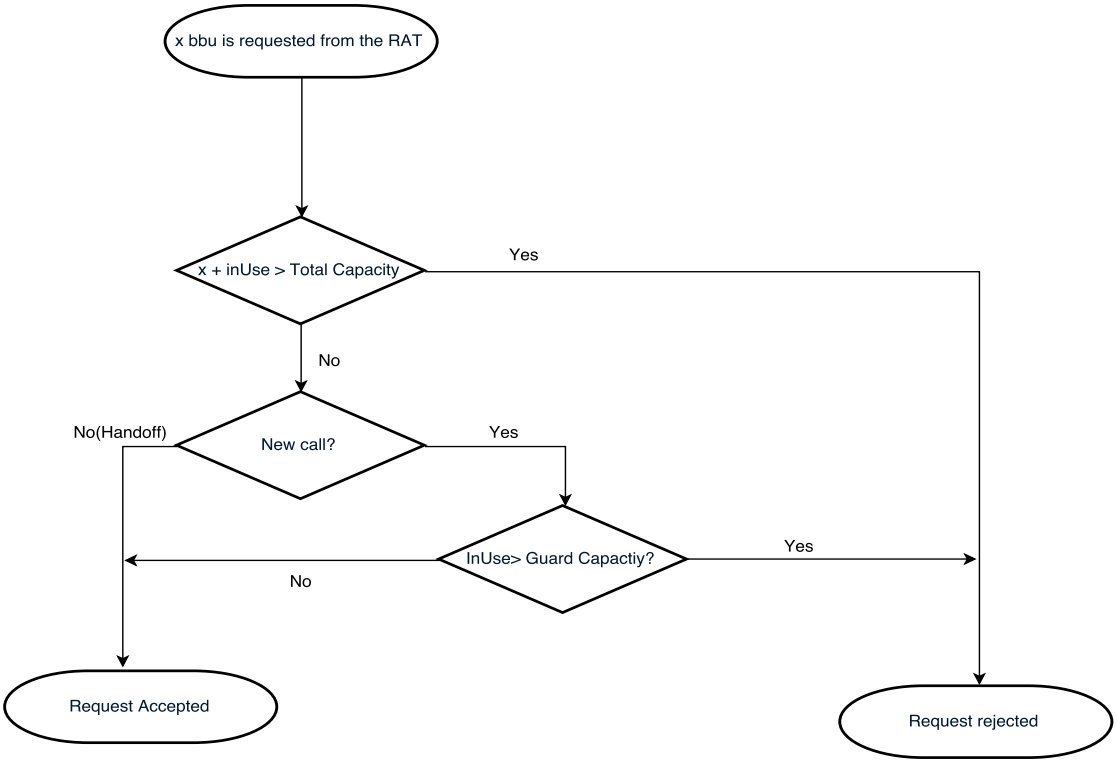


Figure 3.3: Flow chart of CAC

3.4 Traffic and Utility Function Modelling

Based on the analysis given in [4] [19], the traffic is classified in Table I and the utility functions are given in Figure 3.4. It should be noted that a design parameter called rate multiplier M, which determines the maximum bandwidth allocated to an adaptive application, is introduced to facilitate the modelling of adaptive applications. M can be interpreted as the QoS that the user desired, e.g. video resolution.

Table I: Different types of traffic [4]

Application type	Example	Service class	Necessary bandwidth (b_{req})
Elastic	SSH, Instant Messenger, HTTP	Interactive	limitless
Elastic	FTP, Email	Background	limitless
Adaptive	CD-like Audio	Streaming	150 kbps
Adaptive	MPEG4 Video	Streaming	2000 kbps
Hard-Real time	PCM VoIP	Conversation	64 kbps
Hard-Real time	H.323 VoIP	Conversation	20 kbps

The utility functions of different types of application are expressed in 3.4:

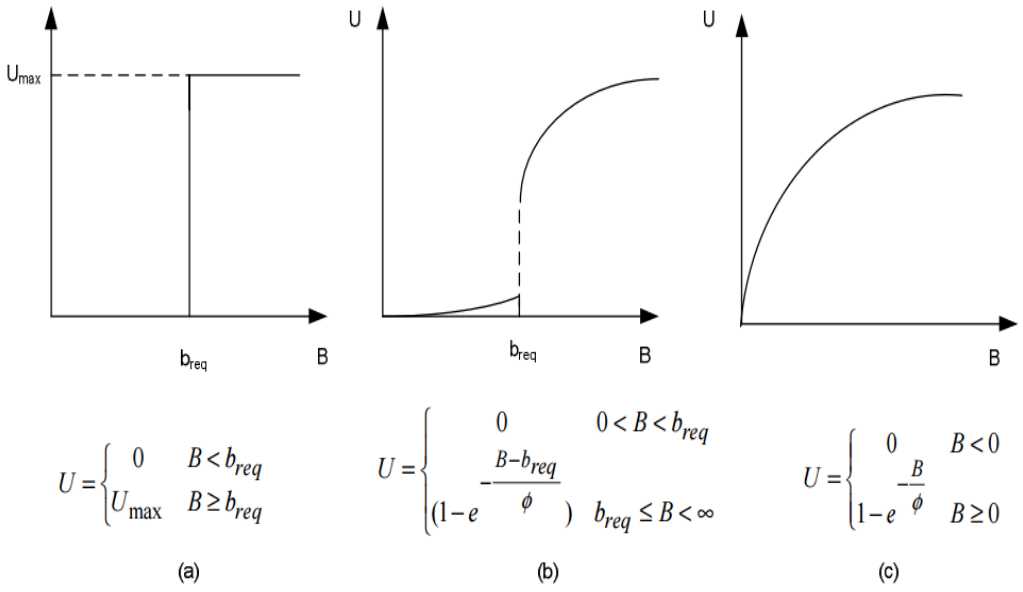


Figure 3.4: Utility functions of different applications [4]

(a) shows the utility function of a real-time application which requires the data to arrive within a certain delay, and such application needs at least b_{req} to function, however surplus bandwidth would not improve the user satisfaction. (b) represents the utility function of a rate-adaptive or delay-adaptive application. This type of application would not yield any user satisfaction when bandwidth is less than b_{req} because the service can not be guaranteed when required bandwidth is not met, and surplus bandwidth would improve the user experience. (c) represents the the utility function of an elastic service which does not have a required bandwidth and the user satisfaction is proportional to the bandwidth available. ϕ is a design parameter that will be chosen during the deigns process.

3.5 Signal Strength Modelling

Signal strength has an impact on the power dissipation of the device and speed of the network. In this particular model, the signal strength has a negative linear relationship with the distance between the transceiver and mobile host. The

signal strength is served as a penalty factor for the bandwidth utility and power dissipation of the mobile host. The factor is calculated as the following:

$$p = 1 - \frac{D_{th}}{D_c}$$

Where:

- p is the penalty factor, and $p \in [0, 1]$
- D_{th} is the distance between the mobile host and the transceiver
- D_c is the radius of transceiver's coverage

The effect of p on bandwidth utility and power dissipation can be expressed as the following:

$$U = U_{before} \times p \tag{3.3}$$

$$P = P_{req} + (1 - p) \times P_e \tag{3.4}$$

Where:

- U denotes the amount of bandwidth utility after taking signal strength into consideration
- U_{before} denotes the amount of bandwidth utility without signal strength penalties
- P is the power dissipation after taking signal strength into consideration
- P_{req} is the amount of power dissipation if there is perfect reception of signal
- P_e is the surplus amount of power dissipation when the mobile host barely receives any signal

3.6 Battery Modelling

The battery of the mobile host is modelled as a value between 1 and 100, where 100 indicates full battery and 1 indicates a flat battery. The amount of battery left affects the weighting of power dissipation in the selection process, the effect can be represented in the following equation:

$$W_{power} = W_{full} + (1 - \frac{B}{100})W_{extra} \tag{3.5}$$

Where:

- W_{power} is the final weight after taking battery into consideration
- W_{full} is the weight power dissipation carries when the battery is full
- B is the battery value between 1 and 100
- W_{extra} is the maximum amount of extra weight that can be contributed by battery

3.7 User Movement Estimation and Prediction Modelling

An arbitrary point in a 2-D Cartesian plane system is chosen as the origin $(0, 0)$. The position of a user, who is spawned at a random location, is represented as a 2-D vector (a, b). Kalman filter is selected as the method to estimate the true position and predict future positions of the user, relevant equations are found in Figure 2.3. Three types of movements are investigated in the movement model, namely straight line movement, turning movement and controlled movement.

3.7.1 Straight Line Movement

The straight line movement is governed by the following relationship:

$$P_0 = (x, y)$$
$$P_1 = (x_1, y_1)$$
$$D = \frac{P_1 - P_0}{|P_1 - P_0|}$$

Where:

- P_0 is the initial position of user
- P_1 is the next position of user
- D is the unit vector that represents the direction of user

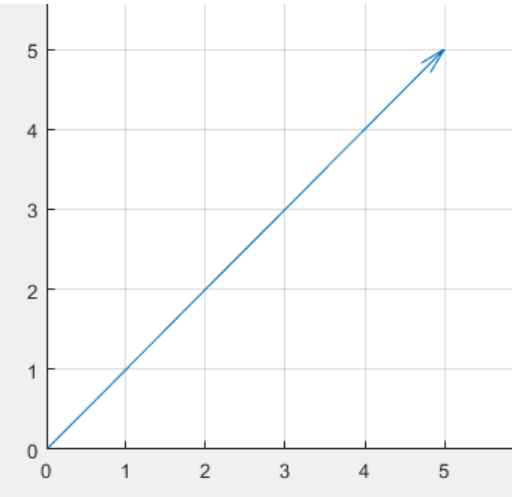


Figure 3.5: Vector Representation of Straight Line Movement

- D stays constant throughout the movement, it should be noted that direction and duration of the movement are random values that are normally distributed.

3.7.2 Turning Movement

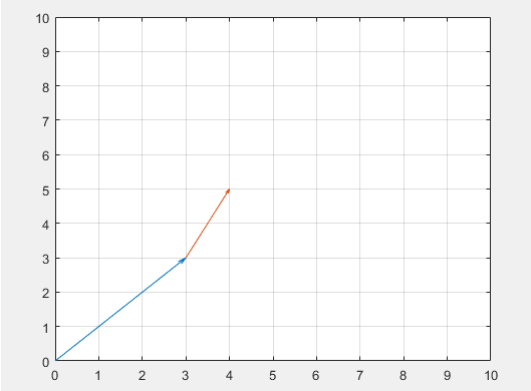


Figure 3.6: Vector Representation of Turning Movement

The turning movement is modelled as a combination of straight line movement in different directions. it should be noted that direction and duration of the

movement are modelled as random values that are normally distributed.

3.7.3 Controlled Movement

The movement is essentially a turning movement whose direction and duration can be controlled during the simulation.

3.7.4 Implementation of Kalman Filter

Based on equations given in Figure 2.3, the matrices are defined as the following:

$$x = \begin{bmatrix} Position \\ Velocity \end{bmatrix}, A = \begin{bmatrix} 1 & dt \\ 0 & 1 \end{bmatrix}, B = 0$$

$$H = \begin{bmatrix} 1 & 0 \end{bmatrix}, Q = \begin{bmatrix} \frac{dt^4}{4} & \frac{dt^3}{2} \\ \frac{dt^3}{2} & dt^2 \end{bmatrix}, P_0 = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, R = MN^2$$

Where:

- x are the quantities that are chosen as the state variables
- dt is the sampling period of the system
- P_0 is the initial error covariance matrix
- MN is the measurement noise

A prediction window of size p is chosen as a parameter, which results in p predictions of the future location. The process yields the following as outputs:

$$x_{prediction} = \{x_1, x_2, \ldots, x_p\} \tag{3.6}$$

$$y_{prediction} = \{y_1, y_2, \ldots, y_p\} \tag{3.7}$$

Where x, y are vector components of current and predicted positions.

3.7.5 Prediction Model Validation

The prediction model is validated by investigating the following indicators:

- Difference between the estimated position and true position with presence of measurement noise
- Difference between the predicted positions and the true positions in future.

Estimated VS True Position

The investigation is aimed to reveal the relationship between changes in parameters and accuracy of the model. The following graph shows the difference between the true position and estimated position with different sampling frequency and measurement noise of different magnitude. The result is obtained by tracking the movement of an arbitrary user. The type of movement investigated and shown is turning movement because it is assumed that the user would travel in the same direction for a period that is much longer in comparison with the sampling period before changing to another direction. Code used to simulate the scenario is adapted from [20].

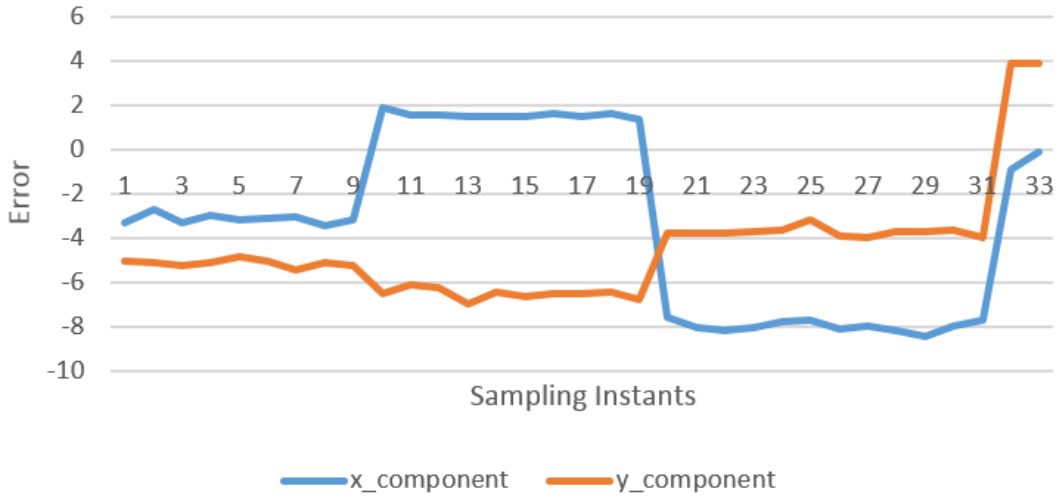


Figure 3.7: Error when measurement noise is 10 and sampling period is 0.5

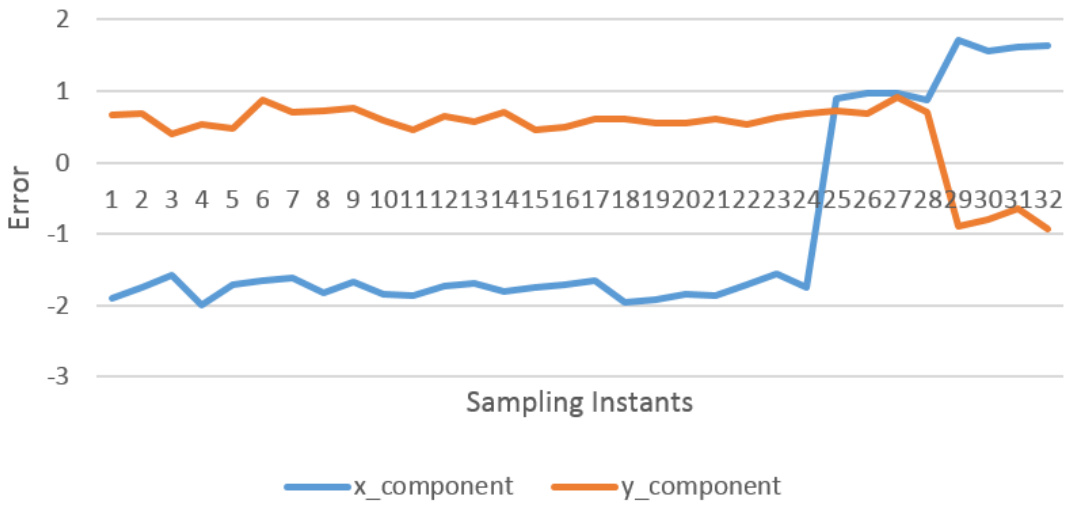


Figure 3.8: Error when measurement noise is 10 and sampling period is 0.1

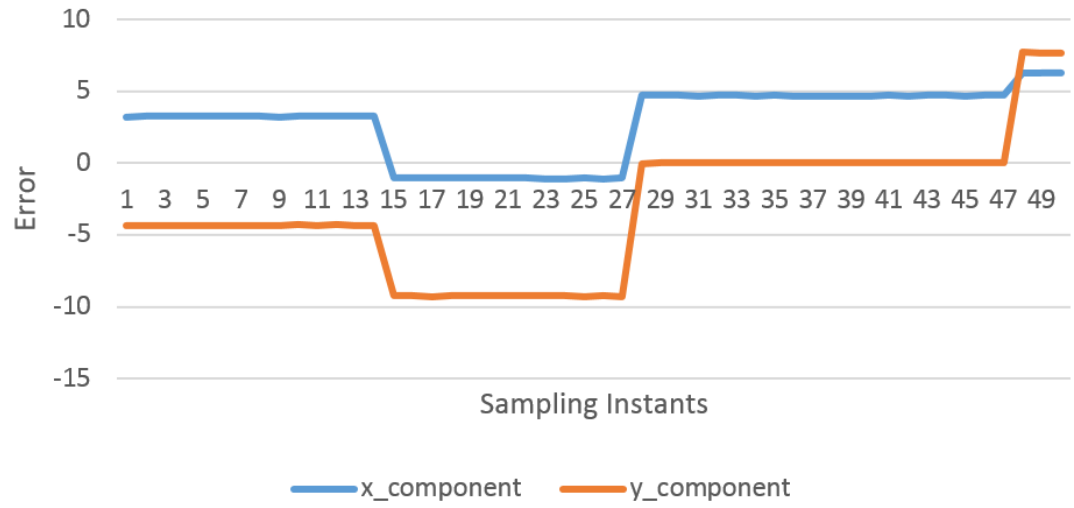


Figure 3.9: Error when measurement noise is 100 and sampling period is 0.5

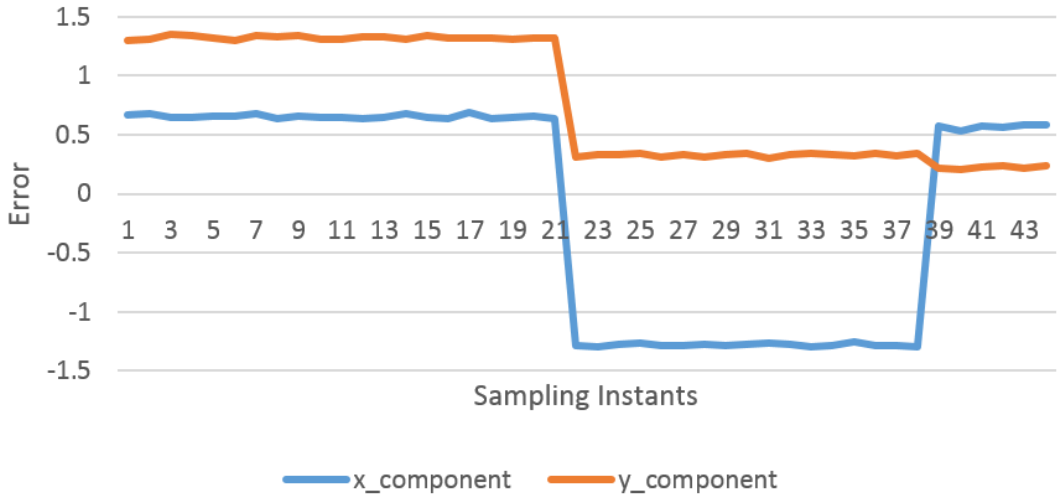


Figure 3.10: Error when measurement noise is 100 and sampling period is 0.1

The relevant indicators are tabulated as follows:

Table II: Relationship between parameters and accuracy of movement model

	Average Error of x	Average Error of y	$\lvert\frac{Error_x}{Noise}\rvert$	$\lvert\frac{Error_y}{Noise}\rvert$
$MN = 10, dt = 0.5$	-3.31	-4.48	0.33	0.45
$MN = 10, dt = 0.1$	-1	0.44	0.1	0.04
$MN = 100, dt = 0.5$	2.88	-3.15	0.29	0.32
$MN = 100, dt = 0.1$	-0.1	0.79	0.01	0.08

It should be noted that the values in the above observations are dimensionless, the quantities can be interpreted differently depending on the context. From the above table, it can be observed that the magnitude of error is correlated to the sampling frequency of the measurement process. Magnitude of measurement noise has a very weak, if at all, influence on the error of the estimation.

Prediction VS Actual Future Position

Using the same set of data as above, the accuracy of the prediction model is evaluated by calculating the error between the actual future positions and predicted positions:

$$e_k = \hat{x}_{t+k|t} - x_{t+k} \tag{3.8}$$

Where:

- e is the error of prediction
- $\hat{x}$ is the predicted position
- x is the actual future position of the user

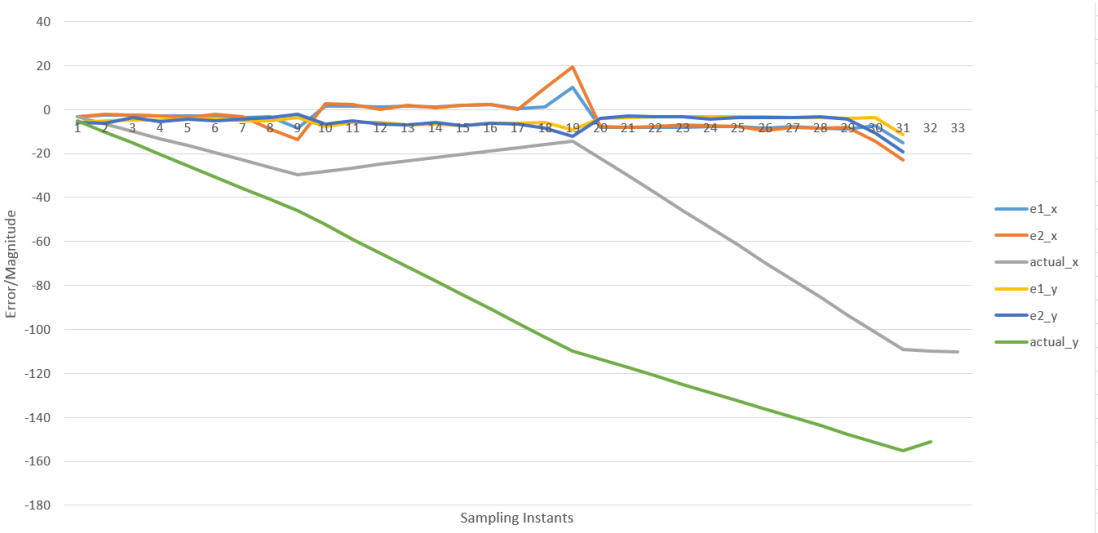


Figure 3.11: Error of prediction and actual position

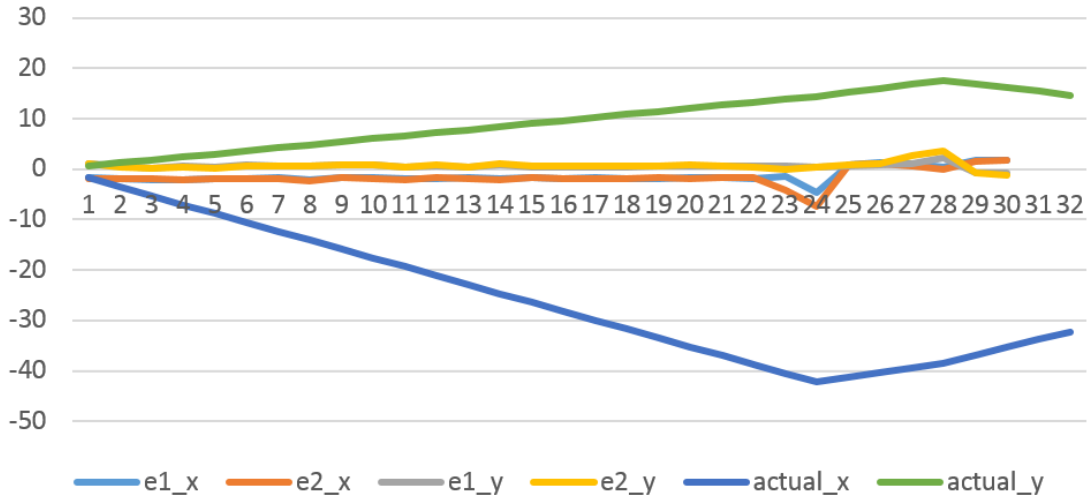


Figure 3.12: Error of prediction and actual position

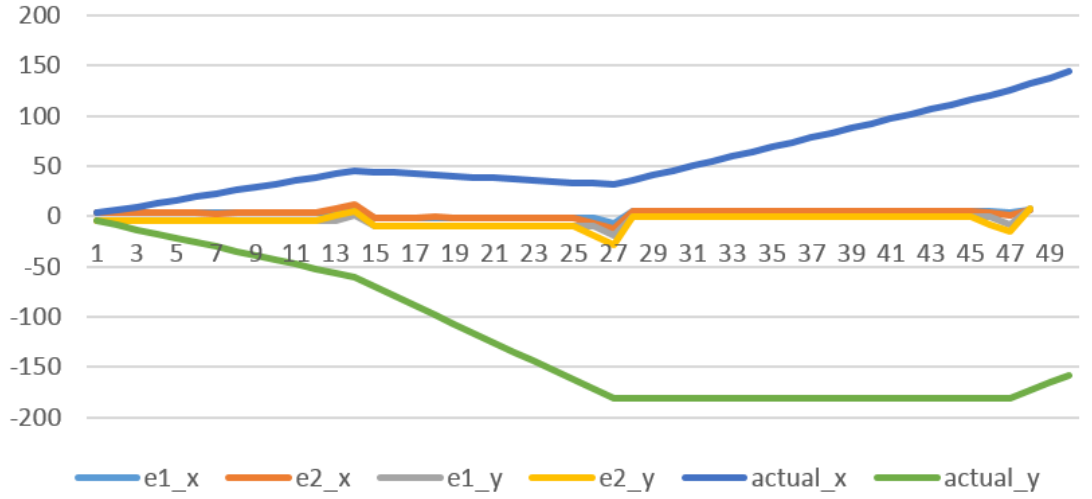


Figure 3.13: Error of prediction and actual position

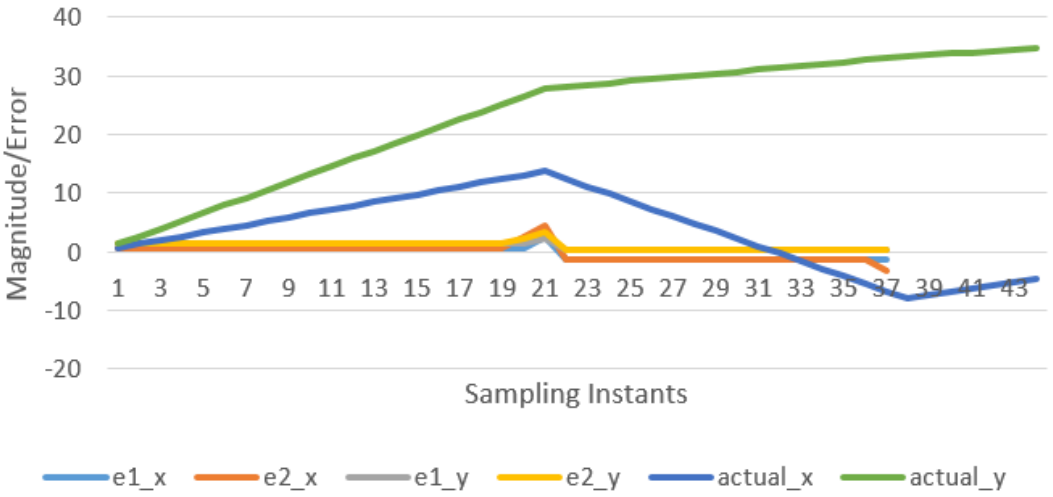


Figure 3.14: Error of prediction and actual position

Based on the data above, the following is evident:

- A small spike in the magnitude of error occurs when the user changes direction, and it takes 1 sampling instant to correct the prediction.
- The accuracy of prediction is greatly enhanced if the sampling frequency is high.
- After taking the measurement noise into consideration, it is evident that the prediction model provides a very accurate approximation of the future position of the user when the sampling frequency is high.

Chapter 4

Design

All simulations are done in JAVA with the help of Apache Commons Library. Code used in simulation can be found in appendix.

4.1 Design Parameters

The following is a list of parameters mentioned in the modelling section. If the value of the parameter is not indicated in the list, it means that the value could vary due to the need of experiment.

- Measurement Noise $MN = 10$
- $\lambda = 2$ for equation 3.1 and 3.2
- $\phi = 1000000$ for equations in figure 3.4
- The maximum bbu that can be allocated to a single user is:
  - 3G: 2000 bbu
  - 4G: 1000000 bbu
  - 5G: 10000000 bbu
  - WIFI: 54000 bbu
- Sampling frequency
- Power cost per bbu

- Monetary cost per bbu
- Guard capacity of the network H as in 3.2
- Prediction window: It is the number of locations predicted
- Uncertainty radius r_{err}: It is to reduce the effect of error in Kalman Filter
- Weight vector for MADM
- W_{extra} in equation 3.5
- P_e in equation 3.4
- rate multiplier M: It determines the maximum amount of bandwidth that will be allocated to an adaptive application

4.2 Algorithm Design

This section is to provide a detailed interpretation of all algorithms involved in the network selection process, and an overview of the entire system will be given to clarify the logic flow of the design.

4.2.1 Candidate Criterion

Candidates refer to the RAT or combination of RATs that will be considered in the decision making process, these candidates can be represented as the vector in Equation 2.16. In order for a RAT to be qualified as a candidate, the following criterion must be met:

- The user must be within the coverage of the RAT
- The CAC process, as shown in Figure 3.3, will allow the allocation of bandwidth requested to the user

It should be noted that there are p predictions of future positions of the user (shown in Section 3.7.4), therefore such criterion must be applied to each predicted position, which then results p lists of candidates.

$$Candidates = \{\{RAT_1, RAT_2 \dots\}, \dots \{RAT_1, RAT_2 ...\}\} \tag{4.1}$$

It should be noted that each list of candidates may not be of equal size as it is possible that the user would move in or out of the coverage of RATs in the predicted future. The algorithm therefore removes all RATs, that are not present in all candidate lists or not in range at the current position, to alleviate potential ping-pong effect due to the movement of user. It should be noted that size of p here indicates the level of conservativeness of the system with regards to connecting to a certain set of interfaces. Large p would result in a more conservative mentality and vice versa. If the user is predicted to move out of coverage of all RATs, i.e. the candidate list is empty, the algorithm would try to maintain the current connection, however if the current connection can not be maintained, then the network selection process would be based on the current position without taking predictions into consideration.

It is acknowledged in the validation of Kalman Filter in Section 3.7.5, that noticeable errors occur as a result of the prediction process, which needs to be dealt with when the candidate lists are created. An uncertainty radius r_{err} is introduced to alleviate the problem, which essentially reduced the coverage of RAT perceived by the algorithm. The user is considered to be in range of the RAT when the entire circle with radius r_{err} is within the coverage. Similar to p, this parameter can also be used to adjust the conservativeness of the algorithm.

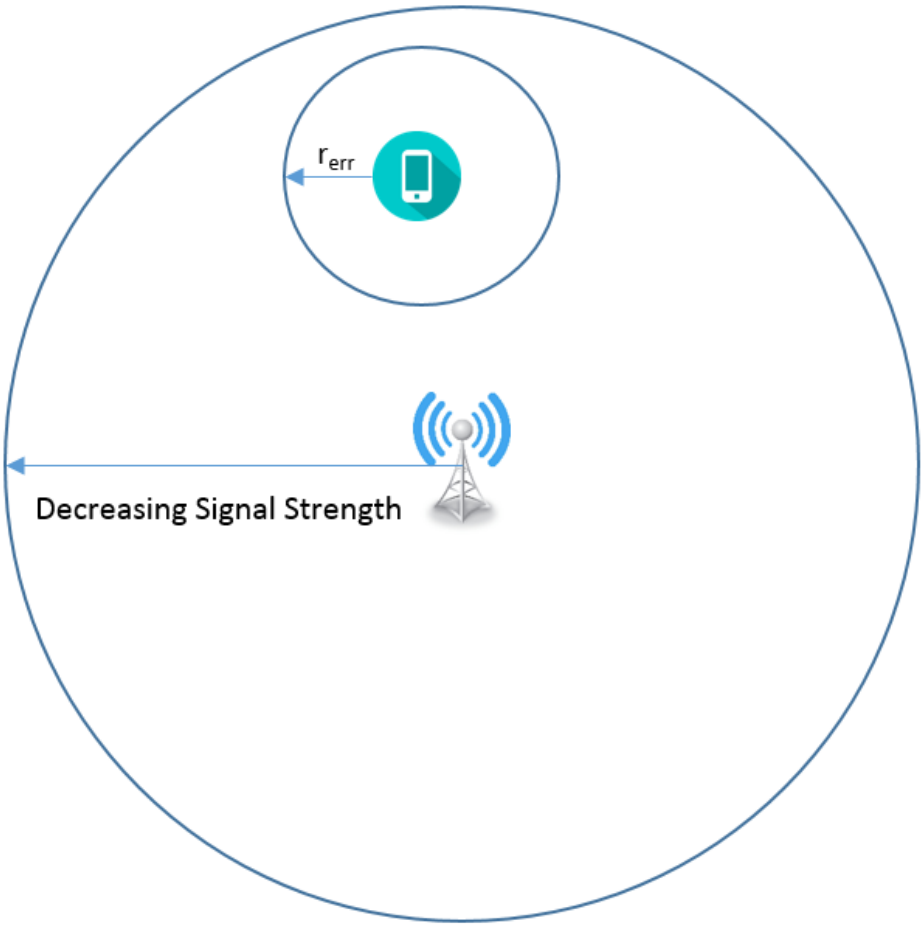


Figure 4.1: Uncertainty Radius

Eventually this process would produce p identical candidate lists if the user is not predicted to move out of range of all RATs. It is therefore reasonable to denote the final candidate list (which is one of the sub-list) of size m as the following:

$$Candidates = \{RAT_1, RAT_2, \ldots RAT_m\} \tag{4.2}$$

It should be noted that these candidates is able to form $2^m - 1$ potential solutions which must be evaluated by the comparison process.

Impact of p and r_{err} on network selection mentality

The candidate criterion described above suggest that a RAT is only qualified as a candidate if all predicted locations are still within the coverage of the same RAT, assuming that this RAT allows the allocation of the bandwidth requested. Such idea can be represented graphically in the following diagram:

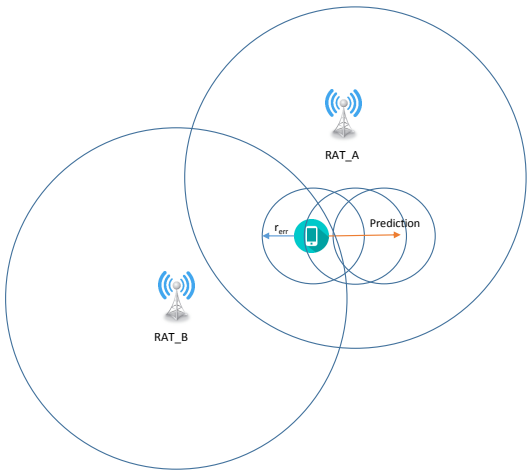


Figure 4.2: Impact of parameters on selection mentality

From Figure 4.2, it can be seen that RAT_A contains all the uncertainty circles whereas RAT_B does not, as a result, RAT_A is a qualified candidate but RAT_B is not. Increasing the prediction window size will result in an increase in the number of uncertainty circles if the sampling rate is kept constant whereas increasing the

uncertainty radius would lead to an increase in the area of each circle, both of which contribute to the total amount of area a RAT must cover in order to be considered as a candidate. It is therefore concluded that increase in any of these two parameters could result in a decrease in a number of candidates, thus making the system more selective or conservative. Such conservativeness is rewarded with stability as the system can guarantee that the user can at least maintain the current connections in the prediction window period if the predictions are accurate, and this could lead to a decrease in number of hand-overs.

4.2.2 Bandwidth Allocation Algorithm

According to Table I, the service is classified into different categories and some of them requires a certain amount of bandwidth to function. This section is to provide a detailed discussion on the bandwidth allocation strategy for each category of applications.

Assume that the following combination of RATs, which are a subset of the candidate list, are selected to provide the bandwidth of the application:

$$RATs_{select} = \{RAT_1, RAT_2, \dots RAT_n\} \tag{4.3}$$

Elastic Application

A greedy or user-centric approach is adopted for this type of application, which means the amount of bandwidth provided by each RAT is only limited by its physical layer capacity or remaining capacity. The amount of bandwidth provided by each RAT in $RATs_{select}$ can therefore be expressed as the following:

$$Bandwidth_i = \begin{cases} C_{single} & C_{remain} > C_{single} \\ C_{remain} & C_{remain} \leq C_{single} \end{cases} \tag{4.4}$$

Where:

- $Bandwidth_i$ refers to the amount of bandwidth provided by RAT_i
- C_{remain} refers to amount of bandwidth available in RAT_i
- C_{single} refers to the maximum amount of bandwidth that can be allocated to a single user by RAT_i

Hard-Real Time Application

The bandwidth allocation algorithm for real-time application focus on providing some form of load balancing to the overall network. More bandwidth will be requested from RAT with more remaining capacity and less from RAT with little remaining capacity. The amount of bandwidth requested from each RAT in $RATs_{select}$ is expressed as the following:

$$Bandwidth_i = \begin{cases} \frac{C_{remain_i}}{C_{totalRemain}} \times C_{requested} & C_{totalRemain} \geq C_{requested} \\ 0 & else \end{cases} \tag{4.5}$$

Where:

- C_{remain_i} denotes the amount of bandwidth available in RAT_i
- $C_{totalRemain}$ denotes the sum of available bandwidth of all RATs in $RATs_{select}$
- $C_{requested}$ denotes the amount of bandwidth requested by the application

Adaptive Application

In comparison with elastic application, a less greedy approach is adopted for this type of application. The rate multiplier M determines the maximum bandwidth the application can obtain, which can be described by the following equation:

$$B_{max} = B_{req} \times M \tag{4.6}$$

Where:

- B_{max} denotes the maximum bandwidth that the network is willing to provide
- B_{req} denotes the minimum bandwidth required to function
- M is the rate multiplier

The bandwidth that can be allocated by interfaces in $RATs_{select}$ is dependent on B_{max} and $C_{totalRemain}$, which is governed by the following equation:

$$C_{requested} = \begin{cases} B_{max} & C_{totalRemain} \geq B_{max} \\ C_{totalRemain} & B_{req} \leq C_{totalRemain} < B_{max} \\ 0 & else \end{cases}$$

At this point, the network is aware of the total amount of bandwidth that needs be allocated by all interfaces in $RATs_{select}$. The amount of bandwidth allocated by each RAT in $RATs_{select}$ is then governed by equation in 4.5

4.2.3 MADM Algorithm

Vector normalisation method in Table III and TOPSIS method are used to facilitate the comparison of results in the MADM process. Assume the following set of RATs, which is one of the potential solutions, is of interest:

$$S_{potential} = \{RAT_1, RAT_2, \dots RAT_i\} \tag{4.7}$$

Attributes

The attributes that are compared in the selection process are as follows:

- Total monetary cost of bandwidth provided by interfaces
- Power dissipation of mobile host
- Bandwidth utility experienced by the user

Monetary Cost The monetary cost is calculated by the following equation:

$$Cost_{total} = \sum_{1}^{i} Cost_i \times Bandwidth_i \tag{4.8}$$

Where:

- $Cost_i$ denotes the cost per bbu of RAT_i
- $Bandwidth_i$ denotes the bandwidth provide by RAT_i

Power Dissipation The power dissipation induced by network is calculated by the following equation:

$$Power_{network} = \sum_{1}^{i} Power_i \times Bandwidth_i \tag{4.9}$$

Where $Power_i$ denotes the power cost per bbu of RAT_i. The total power dissipation is also influenced by signal strength as described in Equation 3.4, as a result, the final power dissipation is expressed as the following:

$$Power_{total} = Power_{network} + (1 - p) \times P_e \tag{4.10}$$

Bandwidth Utility The equations which govern the bandwidth utility of different types of applications are described in Figure 3.4. The bandwidth utility of the network is therefore expressed as:

$$U_{network} = \sum_{1}^{i} U_i \tag{4.11}$$

Where U_i denotes the bandwidth utility provided by RAT_i using the formulas in Figure 3.4. The bandwidth utility is also modelled to be dependent on the signal strength in Equation 3.3, therefore the total bandwidth utility is expressed as the following:

$$U_{total} = U_{network} \times p \tag{4.12}$$

Comparison Process

The modified normalised weight vector (Using equation 2.20 and equation 3.5) is expressed as the following:

$$W = \{W_{Cost}, W_{Power}, W_{Bandwidth}\} \tag{4.13}$$

After calculating the quantity or score of each attribute, it is then possible to establish a normalised decision matrix in the following form:

$$D = \begin{matrix} \text{Cost} & \text{Power} & \text{Bandwidth} \end{matrix} \begin{pmatrix} C_1 & P_1 & U_1 \\ \dots & \dots & \dots \\ C_n & P_n & U_n \end{pmatrix}$$

Where the number of rows indicate the number of potential solutions. The weighted score matrix is therefore:

$$V = D \times W^T \tag{4.14}$$

If any one of the attribute of a particular solution has 0 as its score, then such solution is not deemed as a viable solution, as a result, it is removed from the matrix, preventing the specific combination to participate in the MADM process. This is to avoid"ranking abnormality" behaviour of TOPSIS approach. Once the score matrix is computed, the comparison follows the process in Section 2.3.1.

4.2.4 RAT Connection Algorithm

After finding the most favoured combination of interfaces using MADM, there are still a few complications as to whether or not a connection should be made based on the MADM results. Assume that the user is currently connected to the following interfaces:

$$RAT_{current} = \{RAT_1, RAT_2 \dots RAT_c\} \tag{4.15}$$

Assume that the following set of interfaces is the most favoured set in the MADM process:

$$RAT_{favour} = \{RAT_1, RAT_2, \dots RAT_d\} \tag{4.16}$$

Each time a different set of interfaces are connected, the system will attempt to maintain the connections just made for, if possible, at least the amount of time the prediction window represents, such time period is called dead time. This mechanism is in place to avoid unnecessary hand-overs caused by ping-pong effects. The overall connection algorithm can be represented in the following diagram:

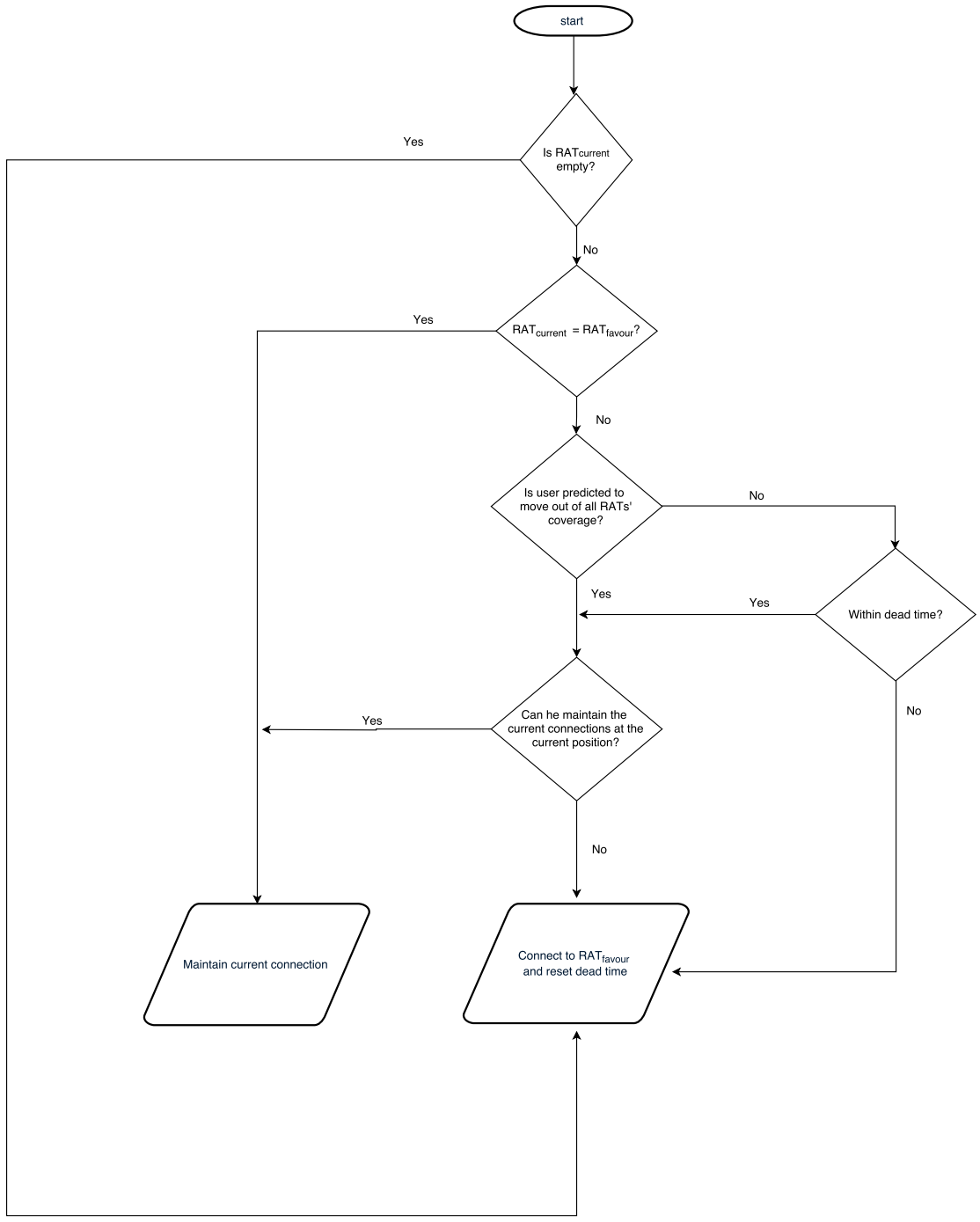


Figure 4.3: Connection Algorithm Overview

4.2.5 Overview

The interactions of different algorithms and components of the system can be described in the following diagram:

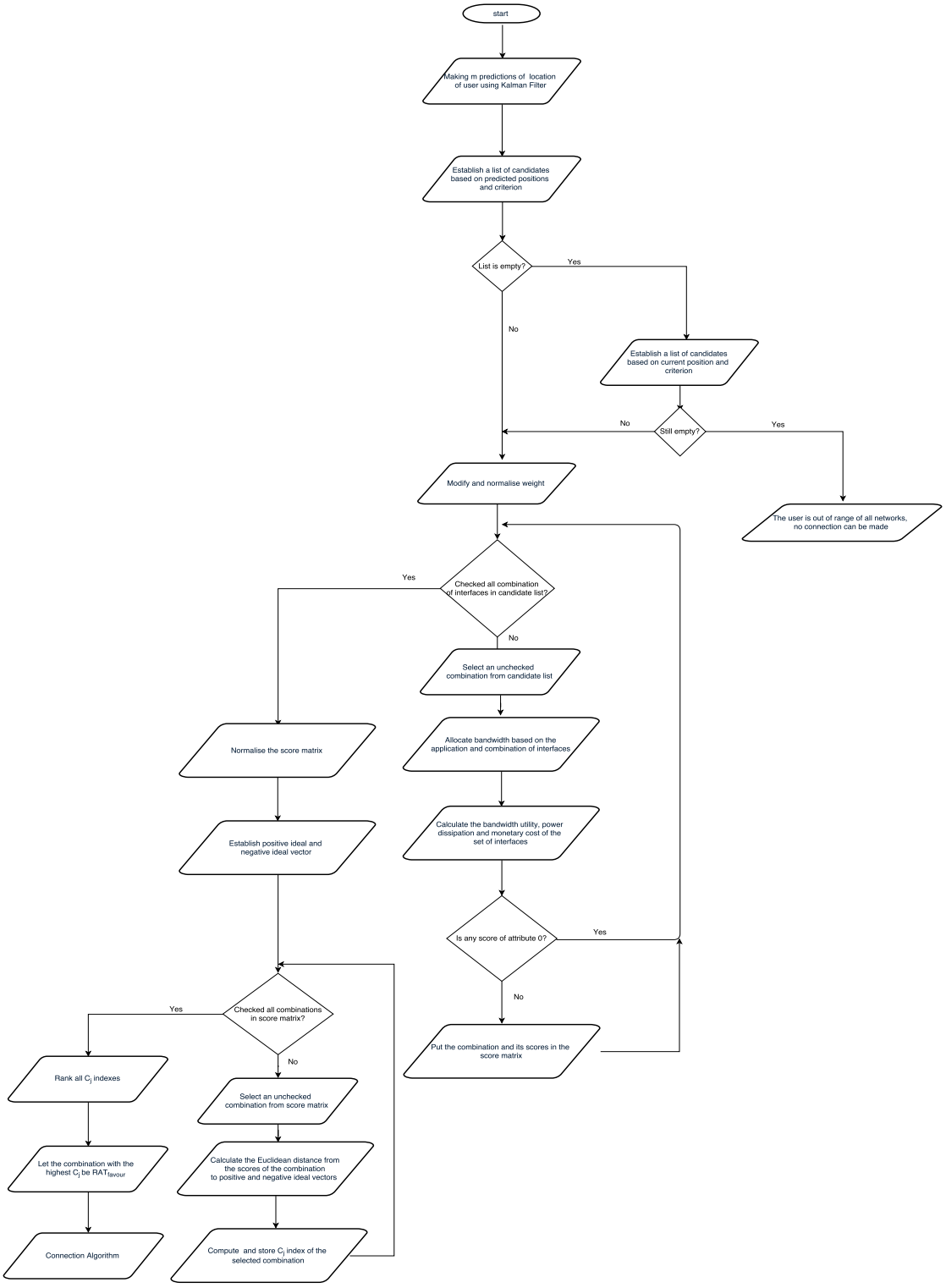


Figure 4.4: System Overview

Where:

- m is the size of the prediction window
- C_j is the index that indicates the relative performance of a set of interfaces
- RAT_{favour} is the most optimal set of interfaces according to MADM
- Connection Algorithm refers to Figure 4.3

4.2.6 Software Structure

A class diagram is included to illustrate the dependencies of the various classes in the simulation:

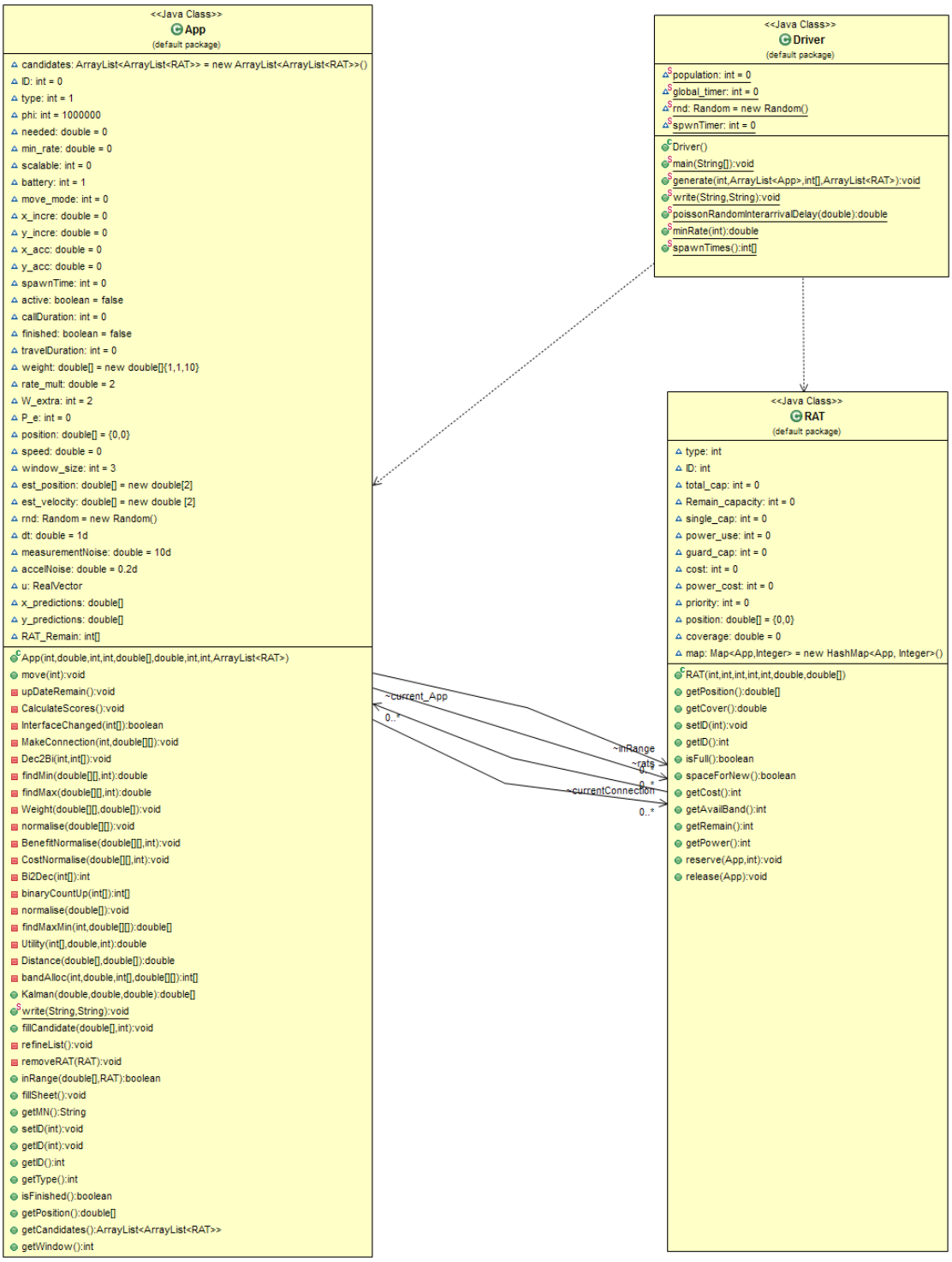


Figure 4.5: Class Diagram of the simulation

Chapter 5

Results & Discussion

The results are based on two simulations to validate the correctness of the implementation and highlight the benefits of implementing such algorithm.

5.1 Validation of Implementation

The scenarios are simulated with the design parameters (found in Section 4.1) defined as the following:

- Sampling period is 1 second
- Prediction window is 5
- r_{err} is 2
- Weight of $\{cost, power, bandwidth\}$ is $\{2, 3, 10\}$
- W_{extra} is 2
- P_e is 1
- M is 2

5.1.1 Experiment 1

This experiment is to evaluate the rationality of the network interface selection decision by investigating the behaviour of the selection algorithm when there is

only 1 elastic application in the network. The logic behind the decision making process will be explained in the following section. The relevant information of the RATs and user in the simulation are expressed in the following tables:

Table I: Relevant information of RATs

Name	ID	Power Cost per bbu	Network Type	Maximum bbu for a single user	Monetary Cost per bbu	Total Capacity	Position	Coverage radius
RAT_1	1	0.1	3G	2000	0.1	20000	$\{0,0\}$	100
RAT_2	2	0.01	4G	1000000	0.01	10000000	$\{50,50\}$	100
RAT_3	3	0.001	5G	10000000	0.001	100000000	$\{50,50\}$	100

Table II: Relevant information of user

Application type	Minimum Bandwidth	Battery	Position	Initial Speed	Movement Mode	Duration
Elastic	N/A	100	$\{60,60\}$	10	Turning Movement	10

From the parameters defined above, the system is expected take on a somewhat conservative approach when making connections to the interfaces due to the low sampling rate and relatively big prediction window size. The network selection algorithm should select a set of interfaces that provide a lot of bandwidth without imposing a significant amount of monetary cost and power dissipation. The simulation results are collected in the following table:

current Position		predictions_x					predcitions_y					current Interface ID	Bandwidth Allocated
62.84533	53.60437	65.99731	70.36809	73.95528	77.60535	79.75203	47.44975	40.09121	33.83767	29.90243	26.93147	3	10000000
65.69065	47.20873	68.75067	71.35952	71.66172	73.1152	77.894	41.19112	35.95375	29.85591	22.29707	14.14293	3	10000000
68.53598	40.8131	71.5019	73.38005	77.32821	81.10224	79.66288	34.59038	27.35473	20.77803	15.07867	11.06832	3	10000000
71.3813	34.41747	74.38991	77.11012	80.34003	84.89149	91.7638	27.83315	21.79289	14.61373	9.19249	4.691697	3	10000000
74.22663	28.02183	77.09934	79.47547	83.04867	85.32099	88.0557	21.97115	15.94849	10.09422	6.747128	0.597689	3	10000000
77.07195	21.6262	80.42037	82.91568	86.12094	88.56688	93.78718	15.20311	9.78687	3.400751	-2.35821	-10.3341	3	10000000
78.87952	28.3888	80.10167	81.16021	83.07616	81.26275	83.30886	35.17785	42.32397	48.52174	57.91633	65.49408	3	10000000
80.68709	35.15139	82.66073	85.2058	87.04452	85.64383	86.30824	42.53161	50.54153	57.82602	62.97482	68.32573	3	10000000
82.49466	41.91399	83.98328	85.49483	87.08378	89.14168	91.29364	48.69202	54.49549	59.40109	63.68018	71.81601	3	10000000
84.30223	48.67658	85.7176	87.49669	90.86998	91.612	94.19499	56.0377	61.78049	69.04023	75.53192	84.21938	3	10000000

Figure 5.1: Simulation results of experiment 1

It should be noted that some values in the above table are rounded off to avoid clumsiness. A graphical representation is included to enhance the visualisation and understanding of the results.

It is obvious to read from the Figure 5.2 that the user moved in two directions during his call. When the user just started moving, his current and predicted positions were in coverage of all 3 RATs but the algorithm decided to connect

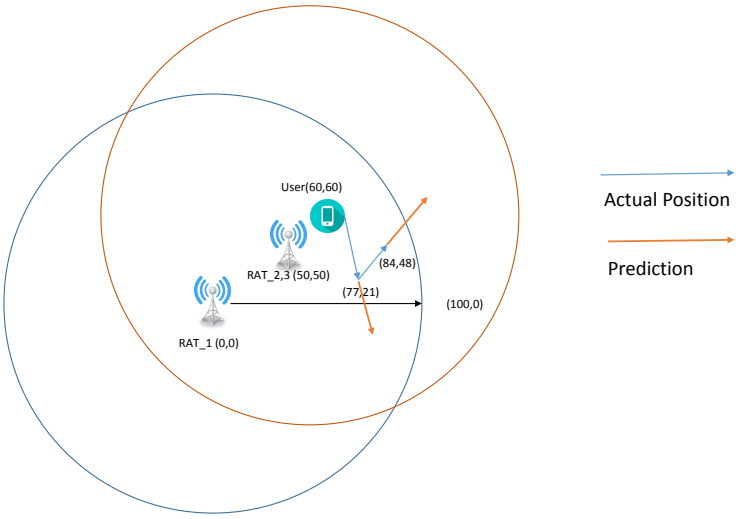


Figure 5.2: Graphical representation of results of experiment 1

to RAT_3 only because the extra cost which is induced by connecting to another interface outweighed the benefits it could bring. This can be observed in the following normalised weighted score matrix in Table III:

Table III: Decision Matrix at position {62,53}

Combination	Bandwidth Allocation	Monetary cost Score	Power cost score	Bandwidth utility score
$\{RAT_3\}$	{10000000}	0.003759	0.00907	0.4351
$\{RAT_2\}$	{1000000}	0.003759	0.00907	0.252
$\{RAT_2, RAT_3\}$	{1000000,10000000}	0.00187	0.004536	0.435226
$\{RAT_1\}$	{2000}	0.00187	0.281	0.00435
$\{RAT_1, RAT_3\}$	{2000,10000000}	0.00368	0.008789	0.43518
$\{RAT_1, RAT_2\}$	{2000,1000000}	0.00368	0.008789	0.25234
$\{RAT_1, RAT_2, RAT_3\}$	{2000,1000000,10000000}	0.00186	0.00446	0.4352

From the table, it is obvious that all combinations have similar scores in the monetary cost category while interfaces that provide more bandwidth having a slightly lower score. It is further observed that the combination $\{RAT_1\}$ has a relatively high score in the power cost category due to the lack of bandwidth

it is providing, and this is evident in its bandwidth utility score. Finally, sets of interfaces that allocate a generous amount of bandwidth score high in the bandwidth utility category and due to the heavy weight on bandwidth utility, the scores are usually much higher in comparison with other categories. It should be noted that bandwidth allocated by RAT_1 and RAT_2 has a very limited impact on the bandwidth utility score and this is due to the fact that they only provide 0.002% and 10% of the bandwidth provided by RAT_3 respectively. The computation of C_j index is as follows:

Table IV: C_j index of different combinations at {62,53}

Combination	C_j index
$\{RAT_3\}$	0.567076349
$\{RAT_2\}$	0.5
$\{RAT_2, RAT_3\}$	0.563515504
$\{RAT_1\}$	0.436536141
$\{RAT_1, RAT_3\}$	0.56686864
$\{RAT_1, RAT_2\}$	0.397094371
$\{RAT_1, RAT_2, RAT_3\}$	0.563463859

From Table IV, combination $\{RAT_3\}$ has a slightly bigger C_j index in comparison with other combinations that include RAT_3, which proved that the extra cost of including RAT_1 and RAT_2 outweighed benefits. It should be noted that the C_j index would be more distinguishable when the difference of cost parameters or QoS amongst different RATs become bigger. At this point we can conclude that this decision is a plausible one as it provides a very good amount of bandwidth utility without inducing too much extra power dissipation and monetary cost.

The user then changed his direction at position {77,21} and he was predicted to move out of the coverage of RAT_1. The system was therefore expected not to consider RAT_1 as one of the candidate even though the user was still within the coverage of RAT_1 at that time. The decision matrix at {82,42} is shown as the following Table V:

Table V: Decision Matrix at position {82,42}

Combination	Bandwidth Allocation	Monetary cost Score	Power cost score	Bandwidth utility score
$\{RAT_3\}$	{10000000}	0.125429	0.18814	0.62884
$\{RAT_2\}$	{1000000,0}	0.125429	0.18814	0.30588796
$\{RAT_2, RAT_3\}$	{1000000,10000000}	0.06271	0.09407	0.6292398

It is observed that RAT_1 was no longer considered in the decision making process, as a result, the size of the decision matrix was reduced. It is quite obvious that combination $\{RAT_3\}$ scored the highest overall, and this is confirmed by the following C_j indexes:

Table VI: C_j index of different combinations at position {82,42}

Combination	C_j index
$\{RAT_3\}$	0.998846201
$\{RAT_2\}$	0.25906817
$\{RAT_2, RAT_3\}$	0.74093183

In Table VI, it can be observed that combination $\{RAT_3\}$ was the clear favourite in comparison with other candidates. The decision is also a reasonable one as it provides a lot of bandwidth without incurring too much cost in other categories. At this point, it is claimed that the criterion selection and MADM components of the algorithm work as intended.

5.1.2 Experiment 2

This experiment focuses on the behaviour of the selection system when a user is predicted to move out of the coverage of all $RATs$ and when the user enters a high quality network for a short amount of time, finally the benefits of prediction mechanism will be highlighted. The system is expected to maintain the current connection if possible, and make network connection decisions based on current position only when it is not possible to maintain the current network connections.

The parameters of the user and network are modified into the following:

Table VII: Relevant information of RATs

Name	ID	Power Cost per bbu	Network Type	Maximum bbu for a single user	Monetary Cost per bbu	Total Capacity	Position	Coverage radius
RAT_1	1	0.1	4G	1000000	0.1	10000000	$\{0,0\}$	50
RAT_2	2	0.01	4G	1000000	0.01	10000000	$\{0,25\}$	15
RAT_3	3	0.001	5G	10000000	0.001	100000000	$\{0,0\}$	10

Table VIII: Relevant information of user

Application type	Minimum Bandwidth	Battery	Position	Initial Speed	Movement Mode	Direction	Duration
Elastic	N/A	100	$\{0,0\}$	3	Controlled Movement	{0,1}	20

The results obtained from the simulations are as follows:

current Position		predictions_x					predcitions_y					current Interface		Bandwidth Allocated	
0	3	-0.0874	-0.1974	-0.1753	-1.202	-0.5299	5.78316	8.58751	12.1765	15.0959	14.5611	1		1000000	
0	6	0.66329	0.04286	0.2375	0.3571	-0.0039	9.33239	11.2278	13.0809	19.1618	24.5782	1		1000000	
0	9	0.52425	-0.3564	-2.3212	-0.7481	1.17632	12.4052	14.9471	18.3685	19.638	25.2617	1		1000000	
0	12	-0.2601	-0.449	0.19226	-2.1147	0.99228	15.4229	19.162	22.8135	26.9321	29.4421	1		1000000	
0	15	-0.4598	-1.4011	-4.3524	-3.8163	-1.3521	17.8028	20.541	20.7644	24.2722	26.7818	1		1000000	
0	18	-0.1138	0.07218	-1.7782	-1.5105	-3.4644	20.3566	22.7439	26.4522	28.5417	31.2346	1	2	1000000	1000000
0	21	-0.016	1.4134	1.17186	1.20958	2.90552	23.7189	26.9835	30.879	34.8538	37.6961	1	2	1000000	1000000
0	24	0.34285	0.54359	2.40709	-1.7299	-1.6586	26.6224	29.8876	34.3639	35.991	34.2281	1	2	1000000	1000000
0	27	-0.1177	-0.0181	-0.0011	-0.2499	1.24114	30.5766	32.3294	35.5588	38.1231	41.6587	1	2	1000000	1000000
0	30	0.59568	1.69314	3.49127	3.77712	0.59059	33.1073	35.533	39.4358	42.2556	47.7223	1	2	1000000	1000000
0	33	0.09149	-0.2549	-2.3499	-2.3914	-0.4976	36.1769	39.2998	41.5396	45.9878	50.915	1	2	1000000	1000000
0	36	-0.1716	-0.4975	0.93062	-0.9944	1.21621	38.332	41.2499	42.9747	46.6012	48.4638	1	2	1000000	1000000
0	39	-0.6515	-0.4902	-0.4076	0.14352	0.21417	42.2592	46.9919	50.9142	55.7674	59.9336	1		1000000	
0	42	0.15374	-0.4851	-0.0278	-2.3928	-3.1731	44.6184	45.195	46.1028	48.5719	49.3994	1		1000000	
0	45	-0.3412	-1.076	-4.1111	-4.235	-3.0075	47.8343	49.9697	52.9761	53.3251	54.2476	1		1000000	
0	48	0.12056	-0.2068	-0.4427	-2.0913	-2.5005	50.1827	51.8128	55.977	61.7567	61.7618	1		1000000	
0	51	-0.1639	-0.5217	0.15515	-2.5614	-2.1372	54.3981	57.9821	61.8198	62.4886	64.1956				
0	54	0.21899	0.03644	1.00663	2.21741	1.37385	56.3732	59.1494	64.092	66.6479	68.8936				
0	57	-0.8187	-1.697	-3.4591	-5.1841	-4.4619	58.9978	62.8884	66.5676	70.4539	74.1025				
0	60	0.12753	0.53458	0.06192	0.2179	1.30866	63.2953	65.6993	68.5853	70.6585	71.3673				

Figure 5.3: Simulation results of experiment 2

The results are shown graphically in the following diagram:

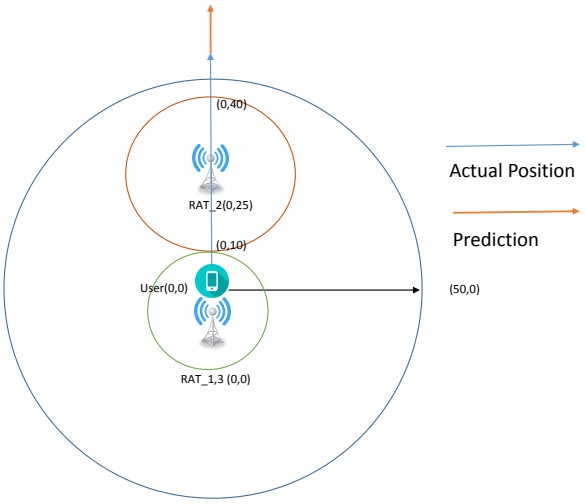


Figure 5.4: Diagram of simulation results of experiment 2

In order to gain a solid understanding of the above results, the discussion is divided into different sections, each of which represents a stage of the user's positions and network states in this particular scenario.

Stage 1

During this stage, the user just started moving away from its original position, and he was in coverage of RAT_1 and RAT_3 at that time while being predicted to move out of coverage of RAT_3. Position {0,3} is an example of this stage. The system made the decision of connecting to RAT_1 only despite the fact that RAT_3 is a network of much higher quality, and this is advantageous because it could avoid frequent vertical hand-overs as the user moves in and out of a high quality network which has a small coverage.

Stage 2

At this stage, the user has just made connection to RAT_1, and he was in coverage of RAT_1 and RAT_2 at that time while being predicted to stay in coverage of these two RATs during the prediction window period. The user maintained his current

connection despite there might be better options and this is due to the dead time effect, an example of this stage would be position {0,15}. This could avoid some of the unnecessary hand-overs or network commitment when the user is making irregular movements or when the predictions are not very reliable. The C_j indexes at {0,15} are as follows:

Table IX: C_j index at position {0,15}

Combination	C_j index
$\{RAT_2\}$	0.194586357
$\{RAT_1\}$	0.64162697
$\{RAT_1, RAT_2\}$	0.745642107

It is observed that the best set of interfaces to be connected to at position {0,15} was actually $\{RAT_1,RAT_2\}$, and this proved that the dead time mechanism was effective against the ping-pong effect when the location predictions were accurate enough.

Stage 3

During this stage, the user experienced the same situation as stage 2 except that the dead time has passed, as a result, the system decided to connect to RAT_2 as well as RAT_1, an example of such stage is position {0,18}.

Stage 4

During this stage, the user was predicted to move out of coverage of all RATs, however it was possible to maintain its connections for the moment, as a result, no changes were made to the connection. An example of this stage is position{0,33}.

Stage 5

During this stage, the user experienced the same situation as stage 4 except that it was no longer possible to maintain the same connections, as a result, the system made a network selection decision without taking predictions into consideration. An example of such stage is position {0,42}.

Stage 6

At this stage, the user was out of coverage of all networks, as a result, no connections could be made. An example of such stage is position {0,60}.

After evaluating decisions made in different stages, it is then claimed that the implementation has matched the algorithm proposed.

5.2 Benefits of Prediction Mechanism

Based on the observation in the section above, it is evident that the benefits of implementing the prediction mechanism are as follows:

- The prediction mechanism enhances the effectiveness of the dead time mechanism when the predictions are reasonably accurate, which greatly reduces the number of hand-overs caused by ping-pong effect.
- The algorithm is able to ensure ubiquitous connection within the prediction window if the positions are predicted accurately.
- The prediction mechanism enhances the rationality of the decision making process, e.g. the algorithm is able to ignore a high quality network that has a relatively small coverage.

Chapter 6

Conclusion

This paper has proposed a location prediction based network selection algorithm for multi-homed mobile terminals within a heterogeneous wireless network environment. The algorithm proposed was implemented and examined in the simulation which is based on the network and movement model established in the study.

Based on the results and discussion above, it is concluded that the algorithm is able to perform a logical selection of set of interfaces based on current and predicted positions of the user when the weight of each attribute and relevant parameters are given. It is also concluded that the algorithm is able to battle the ping-pong effect caused by moving in and out of coverage of a network in quick succession when the future locations of the user are predicted in a reasonably accurate manner. Furthermore it is concluded that the algorithm is able to ensure ubiquitous connection within a prediction window if the predictions are reasonably accurate. Finally it is concluded that all objectives of this study has been met and the source code of the simulation can be found in Appendix A.

Chapter 7

Future Work

In order to improve the simulation model and the robustness of the algorithm, the following is recommended to be included as future work:

7.1 Road Network System

In order to improve the accuracy of predictions, a road network system can be incorporated into the current design. The road network system will greatly improve the accuracy of prediction in the scenario where the user is required to turn into a certain direction, which in turn improves the reasonableness of the decision making process.

7.2 Dynamic Adjustment of Bandwidth Allocated to Elastic Services

The current elastic service allocation strategy could potentially be too greedy which could block the request of other type of services. The resource allocation would be more fair if the network can keep track of the number of service requests at different period of the day, and allocate less bandwidth to elastic services during peak hour and vice versa.

7.3 Re-scaling Bandwidth Allocated to Elastic and Adaptive Services

By re-scaling the number fo bandwidth supplied to adaptive and elastic services, the network system will be able accommodate more users, however this could lead to user dissatisfaction.

7.4 Including More Attributes in Comparison

By including more attributes in comparison, the algorithm will be able to make more informed and well-rounded decision.

7.5 Including Predictions in Comparison

The network will be able to find the overall optimal solution in a prediction window, if it is able to include the scores of predicted positions in the comparison process.

References

[1] V. Troeger, "The simple linear regression model," tech. rep., The university of Warwick, 2012.

[2] G. Welch and G. Bishop, "An introduction to the kalman filter," tech. rep., University of North Carolina at Chapel Hill, 2004.

[3] J. Ylitalo, T. Jokikyyny, T. Kauppinen, A. J. Tuominen, and J. Laine, "Dynamic network interface selection in multihomed mobile hosts," in *System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference on*, pp. 10 pp.–, Jan 2003.

[4] P. N. Tran and N. Boukhatem, "An utility-based interface selection scheme for multi-homed mobile terminals," in *2009 IEEE 20th International Symposium on Personal, Indoor and Mobile Radio Communications*.

[5] P. Sharma, "Evolution of mobile wireless communication networks-1g to 5g as well as future prospective of next generation communication network," *A Monthly Journal of Computer Science and Information Technology*, 2013.

[6] A. L. N. M. German Castignani, Alberto Blanc, "Urban 802.11 community networks for mobile users: Current deployments and prospectives," *Mobile Networks and Applications*, 2012.

[7] *Joint Call Admission Control Algorithm for Fair Radio Resource Allocation in Heterogeneous Wireless Networks Supporting Heterogeneous Mobile Terminals*, 2010.

[8] A. V. B. Mudit Ratana Bhalla, "Generations of mobile wireless technology: A survey," *International Journal of Computer Applications*, 2010.

[9] T. Mshvidobadze, "Evolution mobile wireless communication and lte networks," in *Application of Information and Communication Technologies (AICT), 2012 6th International Conference on*.

[10] J. G. Andrews, "What will 5g be?," *IEEE Journal on Selected Areas in Communications (Volume:32 , Issue: 6)*, 2014.

[11] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[12] M. d. L. V. V. G. C. V. A. M. H. Angel Juan Snchez Garca1, Homero Vladimir Ros Figueroa2, ed., *Motion Prediction of Regions Through the Statistical Temporal Analysis Using an AutoRegressive Moving Average (ARMA) Model*.

[13] C. Dismuke and R. Lindrooth, "Ordinary least squares," *Methods and Designs for Outcomes Research*, vol. 93, pp. 93–104, 2006.

[14] C. Pandey and S. Nemade, "Enhancement of the speech quality by the implementation of second order fast adaptive kalman filter algorithm," in *2014 Annual IEEE India Conference (INDICON)*, pp. 1–6, Dec 2014.

[15] T. Basar, *A New Approach to Linear Filtering and Prediction Problems*, pp. 167–179. Wiley-IEEE Press, 2001.

[16] N. B. Phuoc Nguyen TRAN, "Comparison of madm decision algorithms for interface selection in heterogeneous wireless networks,"

[17] D. M. PAVLICIC, "Normalisation affects the results of madm methods," *Yugoslav Journal of Operations Research*, 2001.

[18] H. C. Olabisi E. Falowo, "Rat selection for multiple calls in heterogeneous wireless networks using modified topsis group decision making technique," in *2011 IEEE 22nd International Symposium on Personal, Indoor and Mobile Radio Communications*.

[19] S. Shenker, "Fundamental design issues for the future internet," *IEEE Journal on Selected Areas in Communications*, vol. 13, pp. 1176–1188, Sept 1995.

[20] A. Commons, "17.2 kalman filter." `http://commons.apache.org/proper/commons-math/userguide/filter.html`. Accessed:12/09/2016.

Appendix A

Simulation software

Driver.JAVA:

```
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Random;

public class Driver {


static int population = 50; // Number of people in system
static int global_timer = 0;

static Random rnd = new Random();
static int spwnTimer = 0;

/*
* Type of Traffic
* 1 - Elastic, Breq = limitless
* 2 - Adaptive, Breq = 150 kbps
* 3 - Adaptive, Breq = 2000 kbps
* 4 - Hard-Real time, Breq = 64 kbps
* 5 - Hard-Real time, Breq = 20 kbps
*/


/*
* 1 bbu = 1kbps
*       case 1:
single_cap = 2000;
case 2:
single_cap = 1000000;
case 3:
single_cap = 10000000;
case 4:
single_cap = 54000; WIFI = 40m
*
*
*
*/

public static void main(String[] args) {

        ArrayList<App> ActiveApps = new ArrayList<App>();
        ArrayList<RAT> rats = new ArrayList<RAT>();
```

```
        int[] timers = spawnTimes();



        RAT RAT1 = new RAT
        (0.01,2,0.01,1000000*10,1000000*6,100,new double[]{0,0});
        //(int power, int type, int cost, int capacity, int guard,double
        //coverage,double[] position)
        RAT1.setID(1);

        RAT RAT2 = new RAT
        (0.01,2,0.01,1000000*5,1000000*3,100,new double[]{0,0});
        RAT2.setID(2);

        RAT RAT3 = new RAT
        (0.01,2,0.01,1000000*8,1000000*4,100,new double[]{0,0});
        RAT3.setID(3); //Creating RATs
        rats.add(RAT1);rats.add(RAT2);rats.add(RAT3);


        for (int i =0;i<population;i++)
        {
                generate(i,ActiveApps,timers,rats); //Create apps
        }

        while (true)
        {
                for (int i =0;i<ActiveApps.size();i++)
                {
                        ActiveApps.get(i).move(global_timer,null);
                        if (ActiveApps.get(i).isFinished())
                        {ActiveApps.remove(i);}
                }
                global_timer ++;
                try {
                Thread.sleep(1000);
                } catch (InterruptedException ie) {
                //Handle exception
                ie.printStackTrace();
                }

                if (ActiveApps.size() ==0)
                {System.out.println("FINISHED"); break;}
                }


        }

public static void generate(int ID,ArrayList<App> active,
        int[] timers, ArrayList<RAT>rats){ //Generate apps
                int type =(int) (Math.round(4*rnd.nextDouble()+1));
                double[] position = {rnd.nextDouble()*5,
                rnd.nextDouble()*5};
                double minrate = minRate(type);
                int battary = (int)(Math.round(99*rnd.nextDouble()+1));
                double speed = 9*rnd.nextDouble()+1;
                int duration = (int) poissonRandomInterarrivalDelay(1/2.0)
                +1;
                active.add(new App(type,minrate,timers[ID],
                battary,position,speed,1,duration,rats));
                active.get(ID).setID(ID);
        }



public static void write(String name, String input){
                try
                {
                        String filename= name;
```

```
                        FileWriter fw = new FileWriter(filename,true);
                        fw.write(input);
                        fw.close();
                }
                catch(IOException ioe)
                {
                        System.err.println("IOException: " +
                        ioe.getMessage());
                }
        }

public static double poissonRandomInterarrivalDelay(double L) {
                // L = 1/Lambda
                double temp = Math.log(1.0-Math.random())/(-L);
                return temp;
        }

public static double minRate(int type){
                switch (type)
                {
                        case 1:
                                return -1; //infinite
                        case 2:
                                return 150;
                        case 3:
                                return 2000;
                        case 4:
                                return 64;
                        case 5:
                                return 20;
                        default:
                                return -1;
                }
        }

public static int[] spawnTimes(){ //Spawn times of the apps
                int[] times = new int[population];
                for (int i =0;i<population; i++)
                {
                        spwnTimer += Math.round
                        (poissonRandomInterarrivalDelay(1/2.0));
                        times[i] = (int) (spwnTimer);
                }
                return times;
        }
}
```

App.JAVA:

```
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Random;

import org.apache.commons.math3.filter.DefaultMeasurementModel;
import org.apache.commons.math3.filter.DefaultProcessModel;
import org.apache.commons.math3.filter.KalmanFilter;
import org.apache.commons.math3.filter.MeasurementModel;
import org.apache.commons.math3.filter.ProcessModel;
import org.apache.commons.math3.linear.Array2DRowRealMatrix;
import org.apache.commons.math3.linear.ArrayRealVector;
import org.apache.commons.math3.linear.RealMatrix;
import org.apache.commons.math3.linear.RealVector;
import org.apache.commons.math3.random.JDKRandomGenerator;
import org.apache.commons.math3.random.RandomGenerator;

public class App {
```

```
ArrayList<RAT> rats = new ArrayList<RAT>();
ArrayList<ArrayList<RAT>> candidates = new ArrayList<ArrayList<RAT>>();
ArrayList<ArrayList<RAT>> originalCandidates =
new ArrayList<ArrayList<RAT>>();
ArrayList<RAT> inRange = new ArrayList<RAT>();
ArrayList<RAT> currentConnection = new ArrayList<RAT>();
int ID = 0;
int type = 1;
int phi =1000000; //parameter
/*
* Type of Traffic
* 1 - Elastic, Breq = limitless
* 2 - Adaptive, Breq = 150 kbps
* 3 - Adaptive, Breq = 2000 kbps
* 4 - Hard-Real time, Breq = 64 kbps
* 5 - Hard-Real time, Breq = 20 kbps
*/
double needed = 0; //Total Resources needed for non-real traffic
double min_rate = 0; //Minimum rate for real-time traffic
int scalable = 0; // 1 for scalable traffic, 0 for not
int battery = 1; // 1 to 100% battery left
int move_mode = 0;
double x_incre = 0; //Velocity
double y_incre =0;
double x_acc = 0;
double y_acc= 0;
int spawnTime =0;
boolean active = false;
int callDuration =0;
boolean finished = false;
int travelDuration = 0; //Default travel duration
double[] weight = new double[]{2,3,10}; //COST POWER BANDWIDTH
double rate_mult = 2;
int W_extra = 2;
int P_e = 1;
int deadTime = 0;
/*
* 0 ---- random
* 1 ---- Turning Movement
*                    ---------
*                             \
*                              |
*                              |
*
*/
double[] position = {0,0}; //Position of the user;
double speed = 0;
int window_size =5;
double[] est_position = new double[2];
double[] est_velocity = new double [2];
Random rnd = new Random();
// discrete time interval in seconds
double dt = 1d;
// position measurement noise (meter)
double measurementNoise = 10d;
// acceleration noise (meter/sec^2)
double accelNoise = 0.2d;
RealVector u;
double [] x_predictions;
double [] y_predictions;
int[] RAT_Remain;
boolean outofrange = false;

public App(int type, double minrate, int spawn, int battery,double[] position
, double speed, int movemode,int duration,ArrayList<RAT>rats){
        this.type = type;
        this.min_rate = minrate;
        this.battery = battery;
```

```
        this.speed = speed;
        this.position = position;
        this.est_position[0] = position[0];
        this.est_position[1] = position[1];
        this.move_mode = movemode;
        this.est_velocity[0] = 0;
        this.est_velocity[1] = 0;
        this.spawnTime = spawn;
        this.callDuration= duration;
        this.rats = rats;
        weight[1] = weight[1] + W_extra*(1-(double)battery/100);
        for (int i =0;i<window_size;i++)
        {
                candidates.add(new ArrayList<RAT>());
                originalCandidates.add(new ArrayList<RAT>());
        }
        writeHeading();


}

private void writeHeading() {
        // TODO Auto-generated method stub
        String builder = "";
        builder += "current Position,"+""+ ",";
        builder += "predictions_x,";
        for (int i =0;i<window_size-1;i++)
        {
        builder += ""+",";
        }
        builder += "predcitions_y,";
        for (int i =0;i<window_size-1;i++)
        {
        builder += ""+",";
        }
        builder +="current Interface,";

        for (int i =0;i<2;i++)
        {
        builder += ""+",";
        }
        builder += "Bandwidth Allocated";
        write(Integer.toString(a)+".csv",builder+"\n");
}

public void move(int global_timer, double[] ControlledDirection)
{
        double magnitude = 0;

        if (global_timer >= spawnTime && global_timer<
        (spawnTime + callDuration))
        //Check if the user is still active
        {active = true;}
        else
        {active = false;}

        if (active == true)
        {
        switch(move_mode)
        {
                case 0:
                x_incre = 2*rnd.nextDouble()-1;
                 //x_incre is more like x_direction
                y_incre = 2*rnd.nextDouble()-1;
                magnitude =  Math.sqrt(x_incre*x_incre+y_incre*y_incre);
                x_incre = 1/magnitude *x_incre;
                y_incre = 1/magnitude *y_incre;

                position[0] += x_incre*speed*dt;
```

```
position[1] += y_incre*speed*dt;


x_predictions = Kalman(position[0],x_incre*speed, x_acc);
y_predictions = Kalman(position[1],y_incre*speed,y_acc);

fillSheet();
break;

case 1:
if (travelDuration ==0)
        //If the user needs to change direction,
        //generate a new direction
{
        x_incre = 2*rnd.nextDouble()-1;
        //x_incre is more like x_direction
        y_incre = 2*rnd.nextDouble()-1;
        magnitude =  Math.sqrt
        (x_incre*x_incre+y_incre*y_incre);
        x_incre = 1/magnitude *x_incre;
        y_incre = 1/magnitude *y_incre;
        travelDuration = (int)(5*(rnd.nextDouble()*5));
        position[0] += x_incre*speed*dt;
        position[1] += y_incre*speed*dt;
        x_predictions = Kalman
        (position[0],x_incre*speed, x_acc);
        y_predictions = Kalman
        (position[1],y_incre*speed,y_acc);
        travelDuration --;
        fillSheet();
}
else
{
        position[0] += x_incre*speed*dt;
        position[1] += y_incre*speed*dt;
        x_predictions = Kalman
        (position[0],x_incre*speed, x_acc);
        y_predictions = Kalman
        (position[1],y_incre*speed,y_acc);
        travelDuration--;
        fillSheet();
}
break;

case 2: //controlled movement
position[0] += ControlledDirection[0]*speed*dt;
position[1] += ControlledDirection[1]*speed*dt;
x_predictions = Kalman
(position[0], ControlledDirection[0]*speed, x_acc);
y_predictions = Kalman
(position[1],ControlledDirection[1]*speed,y_acc);
fillSheet();
break;
}
deadTime--;
for (int i =0;i<window_size;i++)
{
        fillCandidate
        (new double[]{x_predictions[i],y_predictions[i]},
        i);
}

refineList();

if (candidates.get(0).size() == 0)
//If the user is predicted to move out of coverage of
//all RATs, then use the current position
{
        outofrange =true;
```

```
                        for (int i =0;i<rats.size();i++)
                        {
                                if (inRange(position,rats.get(i)))
                                {
                                        candidates.get(0).add(rats.get(i));
                                }
                        }

                }
                RAT_Remain = new int[candidates.get(0).size()];
                upDateRemain();

                CalculateScores();


                //CLEAR LIST
                for (int i =0;i<window_size;i++)
                {
                        candidates.get(i).clear();
                        originalCandidates.get(i).clear();
                }
                outofrange = false;
        }

        if (global_timer>=(spawnTime + callDuration))
        {finished = true;}

        if (finished ==true)//If the user is inactive, release resources
        {
                for (int i =0;i<currentConnection.size();i++)
                {currentConnection.get(i).release(this);}
        }

}

private void upDateRemain() {
// TODO Auto-generated method stub
        for (int i =0;i<RAT_Remain.length;i++)
        {
                RAT_Remain[i] = candidates.get(0).get(i).getRemain();
        }
}

private void CalculateScores() {
// TODO Auto-generated method stub
//Atrributes: POWER, COST, BANDWIDTH
        normalise(weight);
        double[][] scoreMatrix = new double
        [(int) (Math.pow(2, candidates.get(0).size())-1)][3];
        // Score Matrix
        double max_C = 0;
        int RAT_Chosen = 0;
        double[] MaxMin = new double[6];

        MaxMin = findMaxMin(this.type,scoreMatrix);
        // [COST POWER BANDUTILITY]

        normalise(scoreMatrix);
        Weight(scoreMatrix,weight);
        double[] IdealPos = new double[3];
        //Establishing ideal positive and negatives
        double[] IdealNeg = new double[3];

        IdealPos[0] = findMax(scoreMatrix,0);
        IdealPos[1] = findMax(scoreMatrix,1);
        IdealPos[2] =findMax(scoreMatrix,2);
        IdealNeg[0] = findMin(scoreMatrix,0);
        IdealNeg[1] = findMin(scoreMatrix,1);
        IdealNeg[2] = findMin(scoreMatrix,2);
```

```
        for (int i =0;i<scoreMatrix.length;i++)
        {
                double S_plus = Distance(scoreMatrix[i], IdealPos);
                //Calculate Euclean distance to ideals
                double S_Minus = Distance(scoreMatrix[i], IdealNeg);

                double C = S_Minus/(S_Minus+S_plus);
                if (C > max_C)
                {max_C = C; RAT_Chosen = i+1;} //Find the high Cj index
        }

        if (scoreMatrix.length ==1)
        {RAT_Chosen = 1;}

        int[] counter2 = new int[candidates.get(0).size()];
        Dec2Bi(RAT_Chosen, counter2);
        if (InterfaceChanged(counter2) || currentConnection.size() == 0)
        //If interface has changed, or the user is not connected
        {
                if (outofrange == true && currentConnection.size() != 0)
                //If predicted to move out of all RATs and
                //the user is connected
                {
                        boolean carryon = true;
                for (int i =0;i < currentConnection.size();i++)
                //check if its possible to maintain current connection
                {
                        if (!inRange(position,currentConnection.get(i)))
                        {
                                carryon = false;
                        }
                }

                if (carryon == false)
                {
                        MakeConnection(RAT_Chosen,scoreMatrix);
                        deadTime = window_size;
                }
        }

        else if (deadTime > 0 && currentConnection.size() != 0)
        //Still within dead time
        {
                boolean Maintain =true;
                for (int i =0;i<currentConnection.size();i++)
                {
                        if (!inRange(position,currentConnection.get(i)))
                        {Maintain = false;}
                }

                if (!Maintain)
                {
                        MakeConnection(RAT_Chosen, scoreMatrix);
                        deadTime = window_size;
                }
        }

        else
        {
                MakeConnection(RAT_Chosen,scoreMatrix);
                deadTime = window_size;
        }
}
```

```
private boolean InterfaceChanged(int[] counter) {
// TODO Auto-generated method stub
        int size = 0;
        for (int i =0;i<counter.length;i++)
        {
                if (counter[i] ==1)
                {size++;}
        }

        if (size != currentConnection.size())
        //Does number of interfaces match?
        {return true;}

        else
        {
                for (int i =0;i<counter.length;i++)
                 //Are they the same interfaces?
                {
                        if (counter[i] == 1)
                        {
                                if (!currentConnection.contains
                                (candidates.get(0).get(i)))
                                {return true;}
                        }
                }
        }
        return false;
}

private void MakeConnection(int rAT_Chosen, double[][] scoreMatrix) {
// TODO Auto-generated method stub
        int[] counter = new int[candidates.get(0).size()];
        Dec2Bi(rAT_Chosen,counter);
        int req = 0;
        switch(type)
        {
                case 2:
                        req = 150;
                break;
                case 3:
                        req = 2000;
                break;
                case 4:
                        req = 64;
                break;
                case 5:
                        req = 20;
                break;
        }

        int[] alloc = bandAlloc(type, req*rate_mult, counter, scoreMatrix);
        //Get bandwidth allocation

        for (int i =0;i<currentConnection.size();i++)
        {
                currentConnection.get(i).release(this);
                //Release all the old resources it was holding
        }
        currentConnection.clear();
        for (int i =0;i<counter.length;i++)//Obtain new resources
        {
                if (counter[i] ==1)
                {
                        candidates.get(0).get(i).reserve(this,alloc[i]);
                        currentConnection.add(candidates.get(0).get(i));
                }
        }
```

```
        deadTime = window_size; //Reset dead time
}

private void Dec2Bi(int num, int[] counter) {//Decimal to Binary
        // TODO Auto-generated method stub
        for (int i =0;i<counter.length;i++)
        {counter [i] =0;}

        for (int i =0;i<num;i++)
        {
                binaryCountUp(counter);
        }
}

private double findMin(double[][] scoreMatrix, int column) {
//Find minimum of a column in score matrix
        // TODO Auto-generated method stub
        double min = 999999999;
        for (int i =0;i<scoreMatrix.length;i++)
        {
                if (scoreMatrix[i][column]<min)
                {min = scoreMatrix[i][column];}
        }
        return min;
}

private double findMax(double[][] scoreMatrix, int column) {
//Find max of a column in score matrix
// TODO Auto-generated method stub
        double max = 0;
        for (int i =0;i<scoreMatrix.length;i++)
        {
                if (scoreMatrix[i][column]>max)
                {max = scoreMatrix[i][column];}
        }
        return max;
}

private void Weight(double[][] scoreMatrix, double[] weight) {
//Weight the score matrix
// TODO Auto-generated method stub
        for (int i =0;i<scoreMatrix.length;i++)
        {
                for (int j=0;j<3;j++)
                {
                        scoreMatrix[i][j] = scoreMatrix[i][j] * weight[j];
                }
        }
}

private void normalise(double[][] scoreMatrix) {//Normalise  score matrix
// TODO Auto-generated method stub
//  [COST POWER BAND UTILITY]
        CostNormalise(scoreMatrix,0);
        CostNormalise(scoreMatrix,1);
        BenefitNormalise(scoreMatrix,2);
}

private void BenefitNormalise(double[][] scoreMatrix, int column) {
//Benefit normalisation
// TODO Auto-generated method stub
        double temp = 0;
        for (int i =0;i<scoreMatrix.length;i++)
        {
                temp +=Math.pow(scoreMatrix[i][column], 2);
        }
        temp = Math.sqrt(temp);
        for (int i =0;i<scoreMatrix.length;i++)
        {
```

```
                double res = scoreMatrix[i][column]/(temp);
                scoreMatrix[i][column] = res;
        }
}

private void CostNormalise(double[][] scoreMatrix, int column) {
//Cost normalisation
// TODO Auto-generated method stub
        double temp = 0;
        for (int i =0;i<scoreMatrix.length;i++)
        {
                if (scoreMatrix[i][column]!=0)
                {
                        temp +=Math.pow(1/scoreMatrix[i][column], 2);
                }
        }
        temp = Math.sqrt(temp);

        for (int i =0;i<scoreMatrix.length;i++)
        {
                if (scoreMatrix[i][column] != 0)
                {
                        double res = 1/(scoreMatrix[i][column]*temp);
                        scoreMatrix[i][column] = res;
                }
                else
                {
                        scoreMatrix[i][column] = 0;
                }
        }
}

private int Bi2Dec(int[] counter)//Binary to decimal
{
        int temp = 0;
        int pow =0;
        for (int i = counter.length -1 ;i>-1; i--)
        {
                temp += (int)(Math.pow(2, pow))*counter[i];
                pow++;
        }
        return temp;
}

private int[] binaryCountUp(int[] counter) {
//Increment the binary counter
// TODO Auto-generated method stub
        for (int i =counter.length-1;i>-1;i--)
        {
                if (counter[i] == 0)
                {
                        counter[i] = 1;
                        for (int j = counter.length-1;j>i;j--)
                        {counter[j] = 0;}
                        break;
                }
        }
        return counter;
}

private void normalise(double[] vec) {
//Used to normalise weight vector
// TODO Auto-generated method stub
        int size = vec.length;
        double sqr = 0;
        for (int i =0;i<size;i++)
        {
                sqr += vec[i] *vec[i];
        }
```

```
        for (int i =0;i<size;i++)
        {
                vec[i] = vec[i]/Math.sqrt(sqr);
        }
}

private double[] findMaxMin(int type, double[][] scoreMatrix) {
// TODO Auto-generated method stub
/*
* Type of Traffic
* 1 - Elastic, Breq = limitless
* 2 - Adaptive, Breq = 150 kbps
* 3 - Adaptive, Breq = 2000 kbps
* 4 - Hard-Real time, Breq = 64 kbps
* 5 - Hard-Real time, Breq = 20 kbps
*/
        int req = 0;
        //System.out.println(type);
        switch(type)
        {
                case 2:
                        req = 150;
                break;
                case 3:
                        req = 2000;
                break;
                case 4:
                        req = 64;
                break;
                case 5:
                        req = 20;
                break;
        }


        int[] counter = new int[candidates.get(0).size()];
        double[] max = new double[6];

        for (int i =0;i <counter.length;i++)
        {counter[i] = 0;}
        double max_cost = 0;
        double max_power = 0;
        double max_band = 0;
        double min_cost= 99999999;
        double min_power = 999999;
        double min_band = 999999;


        for (int i =0; i <Math.pow(2, counter.length)-1;i++)
        {
        binaryCountUp(counter);
        double temp_cost = 0;
        double temp_power = 0;
        double temp_band = 0;
        int[] alloc = new int[counter.length];
        alloc = bandAlloc(type,rate_mult*req,counter,scoreMatrix);
        if (type!=1)
        {
                for (int j =0;j<alloc.length;j++)
                {
                        temp_cost+= alloc[j] * candidates.get(0).get(j).
                        getCost();//Cost
                        temp_power += alloc[j] *candidates.get(0).get(j).
                        getPower()+alloc[j] *candidates.get(0).get(j).
                        getPower()*(Distance(position,candidates.get(0)
                        .get(j).getPosition()))/candidates.get(0).get(j)
                        .getCover();
                }
```

```
                temp_band = Utility(alloc,req,phi);

        }
        else
        {
                for (int j =0;j<counter.length;j++)
                {
                        if (counter[j] ==1)
                        {
                                temp_cost += candidates.get(0).get(j).
                                getCost()*candidates.get(0).get(j)
                                .getAvailBand();

                                temp_power += candidates.get(0).get(j).
                                getAvailBand()*candidates.get(0).get(j)
                                .getPower()+(Distance(position, candidates
                                .get(0).get(j).getPosition())*
                                candidates.get(0).get(j).getAvailBand()
                                *candidates.get(0).get(j).getPower())
                                /candidates.get(0).get(j).getCover();
                        }
                }
                temp_band = Utility (alloc,req,phi);
        }

                scoreMatrix[Bi2Dec(counter)-1][0] = temp_cost;
                scoreMatrix[Bi2Dec(counter)-1][1] = temp_power;
                //Establishing score matrix
                scoreMatrix[Bi2Dec(counter)-1][2] = temp_band;

                if (temp_cost > max_cost)
                {max_cost = temp_cost;}

                if (temp_power > max_power)
                {max_power = temp_power;}

                if (temp_band>max_band)
                {max_band = temp_band;}

                if (temp_cost < min_cost)
                {min_cost = temp_cost;}

                if (temp_power < min_power)
                {min_power = temp_power;}

                if (temp_band<min_band)
                {min_band = temp_band;}
        }

        max[0] =max_cost; max[1] =max_power; max[2] =max_band;
        max[3]=min_cost;max[4] = min_power;max[5] = min_band;
        return max;
}



private double Utility(int[] alloc, double req, int phi2) {
//Utility function of bandwidth
// TODO Auto-generated method stub
/*
* Type of Traffic
* 1 - Elastic, Breq = limitless
* 2 - Adaptive, Breq = 150 kbps
* 3 - Adaptive, Breq = 2000 kbps
* 4 - Hard-Real time, Breq = 64 kbps
* 5 - Hard-Real time, Breq = 20 kbps
*/
        int total =0;
        double uti =0;
```

```
        for (int i =0;i<alloc.length;i++)
        {
                if (type ==1)
                {
                        total += alloc[i] * (1-Distance(position,candidates
                        .get(0).get(i).getPosition())/candidates
                        .get(0).get(i).getCover());
                }
                else
                {
                        total += alloc[i];
                }
        }


        switch(type)
        {
                case 1:
                        uti = 1- Math.pow(Math.E, -total/(double)phi);
                break;

                case 2:
                case 3:
                if (total >req)
                {
                        uti = 1- Math.pow
                        (Math.E, -(total-req)/(double)phi);
                }

                else
                {uti = 0.0000001;}
                break;

                case 4:
                case 5:
                        if (total > req)
                        {uti =1;}
                else
                        {uti = 0.000001;}
                break;
        }
        return uti;
}

private double Distance(double[] position1, double[] position2) {
//Calculate euclean distance
// TODO Auto-generated method stub
        double temp =0;
        for (int i = 0;i<position1.length;i++)
        {
                temp += Math.pow(position1[i]-position2[i],2);
        }
        return Math.sqrt(temp);
}

private int[] bandAlloc(int type, double req,
int[] counter, double[][] scoreMatrix) {
//Bandwidth Allocation
// TODO Auto-generated method stub
/*
* Type of Traffic
* 1 - Elastic, Breq = limitless
* 2 - Adaptive, Breq = 150 kbps
* 3 - Adaptive, Breq = 2000 kbps
* 4 - Hard-Real time, Breq = 64 kbps
* 5 - Hard-Real time, Breq = 20 kbps
```

```
*/
        int total_cap = 0;
        int[] alloc = new int[counter.length];
        if (type !=1);
        {
                for (int i =0; i<counter.length;i++)
                {
                        if (counter[i] ==1)
                        {
                                total_cap += candidates.get(0).get(i)
                                .getRemain();
                        }
                }

                if (total_cap <req)
                {
                        for (int j =0;j<3;j++)
                        {
                                scoreMatrix[Bi2Dec(counter)-1][j] =0;
                        }
                }

                else
                {
                        for (int i =0;i<counter.length;i++)
                        {
                                if (counter[i] ==1)
                                {
                                int allocate = (int)((double)candidates
                                .get(0).get(i).getRemain()/total_cap *req);
                                if (allocate<1){allocate =1;}
                                alloc[i] = allocate;
                                }

                                else
                                {
                                        alloc[i] = 0;
                                }
                        }
                }
        }



        if (type == 1)
        {
                for (int i =0;i<counter.length;i++)
                {
                        if (counter[i] ==1)
                        {
                                int allocate = candidates.get(0).get(i)
                                .getAvailBand();
                                alloc[i]=allocate;
                        }
                }
        }
        return alloc;
}

public double[] Kalman(double pos, double velo, double acc)
{
        double[] predictions = new double[2*window_size];
        // A = [ 1 dt ]
        //     [ 0  1 ]
        RealMatrix A = new Array2DRowRealMatrix(new double[][]
         { { 1, dt }, { 0, 1 } });

        // B = [ dt^2/2 ]
        //     [ dt     ]
```

```
        RealMatrix B = new Array2DRowRealMatrix(new double[][]
        { { Math.pow(dt, 2d) / 2d }, { dt } });
        // H = [ 1 0 ]
        RealMatrix H = new Array2DRowRealMatrix(new double[][]
         { { 1d, 0d } });
        // x = [ position velocity ]
        RealVector x = new ArrayRealVector(new double[] { pos, velo });

        RealMatrix tmp = new Array2DRowRealMatrix(new double[][] {
        { Math.pow(dt, 4d) / 4d, Math.pow(dt, 3d) / 2d },
        { Math.pow(dt, 3d) / 2d, Math.pow(dt, 2d) } });
        // Q = [ dt^4/4 dt^3/2 ]
        //     [ dt^3/2 dt^2   ]
        RealMatrix Q = tmp.scalarMultiply(Math.pow(accelNoise, 2));

        // P0 = [ 1 1 ]
        //      [ 1 1 ]
        RealMatrix P0 = new Array2DRowRealMatrix(
        new double[][] { { 1, 1 }, { 1, 1 } });

        // R = [ measurementNoise^2 ]
        RealMatrix R = new Array2DRowRealMatrix(new double[]
        { Math.pow(measurementNoise, 2) });

        // constant control input, increase velocity by 0.1 m/s per cycle
        RealVector u = new ArrayRealVector(new double[] { acc });

        ProcessModel pm = new DefaultProcessModel(A, B, Q, x, P0);
        MeasurementModel mm = new DefaultMeasurementModel(H, R);
        KalmanFilter filter = new KalmanFilter(pm, mm);

        RandomGenerator rand = new JDKRandomGenerator();

        RealVector tmpPNoise = new ArrayRealVector(new double[]
        { Math.pow(dt, 2d) / 2d, dt });
        RealVector mNoise = new ArrayRealVector(1);

        for (int i = 0; i < window_size; i++) {
                filter.predict(u);

                // simulate the process
                RealVector pNoise = tmpPNoise.mapMultiply
                (accelNoise * rand.nextGaussian());

                // x = A * x + B * u + pNoise
                x = A.operate(x).add(B.operate(u)).add(pNoise);

                // simulate the measurement
                mNoise.setEntry(0, measurementNoise * rand.nextGaussian());

                // z = H * x + m_noise
                RealVector z = H.operate(x).add(mNoise);

                filter.correct(z);

                double posi = filter.getStateEstimation()[0];
                double velocity = filter.getStateEstimation()[1];

                predictions[i] = posi;
                predictions[i+window_size] = velocity;
                //[postion1,2,....n, velocity 1,2,...m]
        }
        return predictions;
}

public static void write(String name, String input)
{
        try
        {
```

```
                String filename= name;
                FileWriter fw = new FileWriter(filename,true);
                fw.write(input);
                fw.close();
        }
        catch(IOException ioe)
        {
                System.err.println("IOException: " + ioe.getMessage());
        }
}

public void fillCandidate(double[] userPos, int listNum)
//Fill candidates according to predictions
{
        boolean handoff = true;
        if (currentConnection.size() == 0)
        {handoff = false;}

        for (int i =0;i<rats.size();i++)
        {

                if (handoff)
                {
                        if (inRange(userPos,rats.get(i)) &&
                        !(rats.get(i).isFull()))
                        {
                                candidates.get(listNum).add(rats.get(i));
                                originalCandidates.get(listNum)
                                .add(rats.get(i));
                        }
                }

                else
                {
                        if (inRange(userPos,rats.get(i))
                        && rats.get(i).spaceForNew())
                        {
                                candidates.get(listNum).add(rats.get(i));
                                originalCandidates.get(listNum)
                                .add(rats.get(i));
                        }
                }
        }
}


private void refineList() {
//Take out candidates that do not appear in all candidate sub lists
// TODO Auto-generated method stub
        int size = originalCandidates.get(0).size();
        ArrayList<RAT> inRangeNow = new ArrayList<RAT>();
        for (int i =0;i<rats.size();i++)
        {
                if (inRange(position, rats.get(i)))
                {
                        inRangeNow.add(rats.get(i));
                }
        }

        for (int i = 0; i < size;i++)
        {
                for (int j = 0;j<originalCandidates.size();j++)
                {
                        if (!candidates.get(j).contains(originalCandidates
                        .get(0).get(i)))
                        {
                                removeRAT(originalCandidates.get(0)
                                .get(i));
                        }
```

```
                }
        }

        for (int i =0;i<inRangeNow.size();i++)
        {
                if (!candidates.get(0).contains(inRangeNow.get(i)))
                {
                        removeRAT(inRangeNow.get(i));
                }
        }
}

private void removeRAT(RAT rat) { //Remove a RAT from candidate list
// TODO Auto-generated method stub
        for (int i = 0; i < candidates.size();i++)
        {
                if (candidates.get(i).contains(rat))
                {candidates.get(i).remove(rat);}
        }
}

public boolean inRange(double[] userPos, RAT rat)
//If a user is in range of RAT
{
        double x = userPos[0] - rat.getPosition()[0];
        double y = userPos[1] - rat.getPosition()[1];
        if ((Math.sqrt(x*x+y*y) + 2 )<= rat.getCover())
        // Uncertainty is 2!
        {return true;}

        else
        {
        return false;
        }
}


public void fillSheet()
{
        String builder =String.valueOf(position[0] +","+ position[1]);
        for (int i=0; i <window_size;i++)
        {
                builder += "," + x_predictions[i] ;
        }


        for (int i=0; i <window_size;i++)
        {
                builder += "," + y_predictions[i] ;
        }
        write(Integer.toString(this.ID)+".csv", builder);
}

public String getMN() {
        // TODO Auto-generated method stub
        return String.valueOf((int)measurementNoise);
}

public void setID(int ID)
{this.ID = ID;}

public void getID(int ID) {
        // TODO Auto-generated method stub
        this.ID = ID;
}

public int getID() {
// TODO Auto-generated method stub
        return ID;
```

```
}

public int getType() {
// TODO Auto-generated method stub
        return type;
}

public boolean isFinished()
{
        return finished;
}

public double[] getPosition()
{return position;}

public ArrayList<ArrayList<RAT>> getCandidates()
{return candidates;}

public int getWindow() {
// TODO Auto-generated method stub
        return window_size;
}
}
```

RAT.JAVA:

```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;

public class RAT {
        int type;
        int ID;
        int total_cap = 0;
        int Remain_capacity = 0;
        int single_cap = 0;
        int power_use = 0;
        int guard_cap = 0;
        double cost = 0;
        double power_cost = 0;
        int priority = 0;
        double position[] = {0,0};
        double coverage = 0;

        ArrayList<App> current_App = new ArrayList<App>();
        Map<App, Integer> map = new HashMap<App, Integer>();

        public RAT (double power, int type, double cost, int capacity
        , int guard,double coverage,double[] position)
        {
                this.power_cost = power;
                this.type = type;
                this.cost = cost;
                this.total_cap = capacity;
                this.guard_cap = guard;
                this.coverage = coverage;
                this.position = position;
                Remain_capacity = total_cap;
                switch(type)
                {
                        case 1:
                        single_cap = 2000; //3G
                        break;
                        case 2:
                        single_cap = 1000000;//4g
                        break;
                        case 3:
```

```
                    single_cap = 10000000;//5G
                    break;
                    case 4:
                    single_cap = 54000;//wifi
                    break;
            }
    }

    public double[] getPosition()
    {return position;}

    public double getCover()
    {return coverage;}

    public void setID(int i) {
    // TODO Auto-generated method stub
    this.ID = i;
    }

    public int getID() {
    // TODO Auto-generated method stub
    return ID;
    }

    public boolean isFull()
    {return (Remain_capacity <0);}

    public boolean spaceForNew()
    {return (total_cap-Remain_capacity)<guard_cap;}

    public double getCost() {
    // TODO Auto-generated method stub
    return cost;
    }

    public int getAvailBand() {
    // TODO Auto-generated method stub
    if (Remain_capacity>= single_cap)
    {return single_cap;}

    else
    {return Remain_capacity;}
    }

    public int getRemain() {
    // TODO Auto-generated method stub
    return Remain_capacity;
    }

    public double getPower() {
    // TODO Auto-generated method stub
    return power_cost;
    }

    public void reserve( App app, int bandwidth) {
    // TODO Auto-generated method stub
    if (!current_App.contains(app))
    {current_App.add(app);}

    int temp = 0;

    if (map.get(app) != null)
    {temp = map.get(app);}

            map.put(app, temp + bandwidth);
            Remain_capacity = Remain_capacity - bandwidth;
            //System.out.println(Remain_capacity);
    }
```

```
        public int getAppUsage(App app)
        {
        return map.get(app);
        }

        public void release(App app) {
                // TODO Auto-generated method stub
                if (current_App.contains(app))
                {current_App.remove(app);}

                if (map.containsKey(app))
                {
                        Remain_capacity += map.get(app);
                        map.remove(app);
                }

        }

        public int getSingle() {
        // TODO Auto-generated method stub
        return this.single_cap;
        }

        public int getCurrent() {
        // TODO Auto-generated method stub
        return current_App.size();
        }

        public int elasticAlloc() {
        // TODO Auto-generated method stub"
        int alloc = 0;
                for (int i =0;i<current_App.size();i++)
                {
                        if (current_App.get(i).getType() == 1)
                        {alloc += map.get(current_App.get(i));}
                }
                return alloc;
        }

        public int inelasticAlloc() {
        // TODO Auto-generated method stub"
                int alloc = 0;
                for (int i =0;i<current_App.size();i++)
                {
                        if (current_App.get(i).getType() != 1)
                        {alloc += map.get(current_App.get(i));}
                }
        return alloc;
        }
}
```