Performance estimation of wireless networks using traffic generation and monitoring on a mobile device.

by

Ghislaine Livie Ngangom Tiemeni

A thesis submitted for the degree of

Magister Scientiae

in the

Department of Computer Science

University of the Western Cape

UNIVERSITY of the
WESTERN CAPE

2015

University of the Western Cape

Abstract

Faculty of Science

Department of Computer Science

by **Ghislaine Livie Ngangom Tiemeni**

Supervisor: Prof. Isabella M. Venter

Co-supervisor: Prof. William D. Tucker

In this study, a traffic generator software package namely MTGawn was developed to run packet generation and evaluation on a mobile device. The call generating software system is able to: simulate voice over Internet protocol calls as well as user datagram protocol and transmission control protocol between mobile phones over a wireless network and analyse network data similar to computer-based network monitoring tools such as Iperf and D-ITG but is self-contained on a mobile device. This entailed porting a 'stripped down' version of a packet generation and monitoring system with functionality as found in open source tools for a mobile platform. This mobile system is able to generate and monitor traffic over any network interface on a mobile device, and calculate the standard quality of service metrics. The tool was compared to a computer–based tool namely distributed Internet traffic generator (D-ITG) in the same environment and, in most cases, MTGawn reported comparable results to D-ITG. The important motivation for this software was to ease feasibility testing and monitoring in the field by using an affordable and rechargeable technology such as a mobile device. The system was tested in a testbed and can be used in rural areas where a mobile device is more suitable than a PC or laptop. The main challenge was to port and adapt an open source packet generator to an Android platform and to provide a suitable touchscreen interface for the tool.

ACM Categories and Subject Descriptors

B.8 [PERFORMANCE AND RELIABILITY]

B.8.2 [Performance Analysis and Design Aids]

C.4 [PERFORMANCE OF SYSTEMS] Measurement techniques, Performance attributes

Keywords: performance analysis and design aids; measurement techniques; performance attributes, traffic generator.

DECLARATION

I declare that *performance estimation of wireless networks using traffic generation and monitoring on a mobile device* is my own work, that it has not been submitted for any degree or examination in any other university, and that all the sources I have used or quoted have been indicated and acknowledged by complete references.

Signed: _____

Date: _____

ACKNOWLEDGMENTS

First of all, I would like to thank the Almighty GOD for his endless love and benedictions; GOD is my source of inspiration and the foundation of everything I have achieved in life.

I wish to express my sincere appreciation and gratitude to my supervisor, Professor Isabella M. Venter for her assistance, guidance and limitless encouragement. She inspired me through her passion for good research and her admirable work ethics. In addition, special thanks to my co-supervisor Professor William D. Tucker whose insightful criticisms and valuable ideas taught me new skills.

Thanks to Prof Bagula and Carlos for their helpful advice and thanks also to the staff members of the computer science department for their assistance.

I would also like to thank Ms Ann Bytheway, for professionally editing the thesis providing thoughtful comments and suggestions.

Thanks to Josée, Chola, Jean-Jules, Suzy, Alida and all my friends as well as my classmates for their touching encouragement.

Finally, limitless thanks to my lovely family especially; my mother Marie, my sister Alvine, my brother Thierry and his wife Saurelle, my nephew Nathan, my nieces Victoria and Francine; for their unconditional love, their motivation, their unrestricted financial and moral support.

CONTENTS

LIST OF FIGURES

Number *Page*

LIST OF TABLES

Number *Page*

LIST OF CODE SNIPPETS

Number *Page*

LIST OF PUBLICATIONS

Ngangom-Tiemeni, GI, Venter IM and Tucker, WD. (2014). Performance Evaluation of a Wireless Network Using a VoIP traffic generator on a mobile device. *South African Institute for Computer Scientists and Information Technologists Conference (SAICSIT 2014)*, 30 September – 1 October, Centurion, ACM 978-1-4503-3246-0/14/09 http://dx.doi.org/10.1145/2664591.2664626 (Pages 297-303)

Ngangom-Tiemeni, GI, Venter IM, and Tucker WD. (2014). Rural Wireless Mesh Network Analysis on the Go. *Proceedings of the Southern African Telecommunication Networks and Applications Conference SATNAC 2014*, edited by Roy Volkwyn. Port Elizabeth, South Africa, 1 to 3 September 2014. (Pages 307 – 312) (ISBN978-0-620-61966-0).

Ngangom-Tiemeni, GI, Venter IM, Rey-Moreno C and Tucker WD. (2013). A Mobile Platform Traffic Generator for Network Performance Evaluation. *Proceedings of the Southern African Telecommunication Networks and Applications Conference SATNAC 2013*, Stellenbosch, South Africa, 1to 5 September 2013 (Pages 447-448) (ISBN 978-0-620-57883-7)

Ngangom-Tiemeni, GL, Venter IM, Rey-Moreno C and Tucker WD. (2013). Rural wireless network performance analysis on the go. *Faculty of Science Research Open Day 2013*, Bellville, South Africa, 30 October 2013

Ngangom-Tiemeni, GL, Venter IM, and Tucker WD. (2013). A Mobile Tool for Rural Wireless Network Performance Analysis. *South African Institute for Computer Scientists and Information Technologists Conference* (SAICSIT 2013), 7 – 9 October, East London.

ABBREVIATIONS

CDF - Cumulative distribution function

CPU - Central Processing Unit

cRTP - Compressed Real Time Protocol

DSR - Design Science Research

D-ITG - Distributed Internet Traffic Generator

DSRM - Design Science Research Methodology

FTP - File Transfer Protocol

HTTP - HyperText Transfer Protocol

IP - Internet Protocol

IPP - Interrupted Poisson Process

LRD - Long-range dependence

MP - Mesh Potato

MTGawn - Mobile Traffic Generator for Analysis of Wireless Networks

NTP - Network Time Protocol

OS - Operating System

OWD - One-way delay

PDF - Probability Density Function

PMF - Probability Mass Function

QoS - Quality of Service

RAM - Random Access Memory

RTP - Real Time Protocol

RTT - Round Trip Time

SMTP - Simple Mail Transfer Protocol

SNMP - Simple Network Management Protocol

SoC - System on Chip

SRD - Short-range dependence

TCP - Transmission Control Protocol

UDP - User Datagram Protocol

VAD - Voice Activity Detection

VoIP - Voice over Internet Protocol

Wi-Fi - Wireless Fidelity

WMN - Wireless Mesh Network

Chapter 1

INTRODUCTION

1.1 BACKGROUND

Mobile communication forms an integral part of most people's daily lives. In fact, the usage of mobile phones has increased considerably over the past decades for numerous reasons: mobile technologies are indispensable tools for worldwide communications due to their ultra-portable size and convenience; their relative affordability makes them essential for emergencies, travelling and everyday conversations; and mobile phones make it easy and fast to connect with people almost anywhere (and everywhere) in the world to share information anytime. As mobile technology continues to be adopted, the complexity of mobile devices has increased accordingly. In this sense, mobile phones have become multifunctional tools, which offer integrated, and multimedia functionalities. As a consequence, mobile phones have developed, not only as tools to make calls, but also to act as computers, as cameras with multimedia technologies, as calculators, as radios, music players, schedulers, and as instruments that allow for the surfing of the Internet, checking of email and so on.

As technology has evolved, the deployment of multimedia applications such as voice-over Internet protocol (VoIP)—in addition to traditional data services—has significantly increased communication traffic over the Internet (Cheng, Mohapatra, Lee, & Banerjee, 2008) (Fang, 2005). The availability of cheap mobile phones with Wi-Fi capabilities has increased the interest in evaluating the quality of service (QoS) of wireless networks in particular for VoIP (service provided by Wi-Fi networks) to mobile devices. Performance evaluation techniques are useful for the analysis of the QoS of networks whereas network monitoring is used to analyse the performance of networks. For the purpose of network performance analysis, researchers rely on passive and active monitoring techniques. Passive techniques use packet sniffers to watch and capture the traffic flowing on the network; performing the analysis of the traffic without changing it. It requires real end-user traffic and often requires the collection of large amount of data in order to get all the relevant information needed. The data is analysed off-line and not as it is collected so it may be challenging to process large collected data sets (Cecil, 2012). Active monitoring is an alternative to passive monitoring since active monitoring solutions do not rely on real traffic but rather on synthetic traffic characteristics of real user behaviour. However, the main concern with

active monitoring is to develop an appropriate model that will represent the features of realistic traffic flows. Active techniques rely on traffic generator tools to inject packets into a network, and then follow these packets to extract the relevant performance metrics of these packets. Traffic generator software is valuable for generating synthetic, yet realistic workloads that can be analysed to test the quality of service of communication networks. To capture the statistical characteristics of realistic traffic flows, traffic generators require accurate traffic models. If the traffic models do not accurately represent actual traffic then the network performance might be over- or under-estimated (Adas, 1997).

Several researchers have done work towards generating realistic network traffic patterns to assist with performance evaluation (Garcia, 2010) (Orebaugh, Ramirez, Burke, Morris, Pesce, & Wright, 2007) (Avallone, Guadagno, Emma, Pescape, & Ventre, 2004). Literature on the most interesting tools used to estimate the performance of networks (by means of generating realistic traffic) was reviewed in the initial stages of this study. Despite the powerful features of these tools, the literature survey indicated that very little research or development has been done to simplify feasibility testing and monitoring of wireless networks especially those deployed in difficult to reach areas, such as rural areas. In this research, a new tool—called MTGawn, the abbreviation for 'Mobile Traffic Generator for analysis of wireless networks'—was developed to fill this gap. Many computer-based applications are now being deployed on mobile phones making these applications easy to access and use. For this reason, MTGawn was deployed on a mobile device in order to permit the evaluation of network performance in remote areas. Remote areas in South Africa typically lack resources and the deployment of computers in such areas is difficult and could also be impractical. The tool (MTGawn) was developed to ease feasibility testing and monitoring in the field, using an affordable and lightweight technology such as a mobile device, which can be charged using solar panels in areas which lack resources such as electricity.

The main objective of MTGawn was to generate network packets and to send these packets over a Wi-Fi interface of a mobile 'sender' device. A mobile 'receiver' device was then set-up to receive the transmitted packets at the other endpoint of the network to extract performance metrics of interest such as throughput, delay, packet loss and jitter. MTGawn can generate a rich variety of traffic and can simultaneously generate multiple flows between a sender and receiver for analysis.

1.2 PROBLEM STATEMENT AND RESEARCH QUESTION

The aim of this research was to develop a mobile traffic generator that would allow the evaluation of the performance of wireless networks, using a mobile device. Thus, the research question posed was:

"How should a mobile traffic generator be designed to be feasible in order to analyse and evaluate the performance of wireless networks?"

The primary steps of designing a traffic generator involve the modelling of the traffic flowing through a network and then to inject emulated flows into the network. The problem of modelling synthetic but realistic network traffic patterns, characteristic of real user behaviour, and injecting them into the network, has been explored for more than a decade and continues to attract research interest, not only for the purpose of network performance analysis, but also for the development of new services and protocols and for the design of communication systems. Modelling network traffic flows involves the design of a stochastic model capable of representing the relevant features of real-life traffic flows (Adas, 1997) (Hasib, 2006). It is difficult to develop models that capture the statistical characteristics of actual traffic without over- or under-estimating the network performance.

The initial research question can thus be further unpacked as four sub-questions:

a. What techniques should be used for network monitoring and performance evaluation?
b. How should an appropriate traffic model be developed to emulate realistic traffic flows for mobile wireless networks?
c. How should the emulated traffic, over the Wi-Fi interface of a mobile device, be generated?
d. What performance metrics should be taken into consideration?

1.3 RESEARCH FRAMEWORK

A research methodology is the overall approach or design behind the choice of specific methods, which links the choice of methods to the desired outcomes (Crotty, 1998). The methods refer to the techniques used to gather and analyse data with respect to a specific research question or hypothesis (Crotty, 1998). In this research, Design Science Research (DSR) (Brocke & Buddendick, 2006) was the preferred methodology for addressing the specified research questions. DSR is defined as an iterative research process, which consists of six phases: identify; build; document; select; evaluate; and communicate. The knowledge base of theories and existing artefacts –

collected through document analysis and literature surveys – was fed into the design of the prototype.

As previously stated, in network monitoring, the performance evaluation is based either on active or passive methods. The main difference between these two methods is that active methods generate and inject data traffic in the network, whereas passive methods do not, these methods only observe traffic in the network. This research adopted the active method to address the research questions posed.

1.4 RESEARCH AIMS AND GOALS

The research questions were addressed by applying the DSR research framework throughout the research process. During the literature survey and document analysis it was found that no mention was made of a traffic generator tool capable of running on mobile devices (which typically have limited screen surface and limited computing power) while having performance analysis abilities. While reviewing some of the prominent performance evaluation tools, the possible contribution of this work was identified: the aim was thus to make available a cost-effective tool that generates typical network traffic with the capability of collecting performance metrics on affordable and easily accessible devices such as mobile phones or tablets.

To achieve this aim, a network analysis and monitoring tool, MTGawn, able to run packet generation and evaluation on affordable mobile devices (see Figure 1.1) was thus developed. This was done in order to ease feasibility testing and monitoring in remote areas. A further aim was to make sure that, if grid electricity were lacking, then the device could be charged using in situ solar panels or a vehicle's battery power.

MTGawn is a self-contained package deployed on a mobile device. It can emulate traffic such as VoIP (voice over Internet protocol) traffic between mobile phones over a wireless network and analyse network data similar to a computer-based network monitoring tool such as D-ITG but is designed for a mobile device. The development entailed porting a 'stripped down' version of a packet generation and monitoring system to a mobile platform. Yet the main challenge was to adapt an open source packet generator to the Android platform and to provide a suitable touchscreen interface for the tool.

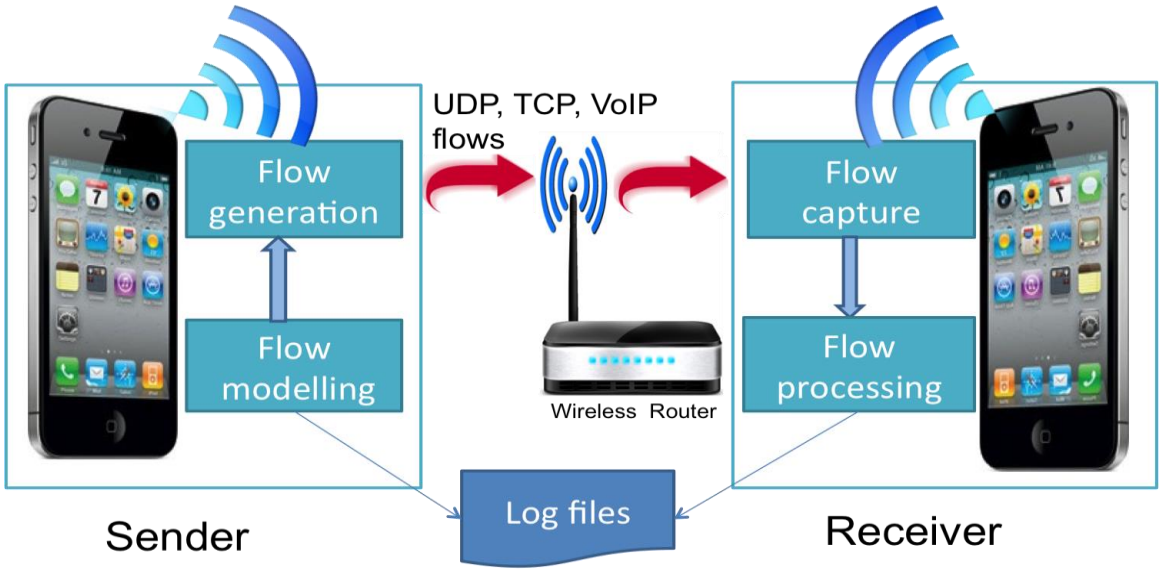


Figure 1.1: MTGawn system

In summary, the main goals of this research were as follows:

- Investigate current tools available for network packet generation and analysis and assess their usability.
- Explore the design of a stochastic model capable of representing the relevant features of real-life traffic flows.
- Emulate a variety of network traffic flows, generating different patterns of traffic, both deterministic and randomized using implementations of the most common transport protocols like transmission control protocol (TCP) and user datagram protocol (UDP).
- Send and receive network traffic over a network interface and extract standard network performance metrics such as throughput, delay, jitter and packet loss.
- Emulate different user activities by generating, sending and receiving multiple flows simultaneously.
- Ease feasibility testing and monitoring of wireless network by deploying the tool on mobile devices with Wi-Fi capabilities.
- Evaluate the tool with a wireless mesh network testbed deployed in the laboratory and compare the results obtained with an existing tool.

1.5 CONTRIBUTION

The results show that MTGawn is a tool that is able to send and receive network traffic over a network interface of a mobile device and extract standard network performance metrics such as throughput, delay, jitter and packet loss. The tool was compared to a computer–based tool namely distributed Internet traffic generator (D-ITG) in the same environment and, in most cases, MTGawn reported similar results to D-ITG.

Furthermore, network traffic generation is useful when performing the measurement of the traffic load to improve throughput and to limit delay for services such as VoIP, in order to optimize end-user experience thus this research is advantageous to the enhancement of quality of network service delivery.

1.6 THESIS OUTLINE

In this chapter the necessary background needed to understand the study was explained, the research questions were stated, the research approach was briefly mentioned and the contribution of this study was highlighted.

This work is further organised as follows:

Chapter 2: Literature review

Network traffic modelling and emulation is considered and presented. Network analysis tools and techniques are discussed and furthermore, a brief introduction to traffic generation software (as well as relevant samples) is presented. Also, a brief summary of network performance metrics considered in this research is also presented.

Chapter 3: Research methodology and design

The approach and methodology followed throughout the research process is presented. The design and implementation of the traffic generator tool MTGawn is discussed and the experiments conducted in order to collect data are explained. The discussion regarding the design includes the architecture, the application interface and the computation of the performance metrics taken into consideration for the network analysis.

Chapter 4: Results

The results obtained while evaluating the performance of a wireless network in a testbed environment is discussed. In addition, D-ITG and the MTGawn tool are compared and the results are explained.

Chapter 5: Discussion and conclusion

The salient points of the work are highlighted and conclusions drawn. In this chapter some suggestions for future work are discussed.

Chapter 2

LITERATURE REVIEW

The previous chapter dealt with the background and motivation for this study. The topic was refined, referring to related literature; to clarify the research problem and the research question. The research approach and methods that were adopted to address the stated research questions were also briefly mentioned.

In this chapter, the existing literature that relates to the research question is discussed in terms of each sub-question namely:

a. What techniques should be used for network monitoring and performance evaluation?

b. How should an appropriate traffic model be developed to emulate realistic traffic flows for mobile wireless networks?

c. How should the emulated traffic, over the Wi-Fi interface of a mobile device, be generated?

d. What performance metrics should be taken into consideration?

In the Section 2.1 **sub-question (a)** is examined by reviewing the techniques used by researchers for network monitoring and performance analysis. It will provide an overview of the approaches and techniques followed by researchers in this field. The Section 2.2 considers **sub-question (b)** by examining network traffic characterization and the network traffic models that have been used in the literature to represent realistic traffic flows. The Section 2.3 deals with **sub-question (c)** by considering how emulated traffic flows are transmitted and received over a network interface of an Android mobile device. The Section 2.4 examines **sub-question (d)** by describing the important parameters or metrics used to characterize the performance of a network during its evaluation.

2.1 TECHNIQUES USED FOR NETWORK MONITORING AND PERFORMANCE ANALYSIS

Network traffic analysis can be explained as the process of capturing or monitoring network traffic and inspecting it closely to determine what is happening on the network (Orebaugh, Ramirez, Burke, Morris, Pesce, & Wright, 2007). In network performance analysis, the objective is to evaluate a statistic of a metric related to the performance of the system. In communication

networks, it is very important for network administrators to be aware of and be able to handle the different types of traffic that traverse their networks. Traffic monitoring and analysis is essential in order to more effectively troubleshoot and resolve issues when they occur (Cecil, 2012). A performance monitoring system generates (or captures) representative packet flows in real or emulated network environments and creates representative workloads in a simulated environment. Active or passive monitoring approaches are used to measure performance characteristics of the traffic.

2.1.1 Passive monitoring

Numerous network-monitoring tools apply passive techniques to analyse the performance of networks. A passive approach uses hardware devices and packet sniffers to watch and capture the traffic flowing on the network. Figure 2.1 depicts a simple example of passive monitoring system where a network tap is placed between two network devices to capture the traffic. The monitoring device (e.g.: packet sniffer) connected to the network tap receives the same traffic as if it were in-line, including all the errors. A network tap is a hardware device, which provides a way to access the data flowing across a network by duplicating all traffic on the link and forwarding it to the monitoring port(s) (Oracle, 2009).

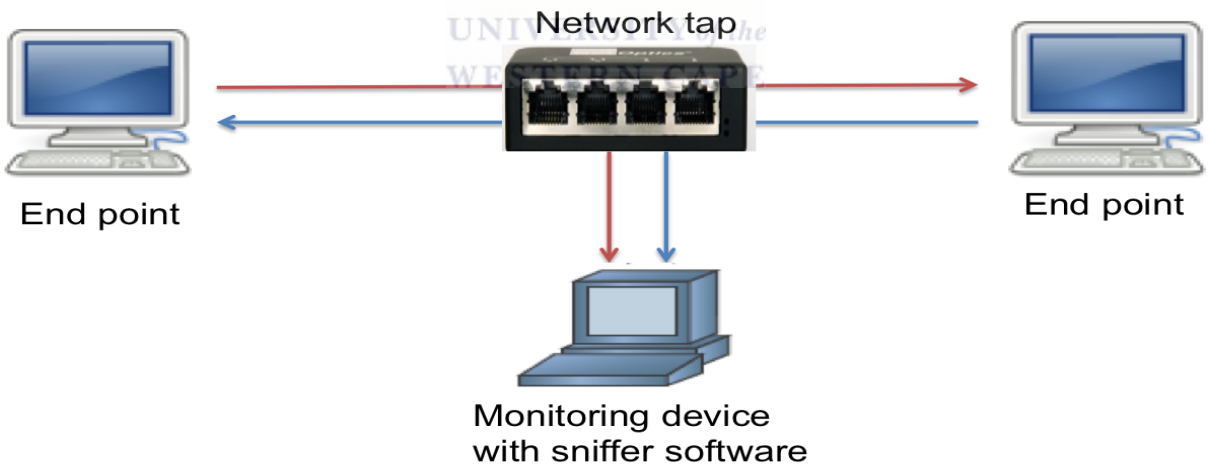


Figure 2.1: Passive monitoring using a network tab and sniffer software
Adapted from (Oracle, 2009)

Passive systems simply perform the analysis of the traffic that flows through the network, without changing it. They help to determine the characteristics of the traffic that flows through the measurement point, like the average rate, the mean packet size or the duration of the TCP connections (Veiga, Pinho, Oliveira, Valadas, Salvador, & Nogueira, 2005). Passive monitoring

computes traffic statistics that are helpful to identify the type of protocols involved when communication problems occur and to determine the bandwidth usage (Deri, 2004).

Some of the prominent passive tools include Wireshark and Tcpdump. Initially developed in 1997 by Gerald Combs, Wireshark (Orebaugh, Ramirez, Burke, Morris, Pesce, & Wright, 2007) (previously called Ethereal) has become one of the most popular tools for network monitoring and performance evaluation. Wireshark is a network protocol analyser, which can be used to read and capture files from a variety number of sources including from other sniffers (such as Tcpdump), routers and network utilities. It is available on both the Windows and the UNIX platform. It uses the well-known library Libpcap to capture data in a Libpcap-format but also has the ability to read the captured data in a variety of other formats. It currently supports over 750 protocols including VoIP. Wireshark has the ability to read packets and display data in ASCII (American Standard Code for Information Interchange) format—an easy to read format. It provides a graphical user interface to browse the captured data, viewing summary and detailed information for each packet. It implements a filter, which helps to find a desired packet without sifting through all of them. Wireshark uses both capture and display filters. The capture filter allows the capturing of certain types of traffic and the display filter provides a powerful syntax to sort the captured traffic.

Tcpdump is another prominent passive monitoring tool. Tcpdump is a powerful Unix-based command-line packet analyser. It also uses Libpcap, a portable C/C++ library for network traffic capture (Garcia, 2010). It has the ability to intercept and display packets being transmitted or received over a network. Tcpdump uses a program called Tcptrace to analyse network behaviour, and the performance of applications that generate or receive network traffic. Tcptrace (Ostermann, 2000) is a tool written by Shawn Ostermann at Ohio University, for the analysis of Tcpdump files**.** It has the ability to read and analyse a variety of other files generated by several popular packet-capture programs such as Windump (Windump is the Windows version of Tcpdump) and can produce several different types of output containing information on each connection seen, such as elapsed time, bytes and segments sent and received, retransmissions, round trip times, window advertisements, throughput, and more. It can also produce a number of graphs for further analysis.

2.1.2 Active monitoring

Active techniques evaluate the performance of networks by injecting traffic on the network between hosts and then performing measurements. The active approach provides explicit control of the generation of packets for measurement scenarios. These include control of the nature of traffic generation, the sampling techniques, the timing, frequency, scheduling, packet sizes and

types (to emulate various applications), statistical quality, the path and function chosen to be monitored (Cottrell, 2001). Active monitoring relies on traffic-generator software to generate synthetic but realistic traffic patterns that can assist in predicting and estimating the performance of networks. A basic example of the active approach is the command Ping which monitors remote Internet protocol (IP) hosts by sending Internet control message protocol (ICMP) packets to the remote host and the remote device sends echo response to acknowledge ICMP packets (see Figure 2.2).

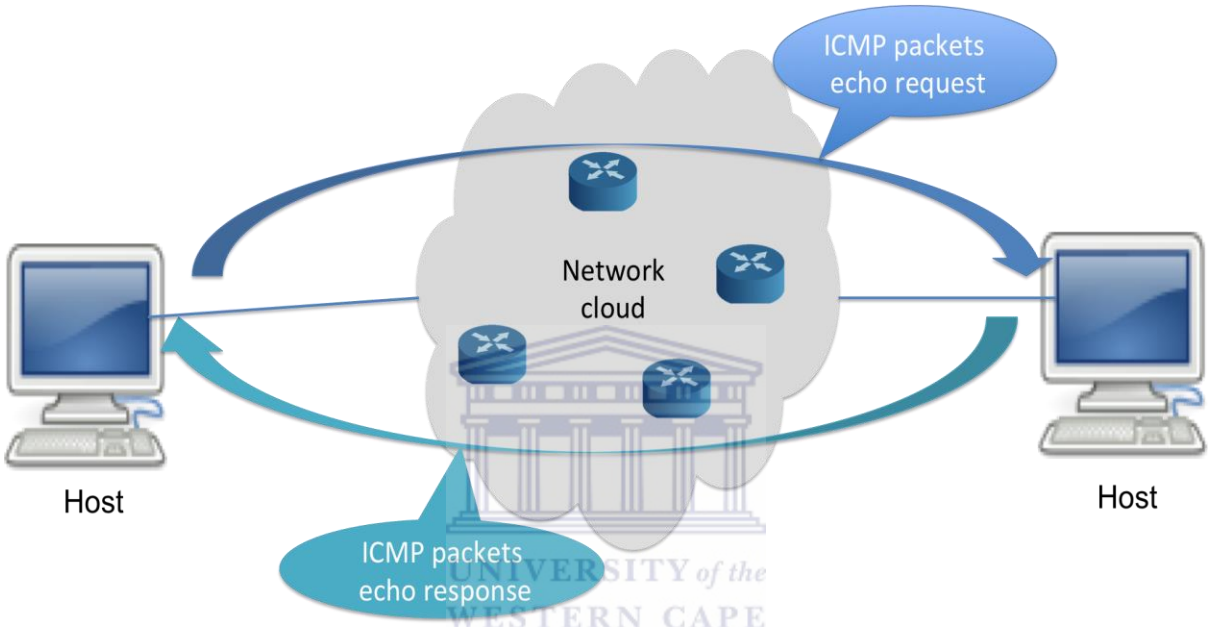


Figure 2.2: Active monitoring with the command Ping

Some of the commonly used performance evaluation tools that apply the active strategy include Brute, Harpoon, Iperf and D-ITG.

According to Nicola et al., it is essential to evaluate the performance of high-speed networks either because of the lack of reliable tools to generate traffic workloads at high rates or because of the inaccessibility of network equipment. For these reasons they implemented a tool called browny and robust traffic engine (Brute), a Linux application allowing high-speed packet generation on personal computers (PC) (Nicola, Giordano, Procissi, & Secchi, 2005).

Similar to Brute, Harpoon (Sommers & Barford, 2004) (Sommers, Kim, & Barford, 2004) is a flow-level traffic generator for router and network tests that focuses on the generation of TCP and UDP packet flows. These packet flows have the same characteristics as routers for the purpose of showing the empirical behaviour of routers under actual conditions. Another significant feature of

Harpoon is its ability to self-configure by automatically extracting parameters from standard net flow logs or packet traces.

Iperf (Iperf, 2008) is a tool to measure the maximum TCP and UDP bandwidth performance and the quality of a network link. It creates TCP and UDP data streams and utilizes the client/server architecture to send a select amount of data from an Iperf client to a listening Iperf server, and measure the time that it takes to transmit/receive the data. Iperf allows the tuning of various parameters and UDP characteristics. Iperf reports bandwidth, delay jitter, datagram loss. Iperf enables an Iperf client to run multiple simultaneous connections to the server as well as both the sending and receiving of data at the same time.

Avallone et al. developed D-ITG that allows generation of transport layer traffic (TCP and UDP) and other types of traffic including VoIP and video. D-ITG (Avallone, Pescape, & Ventre, 2003) has additional functionality such as using different network loads or different network configurations to study scalability problems. It allows the generation of complex and various traffic sources, and offers the option to repeat many times exactly the same traffic pattern (not only its mean value) and get information not only about received packets but also about transmitted packets. With D-ITG, it is also possible to measure the round trip time and one-way delay.

In this review, numerous computer-based traffic generator tools have been mentioned. It is not an exhaustive list, but most of these are PC-based. So far little has been developed for a mobile platform. Recently, Camacho et al. attempted to develop a mobile version of a traffic generator on an Android platform (Camacho, Laner, Svoboda, & Rupp, 2011). The Android mobile traffic generator emulated traffic in three different environments (Android device emulator, Wi-Fi and 3G network) and compared the results between them to see how traffic types behave empirically using each of the connection types. However, the application has a number of limitations: this mobile version does not have the ability of generating important performance statistics such as throughput, delay and jitter thus is not able to evaluate the performance of networks. In addition, the tool cannot simultaneously generate multiple flows. Traffic cannot be sent at the same time to one single device from more than one mobile phone (Various sender devices transmitting to different ports of one receiver device). There cannot be one phone hosting multiple traffic types and multiple phones hosting one traffic type each.

Both passive and active monitoring offer benefits but each has its drawbacks as well. Active techniques create extra traffic on the network, and the traffic or its parameters are synthetic (the traffic is either simulated or emulated). Thus, the main difficulty with the active technique is to

define appropriate models that will accurately characterize realistic traffic flows. In addition, introducing flows into the network can cause interference with the normal traffic on the network. The active approach may cause competition between application traffic and the measurement traffic, reducing the performance of useful applications (Zangrilli & Wekamp, 2003) (Cecil, 2012). However, the volume and other parameters of the introduced traffic are fully adjustable and small traffic volumes are enough to obtain significant measurements (Cottrell, 2001).

Passive techniques capture existing traffic and do not introduce extra traffic in the network. Passive methods simply sniff the network to collect information without disturbing its operation whereas active approaches inject test traffic into the network for the purpose of extracting network characteristics from the test traffic (Huston, 2003).

However, the amount of data captured using passive techniques can be extensive especially when doing flow analysis or trying to capture information on all packets. Furthermore, measurements can only be analysed off-line and not as they are collected, thus processing the huge data sets that have been collected can be a challenging task (Cecil, 2012). The passive approach is limited to periods of time when there is traffic on the network between the hosts of interest and cannot be performed when no existing traffic is available between the sources (Zangrilli & Wekamp, 2003). Yet, the combination of both techniques can be applied in order to provide more useful results as shown by Zangrilli and Wekamp (2003).

2.2 NETWORK TRAFFIC CHARACTERIZATION AND MODELLING

Network traffic is defined as network packets flowing between diverse network traffic sources (Marsic, 2010). Network traffic modelling is useful in a variety of scenarios including the design of network applications, testing the QoS delivered by networks and performance evaluation of communication networks.

The modelling of network traffic primarily involves finding stochastic processes to represent the behaviour of traffic, which can be interpreted as the traffic volume—measured in terms of packets, bytes or bits per time unit, —and its effect on data or packets, which are sent through the network over a specific time unit (Fras, Mohorko, & Cucej, 2012). Typically two different stochastic processes are considered in traffic modelling—one for packet/data sizes and one for packet/data inter-arrival time—to describe network traffic (Fras, Mohorko, & Cucej, 2012). Traffic modelling can represent network traffic by simply describing these two processes in terms of probability distributions (Fras, Mohorko, & Cucej, 2012). However, the heterogeneous and complex nature of traffic running through the network can make it difficult to develop a single and accurate traffic

model that will efficiently capture the traffic characteristics of all types of traffic (e.g.: VoIP, online gaming, video conferencing, and web browsing) in all types of networks (Lee & Fapojuwo, 2005). The choice of a traffic model depends on the type of network and the traffic characteristics of the network under study (Balakrishnan Chandrasekaran, 2009) (Mohammed & Agamy, 2011). In this sense, numerous models have been deployed by researchers such as (Adas, 1997) (Balakrishnan Chandrasekaran, 2009) (Chen, 2007) (Mohammed & Agamy, 2011) to capture either the short-range dependence (SRD) or long-range dependence (LRD) of network traffic. Short-range dependent models include traditional models such as the Poisson process, Markov chains and regression models. Because of the inability of these models to capture the long-range dependence of network traffic, new models such as fractional autoregressive integrated moving average (F-ARIMA) and fractional Brownian motion were developed to capture the LRD. This research focuses on probability distribution and Markov models – in particular the ON-OFF model – for the modelling of UDP, TCP and VoIP traffic. These two aspects (probability distribution and Markov models) are discussed below.

2.2.1 Probability distribution for stochastic processes of network traffic

In network modelling, network traffic is represented in term of two important parameters (Wilson, 2006) (Varga & Olaszi, 2013) (Fras, Mohorko, & Cucej, 2012): packet size and packet inter-arrival time (packet arrival rate). These two parameters are stochastic processes (random variables) and are described by probability distributions. A random variable is fully characterized by its probability distribution function (PDF). Each random variable (packet size or inter-arrival time) has an associated probability distribution, which is a function that describes the relative probability for this random variable to take on a given value. These probabilities distributions are described in Appendix A.

Random variables are classified in two distinct categories (Golder & Golder): a discrete random variable (one that can assume only a finite or countably infinite number of distinct values) or a continuous random variable (one that can assume any non-countable infinite number of values).

The probability distribution of a discrete random variable is called the probability mass function (PMF) whereas that for a continuous random variable, it is called the probability density function (PDF). The PDF or PMF is often noted as $f_X(x)$. The cumulative distribution function (CDF) noted as $F_X(x)$ is another common way to define the probabilities of a random variable. The cumulative distribution function describes the probability that a real-valued random variable X with

a given probability distribution will have a value less than or equal to x. The PDF $f_X(x)$ and the CDF $F_X(x)$ are defined as follows:

$$f_X(x) = P(X = x) \quad (1)$$

$$F_X(x) = P(X \leq x) \quad (2)$$

The PDF and CDF are related as follow:

$$F_X(x) = \int_{-\bowtie}^{x} f_X(t)\, dt \quad (3)$$

The main task for modelling the stochastic process with a probability distribution is to choose the right distribution, one which could be a good representation of the network traffic stochastic process (Fras, Mohorko, & Cucej, 2012). Traffic models often assume that packets arrive as a Poisson process i.e. the packet inter-arrival time follows an exponential distribution (Marsic, 2010). Still, many other distributions such as Pareto, Gaussian, Constant, Gamma, etc. are considered to model both packet size and packet inter-arrival time. The distributions chosen should be able to capture the relevant and representative characteristics of the emulated traffic. The representation of the PDF and CDF of each distribution used in the MTGawn system is shown in Appendix A.

2.2.2 Markov process

Markov models use a finite number of states to model the activities of a traffic source (Adas, 1997) (Mohammed & Agamy, 2011) (Balakrishnan Chandrasekaran, 2009). Markov models can be classified in two categories: Markov chain or a Hidden Markov model. The main difference between these models is that the state is fully observable in the Markov chains, which is not the case in the Hidden Markov model. The Markov chain models the state of a system with a random variable that changes through time. The Markov chain is fully characterized by the Markov property, which states that the distribution for the current sate depends only on the distribution of the previous state i.e. if X_n represents a random variable that defines the state at time n then the set of random variables $\{X_n\}$ will form a discrete Markov chain where the probability of the next value $X_{n+1} = S_j$ depends only on the current state X_n where $S = S_1 + S_2 + \cdots + S_m$ is a given state space (Adas, 1997). As the number of states used in the model increases, the accuracy of the model increases linearly but the complexity of the model increases proportionally as well (Adas, 1997).

The widely used Markov model includes ON-OFF. The ON-OFF model uses two states: ON state and OFF state. The transition time which is the time between the ON and OFF states follow an exponential distribution. The ON-OFF source model is widely used for voice traffic modelling. The ON state represents the speech time (talk spurt) and the OFF state represents the silence time after the talk. During the ON period T_{on}, the voice packets are continuously generated and transmitted at a rate λ_{on}. No packets are transmitted during OFF period T_{off}. The durations of ON and OFF periods follow an exponential distribution with their means equalled $T_{on} = \frac{1}{\alpha_1}$ and $T_{off} = \frac{1}{\alpha_2}$ respectively. The ON-OFF model is defined as a Markovian process with three parameters as shown in Figure 2.3: α_1 (the transition probability rate from ON to OFF state), α_2 (transition probability rate from OFF to ON state) and λ_{on}. During the ON state, the size of the packets generated as well as the time interval between two consecutives packets is related to the codec in use (Hassan, Garcia, & Brun, 2005) (Deng, 1995) (Sriram & Whitt, 1986).

Figure 2.3: ON-OFF model for voice traffic

To conserve bandwidth, a process called voice activity detection (VAD) can be used to detect the absence of audio and prevent the transmission of "silent packets" (TechTarget, 2008). The capability to stop sending real time protocol (RTP) or compressed real time protocol (cRTP) packets during the OFF state is also referred to as "Silence Suppression" (Bitweaver, 2003). Applications use VAD to monitor signals for voice activity and to detect silent periods. As VAD can allow saving up to 35 percent of bandwidth (Cisco , 2006), the application informs the voice protocol and prevents the encoder output from being transported across the network (TechTarget, 2008).

In this research, the emulation of the voice traffic was done based on three different codecs: G.728, G.729, G.723. The codec determines the rate at which the voice is sampled and the size of each packet transmitted. Voice traffic was generated according to the parameterization defined by each codec. Table 2.1 depicts how the voice was sampled for each codec.

Table 2.1: Parameters for voice generation for each codec (Newport Networks , 2005)

Codecs	**G.711**	**G.723**	**G.729**
Sample period (ms)	20	30	20
Frame size (payload)	160	20/24	20
Rate (Packets/s)	50	33	50
Bandwidth (Kbps)	64	5.3/6.4	8

2.3 GENERATION OF EMULATED TRAFFIC ON ANDROID DEVICES

The MTGawn system was only deployed for mobile devices that run an Android operating system (OS). The programming language for an Android application is Java. In Java programming, TCP packets are transmitted and received using Sockets and UDP packets are created using the DatagramPacket and sent using a DatagramSocket. A network packet is divided into two parts: a header and a payload (see Figure 2.4).

Figure 2.4: Structure of a network packet

A TCP packet has a minimum header of 20 bytes and a maximum of 60 bytes. TCP uses a three-way handshake to establish connection, and thus provides a reliable, ordered and error-checked delivery of data. The payload size of a TCP Packet is specified by the maximum segment size (MSS), which is the largest amount of data that can be sent or received in a single segment. The MSS should be small enough to avoid IP fragmentation though large enough to avoid overhead (Husanu & Trifan, 2014). Fragmentation can cause packet loss and excessive retransmissions, which degrade the performance of applications.

To send TCP data, a Socket is opened with the *IP address* and *port* number of the host where the data will be sent (see Code snippet 2-1). A *DataOutputStream* can be used to put the data in the buffer and transmit it calling *write ()*.

```
Socket socket = new Socket (IP address, port)
DataOutputStream dataOutput;
dataOutput = new DataOutputStream(socket.getOutputStream());
dataOutput.write(data);
```

Code snippet 2-1: Sending TCP packet via Socket

The UDP packet header is 8 bytes. VoIP encapsulates RTP or cRTP packets in an UDP packet thus adding an extra 12 bytes header to the packet size, while cRTP adds only a 2 to 4 bytes header as shown in Figure 2.5.

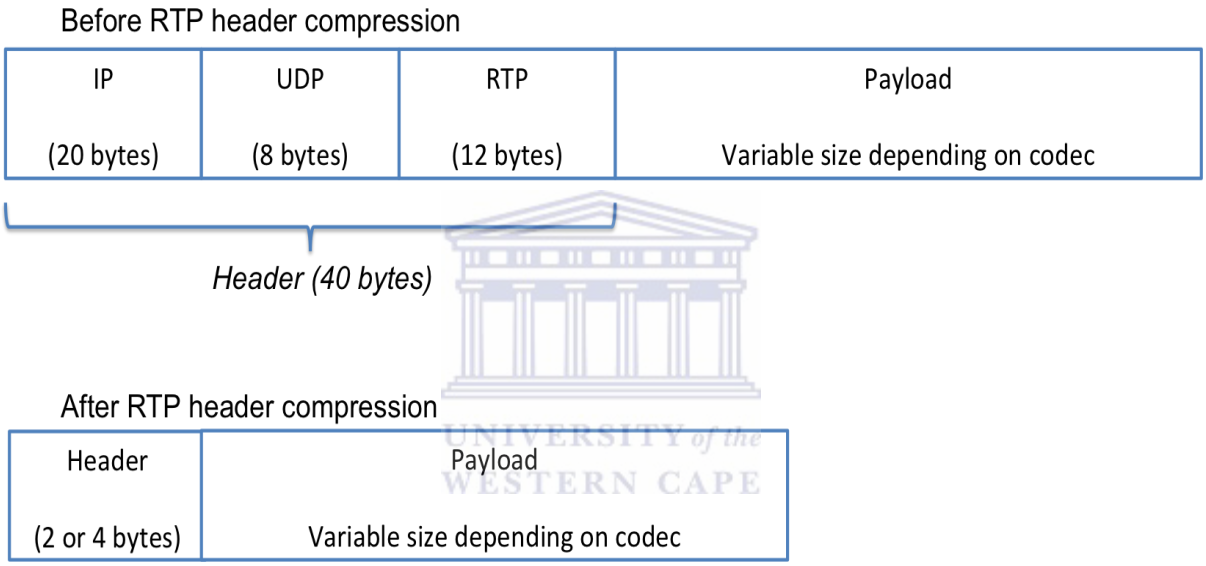


Figure 2.5: Structure of RTP and cRTP packet

Adapted from (Cisco , 2006)

In Java, the max size of a UDP packet (excluding the header) that can be sent or received is 65507 bytes under IPv4. Attempting to send or receive a packet with a buffer that exceeds that limit results in IOException (Friesen, 2013). The *buffer* (the byte array) contains the data to be sent in the UDP datagram. The *length* is the packet data length in the buffer. The *port* indicates the specified port number on the specified host *IP address* where the data should be sent (see Code snippet 2-2).

```
DatagramSocket datagramSocket = new DatagramSocket ()
byte[] buffer = new byte[65507];
DatagramPacket packet = new DatagramPacket (buffer, length, IP address, port);
datagramSocket.send(packet);
```

Code snippet 2-2: Sending UDP packet via DatagramSocket

To receive a packet using UDP in Java, a DatagramSocket is instantiated with the UDP port number that will be used to receive UDP packets (see Code snippet 2-3). The DatagramPacket has no information about the node to send data to, as it does when creating a DatagramPacket for sending data. For receiving data, no destination address is needed. The *receive ()* method blocks until a DatagramPacket is received. The data received is located in the buffer and can be accessed by calling *getData ()*.

```
DatagramSocket datagramSocket = new DatagramSocket (port)
byte [] buffer = new byte [65507];
DatagramPacket packet = new DatagramPacket (buffer, length);
datagramSocket.receive (packet);
byte [] buffer = packet.getData ();
```

Code snippet 2-3: Receiving UDP packet via DatagramSocket

With TCP, a ServerSocket instance is created and listens to incoming connections at the port number *port*. If a connection is established, an *accep*t () method will return a socket representing the opened connection. The data can be received through the Socket using a *DataInputStream* and calling *read ()* (see Code snippet 2-4).

```
ServerSocket serverSocket = new ServerSocket (port)
Socket socket = serverSocket.accept ();
DataInputStream dataInput = new DataInputStream(socket.getInputStream());
dataInput.read (data);
```


Code snippet 2-4: Receiving TCP packet via Socket

2.4 NETWORK PERFORMANCE METRICS

Performance metrics refer to the criterion used to quantify the performance of a system and determine the QoS level of applications. Performance metrics are obtained by processing the data obtained through passive or active methods. A traffic measurement and analysis is done in order to extract performance metrics of interest. Different applications (data/voice/multimedia) have different requirements for network service in order to be useful to users—some applications are sensitive to packet loss while others are impaired by delay or jitter (Marsic, 2013). The common network measurement metrics includes availability, packet loss, delay and throughput (Hasib, 2006). These factors affect the QoS of traditional data services (traditional TCP and UDP based applications), real-time and multimedia systems (VoIP).

2.4.1 Throughput

In the context of communication networks, network throughput refers to the rate of successful message delivery over a communication channel. When considering the performance of a network, throughput is measured either in terms of how many units of information the system can process in a given amount of time (TechTarget, 2000) or by calculating the amount of data transferred between locations during a specified period (Techopedia, 2010). In data transmission, network throughput is the amount of data moved successfully from one place to another in a given time period. It is generally measured in bits per second (bps), kilobits per second (Kbps), megabits per second (Mbps) or gigabits per second (Gbps) (TechTarget, 2000). Network throughput is affected by many factors including bandwidth availability, packet loss, latency and jitter, all of which degrade network throughput and make a link perform like one with lower bandwidth (TechTarget, 2009). In a network communication, other phenomena such as specifications of hosts computer in the network, processing overhead in the software, degree of parallelism both hardware and software support and types of transactions being processed have an impact on the throughput (Loadfocus, 2015).

2.4.2 Packet loss

In a communication network, when packet arrives at a full queue (aka buffer), the packet is dropped (aka lost). A lost packet may be retransmitted by the previous node, by the source end system, or not retransmitted at all. There are many other factors that can cause packet loss in the network. The most important factors are (Hasib, 2006) (Cai, Wolf, & Gong, 2011):

i. Bit errors caused by data link failure, all the bits currently in transit on that link will be lost.

ii. Congestion, which happens when the network routers/switches are sent more packets than they can process and their buffers can accommodate.

Applications using TCP (such as hypertext transfer protocol (HTTP), file transfer protocol (FTP), simple mail transfer protocol (SMTP)…etc.) (Hasib, 2006) are more sensitive to packet loss than delay since TCP is known to be reliable as it offers a handshaking mechanism to avoid loss and guarantee delivery and ordering of packets at their arrival. Time-sensitive applications (such as VoIP, simple network management protocol (SNMP), the network time protocol (NTP) and other emerging multimedia applications such as video conferencing, teleconferencing…etc.) (Hasib, 2006) often use UDP because dropping packets is preferable to waiting for delayed packets, which may not be an option in a real-time system.

2.4.3 Delay

As a packet travels from one node (host or router) to the subsequent node (host or router) along its path, the packet suffers from several different types of delays at each node along the path. The most important of these delays are the nodal processing delay, the queuing delay, the transmission delay, and the propagation delay. Together, these delays accumulate to give a total nodal delay.

VoIP is one of the main applications that are really affected by packet delay. The main issue of VoIP is a great potential for degraded voice quality due to packet latency when the underlying network links experience heavy traffic loads. When a VoIP media server is streaming voice traffic, audio data is periodically processed and sent out over the network. The user then receives packets at regular intervals. During the process, delayed packets may need to be dropped so as not to disrupt real-time playback. Packets that contain voice data can be lost for several reasons including: insufficient bandwidth due to poor capacity planning; packets arriving at their destination too late; and network outages. Since voice quality is very sensitive to packet loss (even 1% packet loss can jeopardize voice quality), such packet drops caused by delayed packet delivery, can result in degraded voice quality. Therefore, timely processing and delivery of audio data is critically important to VoIP services (Heo, 2011). To ensure an intelligible audio reception, voice-carrying packets should not be dropped excessively, delayed or be subjected to highly variable delays (Barbosa, Kamienski, Mariz, Callado, Fernandes, & Sadok, 2007). Delay can have a considerable impact on conversational quality. Delay leads to conversational interaction problems. For example, it can lead to call participants interrupting each other (doubletalk) or to excessive pause in speech. Frequent interruptions can obviously be quite annoying and excessive periods of silence can be misinterpreted as delays in response, which can alter the apparent emotional content of speech (Telchemy, 2006).

2.4.4 Jitter

The variation in delay is called jitter, which also causes damage to voice quality. If the delay variation is too high, packets arrive too late to be of any use, and are discarded. A jitter buffer helps reduce the impact of this effect by buffering the packets for a short while before playing them back. The jitter buffer will also fix any out-of-order errors by looking at the sequence number in the RTP frames. This has the effect of smoothing the packet flow, increasing the resiliency of the codec to packet loss, delayed packets and other transmission effects. Even though this technique effectively reduces packet loss, the downside of the jitter buffer is that it can add a higher playback delay (Alvarion, 2006) (Aktas, Schmidt, Weingärtner, Schnelke, & Wehrle, 2012). Monitoring jitter

and packet loss in a network can be achieved by monitoring the media stream by using Real-time Transport Control Protocol (RTCP) (Mundra & Hernandez, 2004).

2.5 SUMMARY

An accurate estimation of network performance is essential for the success of any network. Two approaches namely passive and active can be used to perform this task (Cecil, 2012) (Huston, 2003) (Zangrilli & Wekamp, 2003). Traditionally, most businesses have employed passive methods to monitor application performance within their network. Typically, these systems were hardware-based tools that were placed inside the network with the goal of monitoring real user behaviour and capture traffic flowing through the network. Even though these tools were proven to be advantageous, they had several disadvantages including the large amount of data that needed to be processed off-line, after the data collection. Over the years, active monitoring has increasingly become the focus of attention. The concern of the active approach is the choice of appropriate traffic models to emulate real life traffic flows thus researchers have proposed several traffic models and the choice of the suitable model depends upon many parameters such as the type of traffic to be generated (UDP, TCP, VoIP…. etc.).

The collected data or injected test traffic is analysed in order to estimate the performance of the network in terms of the relevant metrics such as throughput, packet loss, delay and jitter. The performance monitoring tools presented in the literature are mostly PC based. This research employed the active approach to evaluate the performance of wireless communication networks by developing a mobile traffic generator tool named MTGawn. The next chapter describes the approach and methodology used in this research for the design of the proposed MTGawn tool.

Chapter 3

RESEARCH METHODOLOGY AND DESIGN

In the previous chapter, the key concepts related to traffic generator and network performance estimation was explained. The contribution of many researchers in the domain of network performance evaluation was investigated and some common network analysis tools were reviewed. This chapter deals with the research approach used to investigate the research question posed which is "How should a mobile traffic generator be designed to be feasible in order to analyse and evaluate the performance of wireless networks?"

3.1 RESEARCH APPROACH

According to Crotty (1998), epistemology, theoretical perspective, methodology and methods are four basic building blocks that frame the research process and give rise to the following questions:

i. What methods are to be used?

ii. What methodology guides the choice and use of the proposed methods?

iii. What theoretical perspective underpins the preferred methodology, and

iv. What epistemology informs the suggested theoretical perspective?

Firstly, a researcher has to adopt an epistemology or a stance towards the nature of knowledge such as objectivism or subjectivism. This viewpoint guides the entire research process as well as informs the chosen theoretical perspective the researcher assumes, for example post-positivism or interpretivism. Then, the theoretical perspective implied by the research questions, dictates the researcher's choice of a methodology. Finally, the methodology or plan of action stimulates the choice of methods that will be used during the research to collect and process the data. These four basic elements of research represent distinct hierarchical levels, which inform one another as depicted in Figure 3.1.

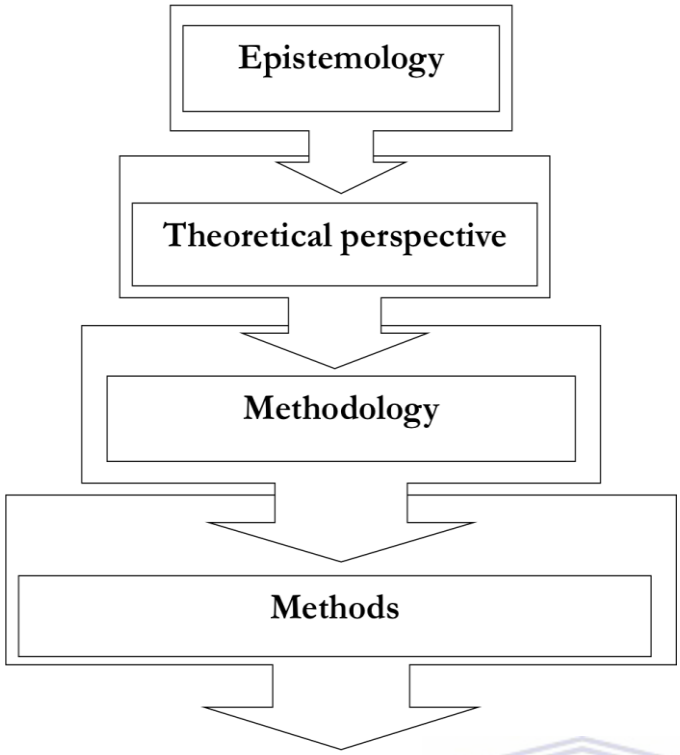


Figure 3.1: The four basic elements of research
Adapted from (Crotty M. , 1998)

3.1.1 Epistemology

In the research literature, a frequent term used when referring to the philosophy lying behind a research approach is ontology. Ontology is the study of being and deals with ways of constructing reality and describes our view (whether claims or assumptions) on the nature of reality (Flowers, 2009). When considering that different views exist regarding what constitutes reality, another question must be how is that reality measured, and what constitutes knowledge of that reality. This leads us to questions of Epistemology (Flowers, 2009). Epistemology refers to the theory of knowledge and a view of how it is acquired. It is usually coupled alongside ontology informing the theoretical perspective. Ontology is more concerned with 'what is', with the nature of existence, with the structure of reality as such, while epistemology deals with 'the nature of knowledge, its possibility, scope and general basis' (Crotty M. , 1998). Ontology and epistemology are conceptually related and usually merged together.

Crotty suggests three epistemological stances: Objectivism, Subjectivism, Constructivism and their variants. The first stance, objectivism maintains that knowledge exists whether we are conscious of it or not (the existence of knowledge is independent of our consciousness of it) (Crotty M. , 1998). Researchers with this stance seek for explanations by developing and testing hypotheses and theories. The second stance, constructivism is in contradiction with the objectivism stance as it

sustains that there is no objective truth waiting to be discovered by us. Knowledge (i.e. truth or meaning) comes into existence in and out of our engagement with the realities in our world. Meaning is not discovered, but constructed and inflicted by our mind (there is no meaning without a mind) — different people may construct meaning in different ways, even in relation to the same phenomenon (Crotty M. , 1998). The third stance, subjectivism maintains that knowledge is generated from the mind, without reference to reality. While constructivism acknowledge the influence of the reality in the generation of meaning, subjectivism is the denial of reality and holds that gaining knowledge about the world is done through introspection (Landauer & Rowlands, 2001) (Crotty M. , 1998).

Each epistemological stance implies a profound difference in how the research is approached and how the research outcome is presented (Crotty M. , 1998). Thus, a qualitative or quantitative research approach differs by contrasting epistemological and ontological aspects and how these beliefs and views fit with their different intellectual goals (Bahari, 2010). Usually, researchers associate objectivism with a quantitative approach while qualitative research approach is commonly aligned with constructivism or subjectivism (Bahari, 2010) (Crotty M. , 1998). This research takes on an objectivist stance and quantitative research methods are thus appropriate.

3.1.2 Theoretical perspective

A theoretical perspective is termed as a research paradigm or research philosophy and refers to the philosophical assumptions related to the underlying epistemology that guides the research. Herbert and Irene (Rubin & Rubin, 2012) suggest two research paradigms, which differ in terms of the goals of the research and the ways used to achieve these goals: the paradigms are positivism and naturalism. They propose a list of questions that will help differentiate between the two stances (Rubin & Rubin, 2012):

- **i.** Is the purpose to test theories and discover general principles, or is it to describe and explain complex situations?
- **ii.** Should the work be primarily deductive; that is, should it start out with broad theories and suppositions and then systematically test their implications?
- **iii.** Or should it be inductive; that is, should it build explanations from the ground up, based on what is discovered?
- **iv.** Is there one truth out there that the researcher is trying to measure, or are there many possibly contradictory ones?

Positivists claim the existence of a single and objective reality or truth that can be observed and directly measured. Naturalists argue that reality is indirectly measureable i.e. reality can be changed and perceived differently through the interpretations of people. The purpose of a positivist is to work out theories and prove or disprove their hypotheses. Naturalists describe and explain a complex situation or process. Their purpose is more to explain and understand what has happened in a specific circumstance, than to prove or disprove a hypothesis.

Researchers that underwrite a positivist approach have an underlying objectivism epistemological stance (Crotty M. , 1998) and use quantitative methods that emphasize measuring and counting (Rubin & Rubin, 2012). This research identifies with the positivism perspective, as it is more concerned with an objectivist stance and a quantitative approach.

3.1.3 Methodology

The procedures by which researchers go about their work of describing, explaining and predicting phenomena are called their research methodology (Rajasekar, Philominathan, & Chinnathambi, 2006). Rajasekar et al. defined methodology as the study of methods by which knowledge is gained and which is a systematic way to solve or address a research problem. A research methodology is concerned with the following (Rajasekar, Philominathan, & Chinnathambi, 2006):

i. Why is a particular research study undertaken?

ii. How did one formulate a research problem?

iii. What types of data were collected?

iv. What particular method has been used?

v. Why was a particular technique of analysis of data used?

Thus the methodology refers to a strategy of inquiry or a work plan that moves from the underlying philosophical assumptions to the research design and data collection. Throughout the research process, the selection and use of particular methods to collect and analyse data is influenced by the choice of a specific research methodology.

An appropriate methodology suitable for addressing the research problem of this thesis is design science research (DSR). Brocke & Buddendick (2006) explains that DSR is an iterative process where each cycle consists of six phases: identify, build, document, select, evaluate, and communicate (see Figure 3.2).

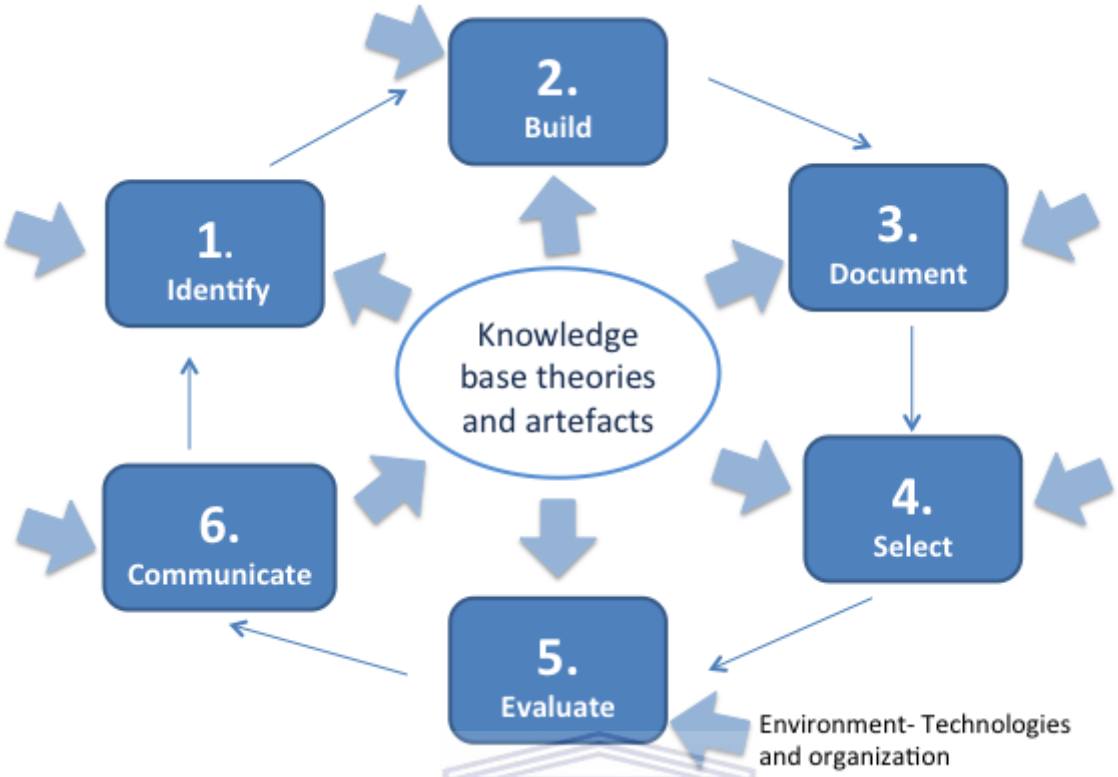


Figure 3.2: The basic form of DSR cycle

Adapted from Brocke and Buddendick (2006)

Peffers et al. (2008) who also describe the DSR methodology suggest six DSR activities namely: problem identification and motivation; define the objectives of a solution; design and development; demonstration; evaluation and communication.

Table 3.1: DSR activities according to Brocke and Buddendick (2006) and Peffers et al. (2008)

Brocke and Buddendick (2006)	**Peffers et al. (2008)**
1. Identify	i. Problem identification & motivation ii. Define objectives of a solution
2. Build 3. Document	iii. Design and development
4. Select 5. Evaluate	iv. Demonstration v. Evaluation
6. Communication	vi. Communication

The two models are quite similar and interrelated as illustrated in Table 3.1: the first phase "Identify" is broken into two activities "problem identification & motivation" and "define objectives of a solution". The two phases "build" and "document" are combined in one activity "design and development". The "select" and "evaluate" phases can be associated with the "demonstration" and "evaluation" activities. Finally the "communication" part remains the same in both cases.

During their study of the DSR as a methodology, Brocke & Buddendick (2006) and Peffers et al. (2008) both clearly elaborate key aspects that need to be considered in each phase while applying the DSR in research (see Table 3.1). Their concepts can be summarised as follows:

- Identify (Problem identification & motivation and Define objectives of a solution): In this phase, the following questions should be answered: What is the problem? How should the problem be solved? What are the specific criteria that a solution for the problem should meet? The researcher clearly needs to identify the research problem and needs to justify the value of a solution. This phase also involves defining the objectives of the research and the derivation of requirements. The focus is the understanding of the problem's relevance, current solutions and their weaknesses.
- Build and document (Design and development): In this phase, the researcher should develop an artefact capable of solving the problem and delivering functionalities, then

represent and document the artefact. This is achieved by building constructs, models, methods, or instantiations in which a research contribution is embedded and the application of methods, technologies and theories to create an artefact that delivers utility.

- Select and evaluate (Demonstration and Evaluation): After the design of a solution, the use of the artefact to solve one or more instances of the problem should be illustrated. Before the evaluation, the evaluation criteria and techniques should be known and chosen. The principal aim of these phases is to determine how well the artefact works. The system is tested and analysed according to the selected evaluation metrics.
- Communicate: In this phase, the solution to the problem and its novelty and effectiveness can be compared to other work and communicated to relevant audiences. The cycle can be iterated and the results obtained can serve as new requirements to solve other instances of the problem.

3.1.4 Methods

The techniques or procedures used to gather and analyse data related to some research question are referred to as research methods. This research adopted an active method to evaluate the performance of networks on a mobile device by developing a traffic generator MTGawn to generate emulated traffic flows over a wireless interface of a mobile device. In the context of this study, an experimental design and quantitative analysis of the results was implemented (see Figure 3.3).

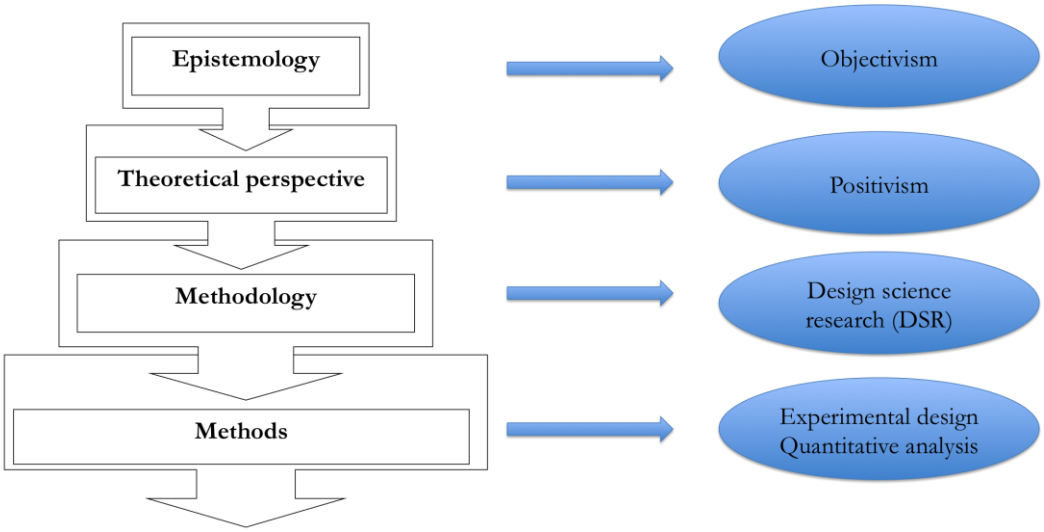


Figure 3.3: Research framework

Adapted from (Crotty M. , 1998)

3.2 RESEARCH DESIGN

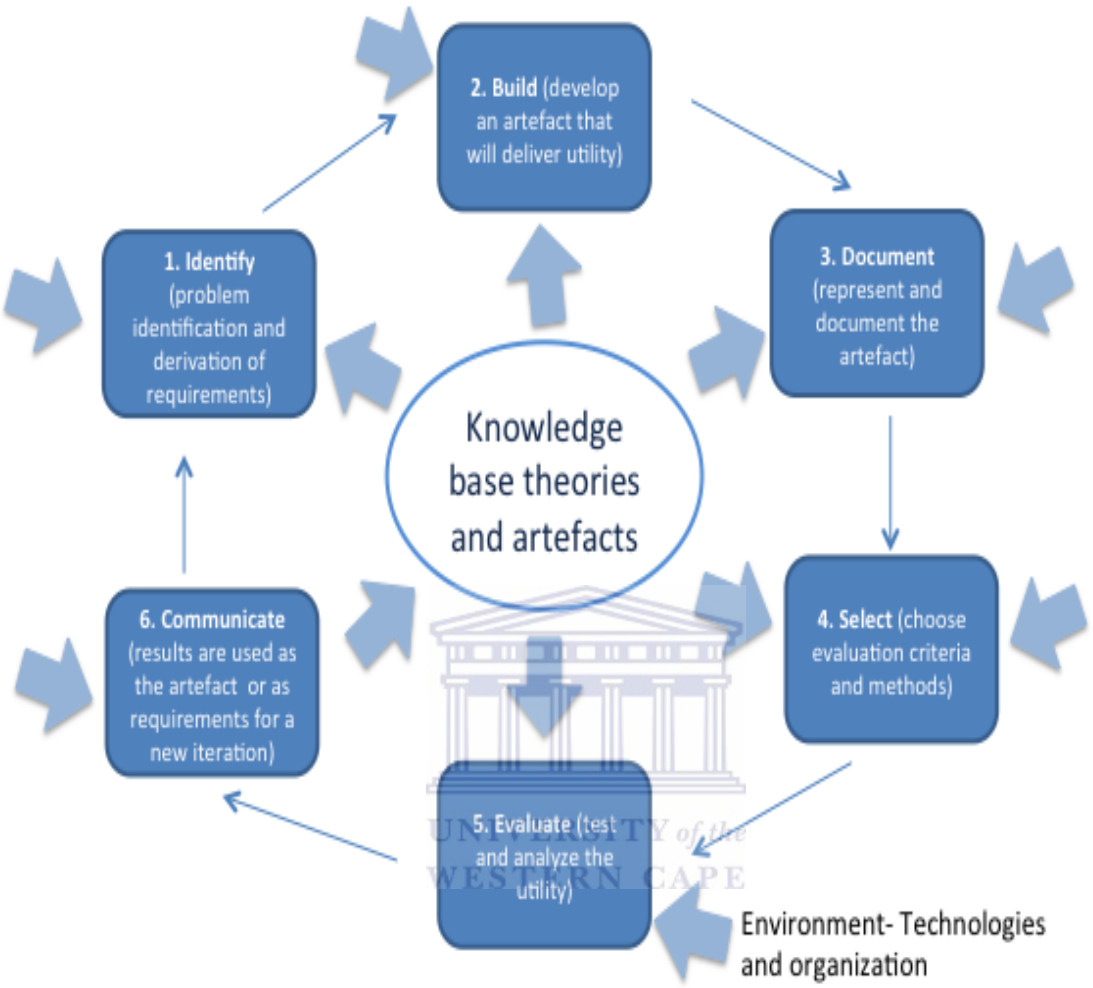


Figure 3.4: DSR activities

This work applied an active monitoring approach to generate network traffic with realistic characteristics for the purpose of network performance analysis. Design science research (DSR) was used to design the traffic generator tool, MTGawn. The research process was executed in three phases namely (see Figure 3.5):

- Phase 1: Problem identification (Identify)
- Phase 2: Prototype design and presentation (Build & Document)
- Phase 3: Prototype evaluation and report (Select, Evaluate & Communicate)

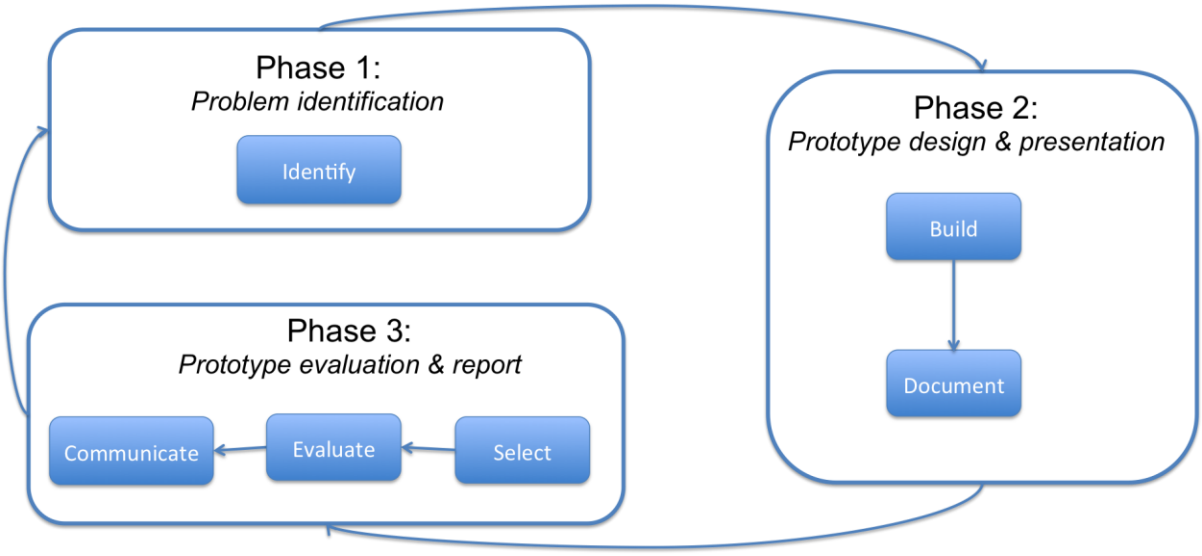


Figure 3.5: Different phases of the research design associated with DSRM

3.2.1 Phase 1: Problem identification

This activity entailed identifying the problem and defining the objectives of a solution. A research problem is referred to as the focus or reason for engaging in the research. In this thesis, methods such as a literature survey and document analysis were applied to define and delineate the research and clarify the research problem: To design a mobile traffic generator to evaluate the performance of wireless networks in remote areas. The research question is thus: "How should a mobile traffic generator be designed to be feasible in order to analyse and evaluate the performance of wireless networks?". This main research question was framed as four sub-questions in order to easily analyse the problem:

a. What techniques should be used for network monitoring and performance evaluation?

b. How should an appropriate traffic model be developed to emulate realistic traffic flows for mobile wireless networks?

c. How should the emulated traffic, over the Wi-Fi interface of a mobile device, be generated?

d. What performance metrics should be taken into consideration?

The outputs of this phase clarified the research problem and enabled the formulation of a research question to guide the research design.

3.2.2 Phase 2: Prototype design and presentation

The next phase of the research process involved addressing the research question by developing a prototype. This phase entailed the design and the representation of a prototype capable of solving

the problem and delivering functionalities. In this research, the prototype design included both the design of the user interface and the design of the architecture for the proposed MTGawn tool.

Designing a traffic generation tool involved creating a stochastic model to appropriately represent the relevant features of the type of traffic generated and injected (transmission then capturing) into the network. The traffic was modelled as a dynamic stochastic (i.e. random) process using probability distributions. The generation of traffic involved the communication between two mobile Android devices, one acting as a sender and the other as a receiver.

Different traffic flows can be present in a network at the same time, because users often browse the web, read emails, send text messages, play games, make voice calls and stream videos simultaneously. In such a case data traffic is the result of parallel user activities (Varga & Olaszi, 2013). Network traffic is varied and complex. In this research, the focus was on a limited range of traffic flows UDP, TCP and VoIP as well as a mixture of these flows. This was done in order to represent real-life traffic. DSRM was applied to design a tool that allowed the sending and receiving of single traffic flows as well as of multiple traffic flows. Two incremental prototypes were developed: The first prototype (**Prototype A)** was designed to allow the generation and transmission of single UDP, TCP and VoIP flows; this prototype was improved by adding the ability to generate and transmit multiple traffic flows simultaneously **(Prototype B) see** Figure 3.6.

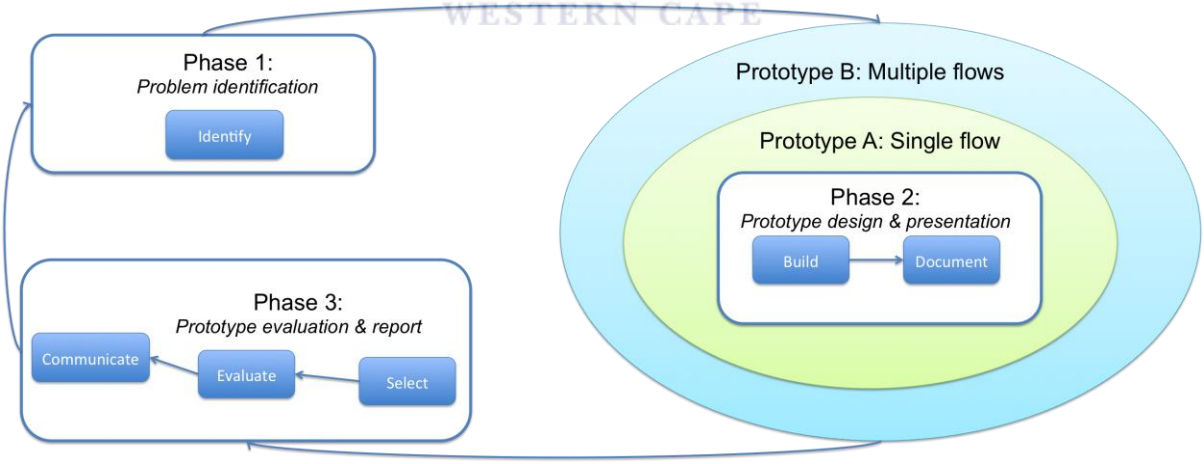


Figure 3.6: Incremental prototype design approach

3.2.2.1 General architecture of a MTGawn prototype

In this implementation, the system constitutes four modules with different roles: modelling; sender; receiver and analyser. Each component plays a specific role, but all the components communicate simultaneously to provide the underlying features. The system structure follows a linear form, i.e. the flow is first created (Modelling) then sent over the network (Sender) and

received at the other endpoint of the network (Receiver), and lastly analysed (Analyser). Figure 3.7 gives an overview of the MTGawn architecture. The 'request manager' is integrated into both the sender and the receiver module and its role is to manage parallel incoming and outgoing traffic (which can involve single or multiple senders or receivers). The request manager's purpose is to allow multiple flows to be sent and received simultaneously. The user interface of all components is discussed in Appendix B.

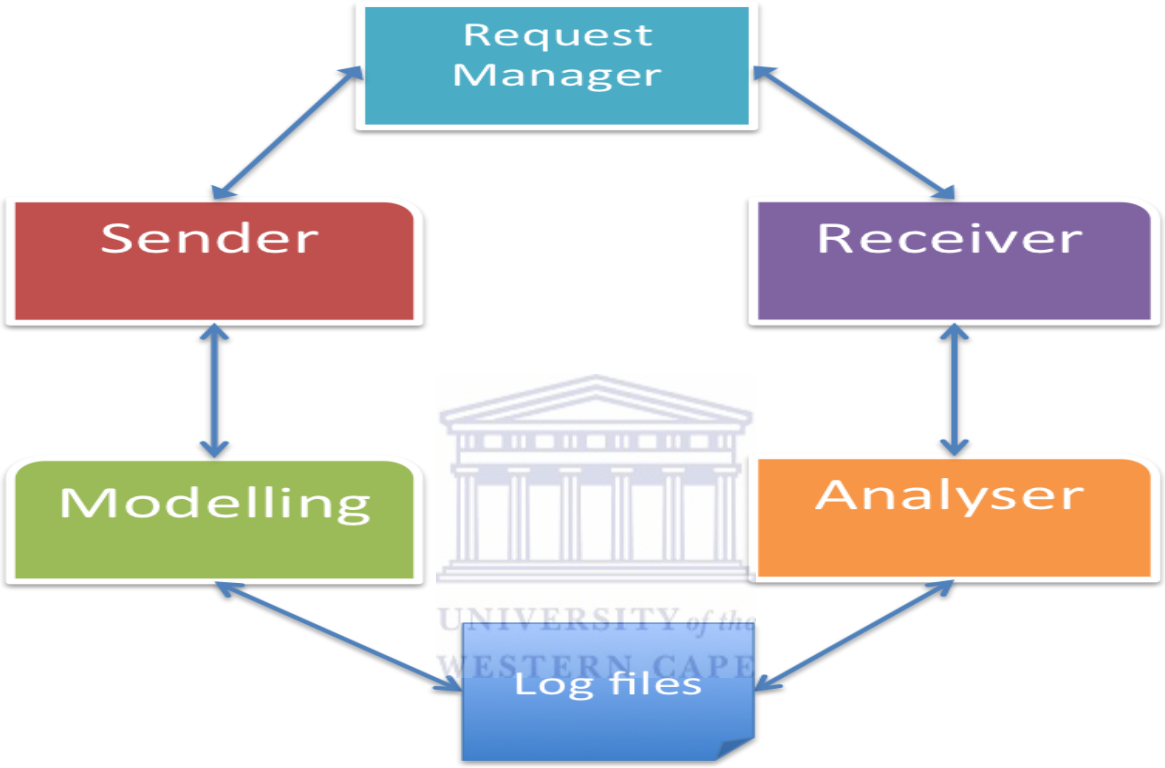


Figure 3.7: Architecture design of a MTGawn prototype

Modelling

The 'modelling' module is in charge of creating a stochastic model to appropriately represent the relevant features of the type of traffic being studied. In the performance evaluation of wireless communication networks, modelling traffic flows is an important task because the behaviour of the developed model should mimic the behaviour of the realistic flow (Andreev, Anisimov, Koucheryavy, & Turlikov, 2010; Avallone, Guadagno, Emma, Pescape, & Ventre, 2004). To create a successful model it is essential to classify a particular network's activities, because different user activities produce different traffic patterns and each traffic pattern can be characterized by various parameters. Custom (UDP and TCP) and Voice (VoIP) were modelled in this thesis.

To model traffic, what was taken into consideration was that traffic flows that circulate between a sender and a receiver in a communication network are characterized by two significant parameters:

the size of each transmitted packet and the elapsed time between packet transmissions (the rate at which packets are transmitted over the network). Packets circulating in the network have different sizes and the time between two consecutive packets is not always constant. For the sender, each time period represents the elapsed time between the transmission of the current packet and the transmission of the next packet, while for the receiver each time period represents the elapsed time between the reception of the previous packet and the reception of the current packet (as shown in Figure 3.8).

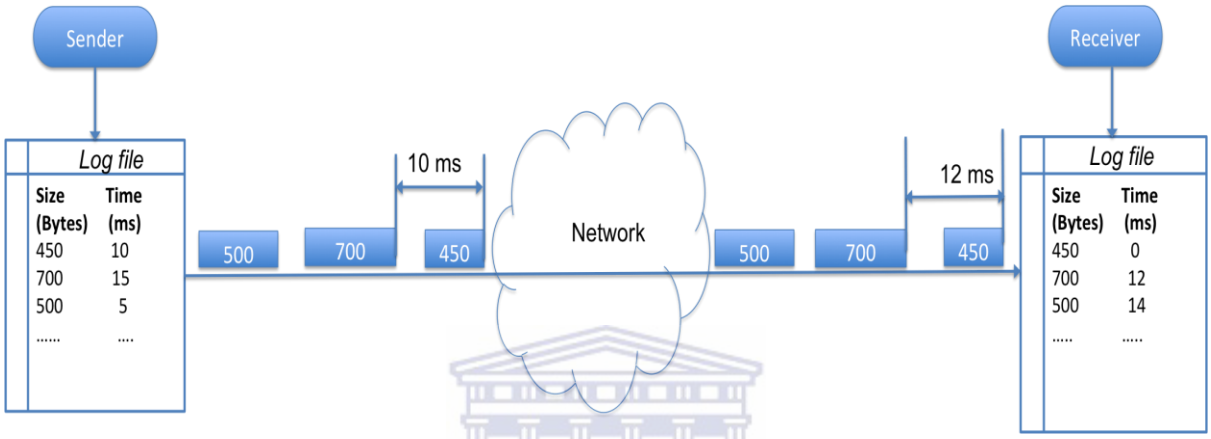


Figure 3.8: Network traffic flowing between a sender and receiver

For the flow modelling, the following information needs to be provided (see Figure 3.9):

- ✓ **Description:** Words that describe the flow.
- ✓ **Destination port:** The port used at the destination to receive the flow.
- ✓ **Destination IP:** The IP address of the receiver.
- ✓ **Type of duration:** define the duration of the generation. It is either "per packet" or "per millisecond".
- ✓ **Number of packet or duration:** specify how many packets the flow contains or how long the generation will last.
- ✓ **Type of flow:** three types of flows: Voice or Custom
- ✓ **Protocol:** it is either UDP or TCP. In the case of voice, the TCP option will not be enabled.
- ✓ **Delay:** specify whether the sending process will be unidirectional i.e. one way delay (OWD) or bidirectional i.e. round trip time (RTT)

- ✓ **Size distribution:** In case of the "Custom" flow, the size of each packet in bytes typically follows a statistical distribution (see Appendix A).
- ✓ **Time distribution:** In the case of the "Custom" flow, packet inter-arrival (represents the time between two consecutive packets) typically follows a statistical distribution (see Appendix A).
- ✓ **Voice protocol:** In case of the "Voice" flow, the protocol used to carry the voice RTP or cRTP.
- ✓ **Voice codec:** The type of codec used to convert the voice (G.711, G.723, G.729).
- ✓ **VAD (Voice activity detection):** Either "yes" or "no". Determine whether or not silence suppression should be taken into consideration.

Figure 3.9: MTGawn Modelling interface

Sender

The 'sender' module is the core function responsible for sending the flows over the network. To do so, it uses the type of transport protocol appropriate for the specific type of flow (UDP or TCP). In the case of voice traffic, either RTP or cRTP packets are created and encapsulated in the UDP packet to carry the voice packet. Each packet sent is saved in a sender log file. Figure 3.10 illustrates the sender interface.

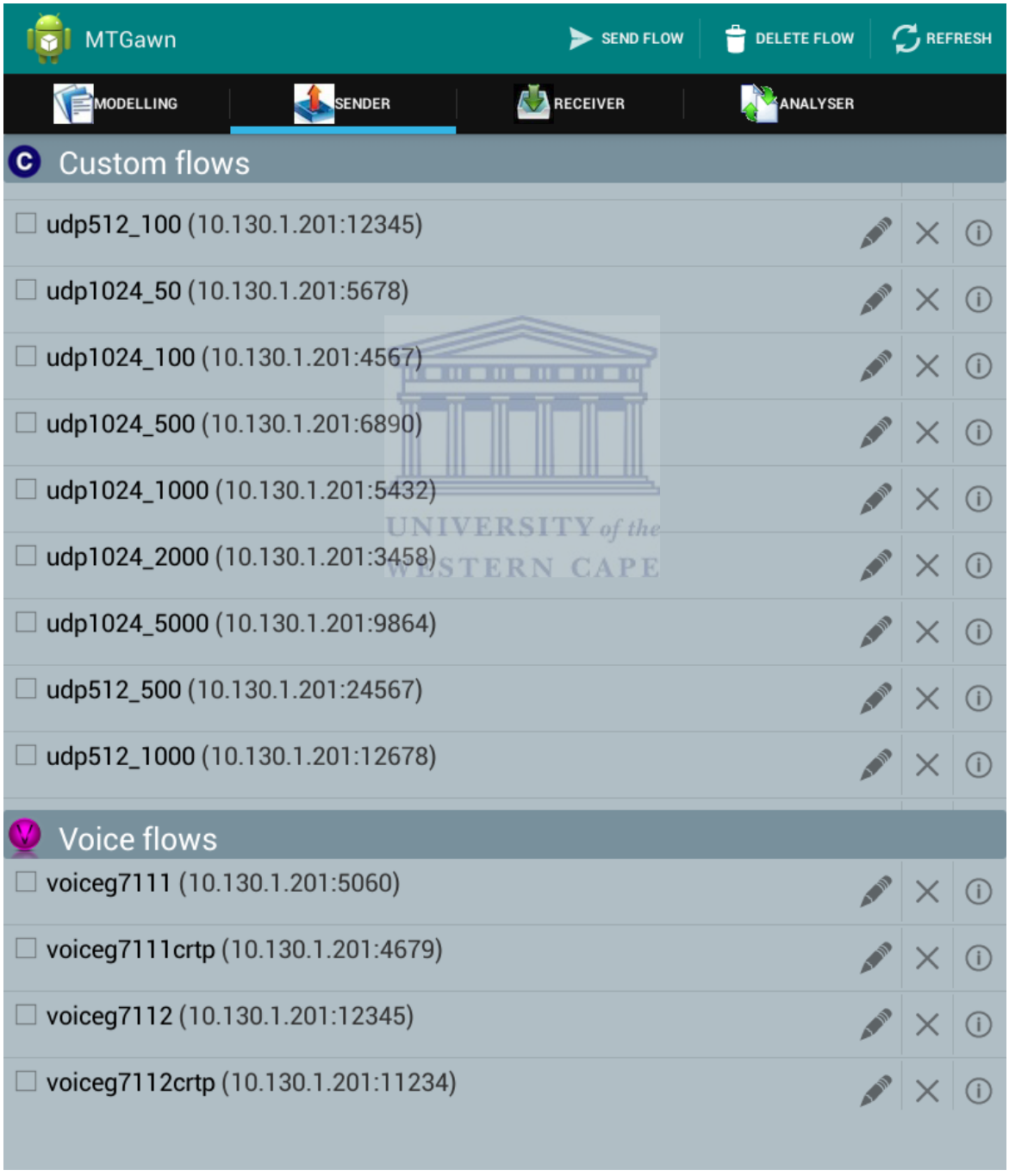


Figure 3.10: MTGawn Sender interface

Receiver

The 'receiver' module role is to receive the flow, manage it and control the parallel incoming flows. The relevant information about each packet received is saved in the receiver log file. The user has to start the receiver before beginning sending data, and stop the receiver when he or she does not want to accept incoming flows anymore (see Figure 3.11).

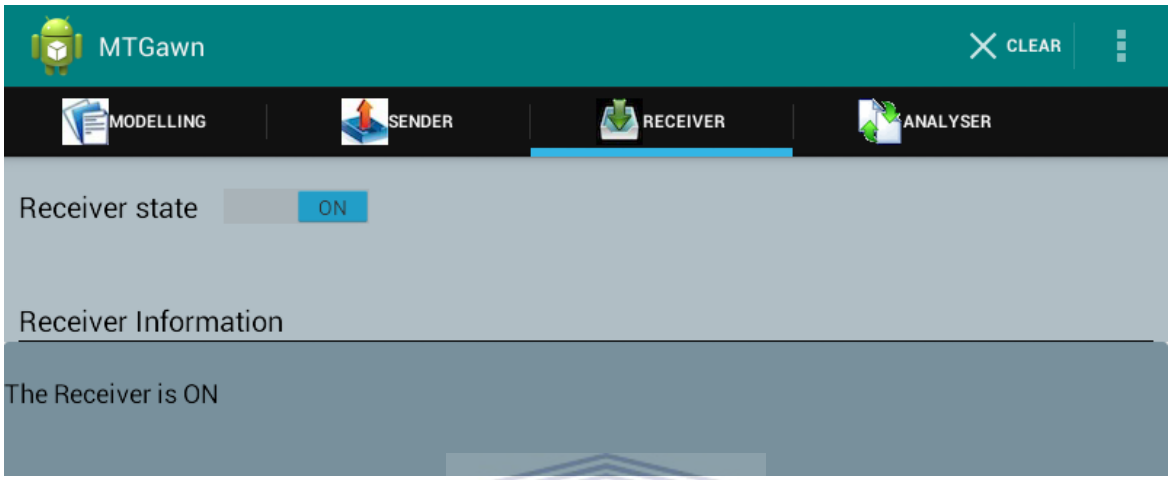


Figure 3.11: MTGawn Receiver interface

Analyser

The 'analyser' module is responsible for analysing the log files of senders and receivers in order to compute the QoS of the network under consideration. When the type of delay is set to one-way delay (OWD), performance metrics can be extracted from the receiver log file. If the delay is set to round trip time (RTT), the quality of service metrics of the network can be extracted from both the sender and receiver log files. Each log file contains detailed information, such as the flow identifier, the packet identifier, the time sent, the time received, the payload size, the source and destination IP address and port number of the packet—for each packet sent and received during the generation process. Figure 3.12 is one screenshot of the analyser interface.

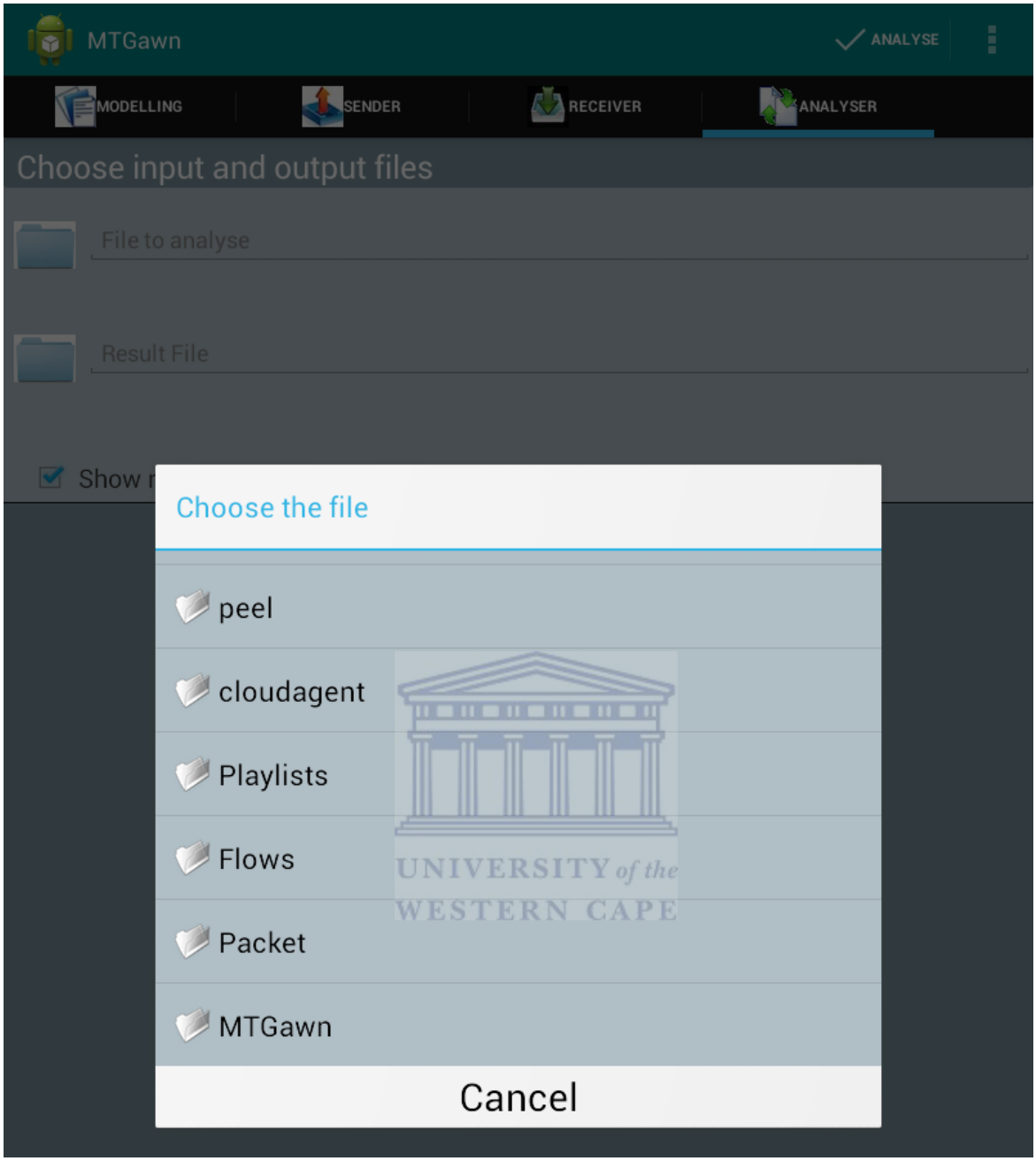


Figure 3.12: MTGawn Analyser interface

3.2.2.2 Prototype A: Single flow

Prototype A entails modelling (Modelling module), generation (Sender module), capturing (Receiver module) and analysing (Analyser module) of single TCP, UDP and VoIP traffic flows. To model custom TCP and UDP traffic, both packet size and time between two consecutive packets (inter-arrival time) in the network were emulated using various probability distributions such as Constant, Uniform, Exponential, Poisson, Normal, Lognormal (Gaussian), Gamma, Geometric, Cauchy, and Pareto and Weibull distribution (see Appendix A for a detailed description of the different distribution types) to randomize them. The constant distribution was used in the

experiments. For VoIP flow, a codec used to encode and decode the voice determine the packet size and the packet rate of the flow.

Each flow contains either a predefined number of packets to be generated or the duration of the flow in milliseconds. The traffic generation and capturing of single flows in MTGawn is performed as follows:

i. For custom UDP and TCP flows, the first packet is randomly generated with a packet size and a packet inter-arrival time that follow the distribution chosen for each of them respectively. For voice flows, packets are generated according to packet size and inter-departure time corresponding to the codec used for voice coding/decoding.

ii. Then, the generation is delayed for the period of time (inter-arrival time), which represents the time between the transmission of the current packet and the transmission of the next packet.

iii. After the elapsed time, another packet is randomly generated according to the same packet size distribution and inter-arrival time distribution chosen.

iv. The system moves back to step (ii) and the same cycle is repeated until all packets are generated.

Figure 3.13 illustrates the interaction between sender and receiver during the transmission process.

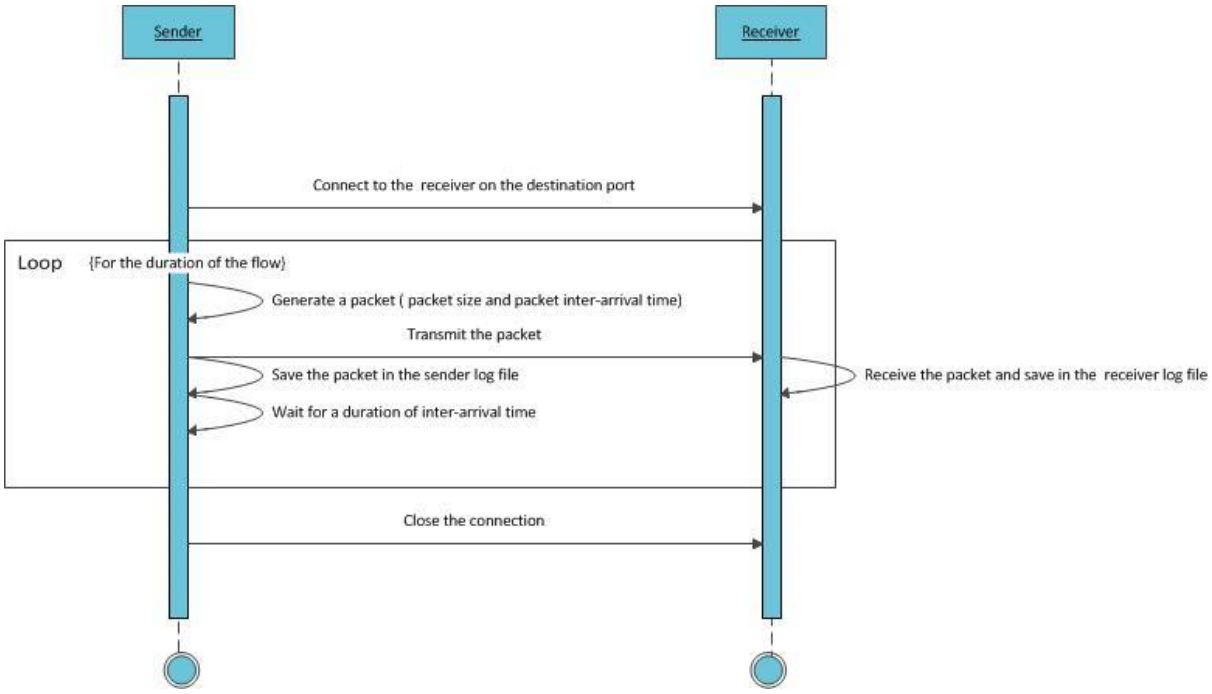


Figure 3.13: Generation of a single flow in the MTGawn system

3.2.2.3 Prototype B: multiple flows

Figure 3.14 illustrates the functioning of MTGawn when multiple flows are sent simultaneously. To allow transmission of multiple flows, a thread was created for each flow to be transmitted and another thread was created to receive the flow at the receiver side. A thread is a concurrent unit of execution. Java multithread ability was used to allow the sending of flows simultaneously in parallel. By definition multitasking is when multiple processes share common processing resources such as a central processing unit (CPU). Multithreading extends the idea of multitasking into applications where you can subdivide specific operations within a single application into individual threads. Each of the threads can run in parallel. The operating system (OS) divides processing time not only among different applications, but also among each thread within an application. All threads run concurrently and each thread handles its allocated flow while at the same time making optimal use of the available resources such as multiple CPUs.

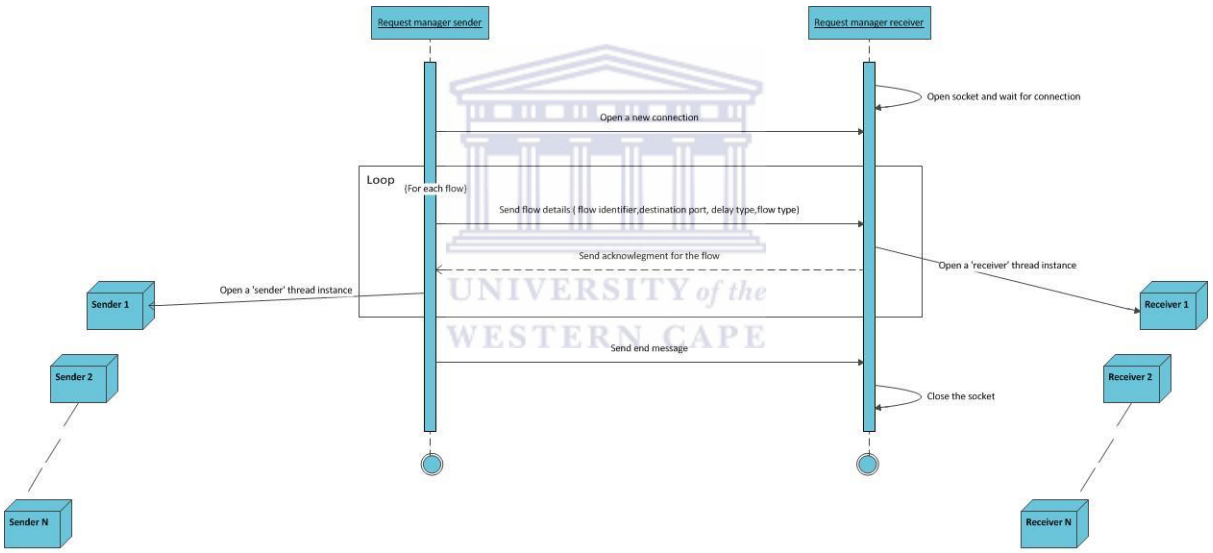


Figure 3.14: Generation of multiple flows in the MTGawn system

3.2.3 Phase 3: Prototype evaluation and report

After deployment of a prototype for the MTGawn tool, the aim of this phase was to evaluate each prototype and to draw conclusion from the results obtained. The evaluation of the prototype involved the testing of the prototype using a wireless network deployed in a laboratory testbed environment (see section 3.2.3.1). The wireless network testbed was deployed in a research lab.

Several experiments were performed. During each experiment packets were transmitted by the sender and then captured by the receiver. The sender and receiver generated log files containing detailed information about each packet transmitted and received during the generation process. This phase also involved the processing of the captured traffic i.e. analysis of the log files of both

sender and receiver. Performance metrics were chosen to determine the functioning of the network or protocol under evaluation. VoIP and UDP based applications are very sensitive to delay and jitter while TCP based applications are more sensitive to packet loss. Thus the following evaluation criteria were defined: throughput, delay, jitter and packet loss (see section 3.2.3.2). After the generation and capturing of flows, the prototype was programmed to perform a statistical analysis of the log files. EXCEL was used to interpret the results. Results were interpreted and presented as text, tables, graphs, diagrams and charts.

The design of MTGawn was inspired by a highly regarded computer-based traffic generator called D-ITG. For each experiment executed using MTGawn, a similar experiment was executed using D-ITG. For each prototype, a comparative analysis was done in order to confirm the feasibility of the tool. Hence, the results obtained using MTGawn were compared to results obtained using D-ITG. MTGawn and D-ITG experiments were tested in the same laboratory testbed environment.

3.2.3.1 Experiment setup

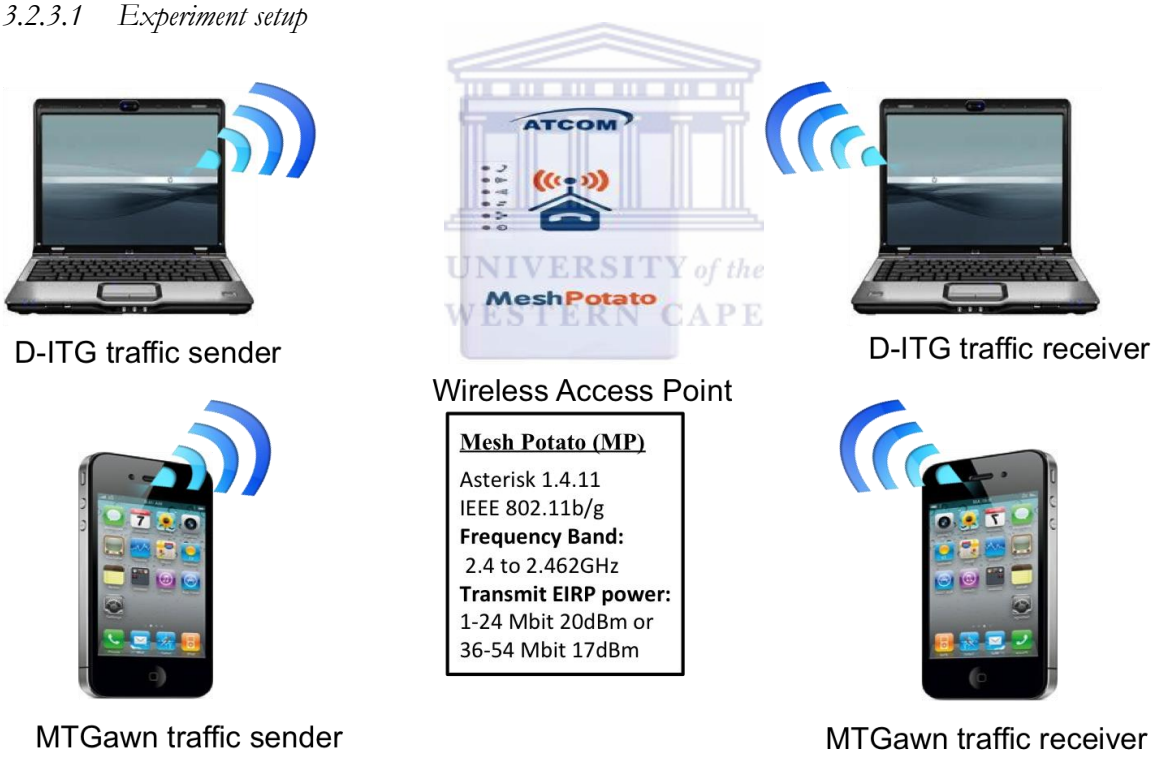


Figure 3.15: Wireless testbed network.

Figure 3.15 illustrates the network testbed environment, which consisted of a wireless mesh network (WMN) deployed in a laboratory. A small wireless network, which consisted of a mesh potato (MP) device acting as a wireless access point was designed and configured. Two Android phones were connected to the wireless network and configured, one as a sender and the other as a receiver. The MTGawn tool was installed on both phones to send and receive traffic. Similarly

two computers were wirelessly connected to the testbed network; both computers had D-ITG installed on them. One computer was set to act as a D-ITG traffic sender and the other one as a D-ITG traffic receiver. The distance between the two mobile phones and the distance between the two computers was approximately 5 meters.

In the experiments, the two Android phones used and the two computers had different requirements in terms of hardware and software:

- **D-ITG traffic sender:** Ubuntu 14.04 (trusty) 64-bit with Processor (Intel® Core™ 2 Duo CPU E6550 @ 2.33 GHz) and 4GB memory.
- **D-ITG traffic receiver:** Ubuntu 12.04 (precise) 64-bit with Processor (Intel® Core™ i3 CPU 540 @ 3.07GHz * 4) and 3,7 GiB memory.
- **MTGawn traffic sender:** Samsung Galaxy Tab 3 10.1 P5200 running Android OS v4.2.2 (Jelly Bean); Chipset: Intel Atom Z2560; CPU: Dual-core 1.6 GHz; GPU: PowerVR SGX544MP2 and 1 GB RAM. Wi-Fi 802.11 a/b/g/n, dual-band, Wi-Fi Direct.
- **MTGawn traffic receiver**: Samsung Galaxy Note 10.1 N8000 running Android OS v4.0.3 (Ice Cream Sandwich); Chipset: Exynos 4412 Quad; CPU: Quad-core 1.4 GHz Cortex-A9; GPU: Mali-400MP4 and 2 GB RAM. Wi-Fi 802.11 a/b/g/n, dual-band, Wi-Fi Direct.
- **The mesh potato (MP) device**: Atheros AR2317 system on a Chip (SoC) running Asterisk 1.4.11 Firmware with MIPS 4k processor 180 MHz and 16 MByte RAM.

3.2.3.2 Performance metrics

As previously mentioned, different applications (data/voice/multimedia) have different requirements for network service in order to be useful to users—some applications are impaired by message loss and others are impaired by delay or jitter (Marsic, 2013). This research focused on throughput, delay, jitter and packet loss which were computed as follow:

i. **Throughput:** is defined as the number of bits transmitted or received per unit of time. Throughput is computed by dividing the total number of bits sent or received by the total time (duration) of the generation.

$$\textbf{Throughput} = \frac{\textbf{\textit{total number of bits sent or received}}}{\textbf{\textit{total time}}}$$

ii. **Delay:** If S_i is time of transmission for packet *i* and R_i is the time of arrival for packet *i*, then the delay (D) is determined as the difference between the received and transmitted time.

$$D_i = R_i - S_i$$

Average Delay = $\frac{\sum_i^n D_i}{n}$ where *n* is the total number of packet received.

iii. **Jitter:** The inter-arrival jitter is defined as the difference (D) in packet spacing at the receiver (R) compared to the sender (S) for a pair of packets. For two packets (i and j), D is expressed as:

$$D(i, j) = (R_i - R_j) - (S_j - S_i) = (R_j - S_j) - (R_i - S_i)$$

Average Jitter = $\frac{\sum_i^n |D(i-1,i)|}{n}$ where *n* is the total number of packet received.

iv. **Packet loss:** The (amount of) packet loss is defined as the difference between the numbers of transmitted and received packets. If n represents the number of packets transmitted and m represents the number of packets received, then the percentage of packet loss is defined as:

Packet loss (%)= $\left(\frac{n-m}{n}\right) \times 100 = 100 - \frac{m}{n} \times 100$

3.3 SUMMARY

In this chapter, the four important elements that influence the way in which research is undertaken: Epistemology; theoretical perspectives; methodology and methods, were discussed.

Epistemology and theoretical perspectives describe the philosophical assumptions about the nature of reality, knowledge and how it can be obtained. These assumptions guide the research and justify the chosen methodology and methods. Design science research (DSR) was the methodology used to design a prototype for the MTGawn system. The next chapter describes and summarizes the main results obtained during the testing of the prototypes.

Chapter 4

RESULTS

The previous chapter outlined the approach and methodology adopted throughout this research. A cyclical Design Science Research (DSR) methodology was applied during the research process to design a prototype to address the research problem in an incremental manner. The research process went through many cycles. Each cycle consisted of identifying or redefining the research questions in order to clarify the research problem (Phase 1), then addressing the problem by designing or improving an existing prototype to solve the problem (Phase 2), thereafter testing the prototype and interpreting the results obtained (Phase 3). A prototype solves an instance of the problem or improves an existing prototype by adding new functionalities to the system. Each cycle lead to the creation or the improvement of the prototype. Prototype A was concerned with the modelling, sending and receiving of single traffic flow and Prototype B was concerned with the modelling, sending and receiving of multiple traffic flows

In this chapter the results achieved for different traffic flows using the MTGawn prototypes are discussed and compared to those results using D-ITG. The aim was to determine if the performance of MTGawn compared favourably to that of D-ITG (D-ITG was considered as the benchmarking tool). In order to do so, several experiments were completed for each prototype (see Table 4.1). The same traffic parameters were used for both D-ITG and MTGawn. Results were classified in terms of throughput, packet loss, delay (measured in terms of RTT delay) and jitter. Both traffic generators have the ability to generate log files at both receiver and sender side.

Table 4.1: Experiments done for each prototype.

Prototype A	**Prototype B**
Experiment 1: *Single UDP flow* Experiment 2: *Single TCP flow* Experiment 3: *Single VoIP flow*	Experiment 1: *VoIP flow with UDP background traffic.* Experiment 2: *Multiple VoIP flows*

4.1 RESULTS FOR PROTOTYPE A: SINGLE FLOW

The first DSR cycle lead to the design of MTGawn Prototype A which allowed the sending and receiving of single TCP, UDP and VoIP flows. UDP and TCP flows were generated using various distributions for the representation of both time between packets and packets size. Voice flow was generated taking into consideration various codecs, and included many samples for each type of codec. To perform this test, single UDP, TCP and VoIP traffic flows were generated and transmitted independently over the wireless network. To improve the accuracy of the results, several independent runs were performed for each experiment (i.e. each flow) for both D-ITG and MTGawn and the average results, for each case, were calculated. For UDP and TCP flows, a constant distribution (see Appendix A) was used to model both packet size and the time between two consecutive packets (time between two consecutive packets in millisecond = 1000/packet rate).

Table 4.2 and Table 4.3 summarize the parameters used for the experimentation. The duration of all the experiments was set to 1 minute (60 s).

Table 4.2: Parameters for UDP and TCP traffic generation.

Protocol	**Packet size**	**Packet rate**
Protocol = UDP or TCP	Size distribution = Constant Packet size (bytes) S = {128; 256; 512; 1024}	Time distribution = Constant Packet rate (Pkt/s) R = {100; 500; 1000; 2000; 3000; 4000; 5000}

Table 4.3: Parameters for VoIP traffic generation

Codecs	**Samples**	**Frame size**	**Frame rate**	**Packet size**
G.711.1	1	80	100	80
G.711.2	2	80	50	160
G.729.2	2	10	50	20
G.729.3	3	10	33	30
G.723.1	1	30	26	30

4.1.1 Single UDP flow

i. Throughput

Figure 4.1 (as well as Figure C.1, Figure C.2 and Figure C.3 in Appendix C) reports the comparison between the expected throughput and the actual throughput at sender and receiver side for different traffic loads. Traffic load (Bandwidth) depends on packet size and packet rate variation. In fact, the expected throughput is expressed as the product of packet rate and packet size. Overall, D-ITG and MTGawn show consistent similarity. However, D-ITG presents a slightly higher performance than MTGawn except for the case when the packet rate = 5000 Pkt/s and packet size = {128, 256, 512} bytes. In the case of R=5000 Pkt/s and S=128 bytes, D-ITG reaches a throughput of 2152,79 Kbps while MTGawn reaches a throughput of 2275,19 Kbps at sender side. D-ITG reaches a throughput of 2058,65 Kbps while MTGawn reaches a throughput of 2175,59 Kbps at receiver side as shown in Table 4.4. It is important to notice the fact that when the rate becomes greater than 2000 Pkt/s, the performances of both tools reduces considerably with respect to throughput, as shown in Figure 4.1 and Appendix C.

Table 4.4: Throughput for single UDP flow with packet rate R = 5000Pkt/s

Packet size	**D-ITG (Kbps)**		**MTGawn (kbps)**	
	Sender	Receiver	Sender	Receiver
S = 128 bytes	2152,79	2058,65	2275,19	2175,59
S = 256 bytes	3821,85	3623,40	3967,52	3965,85
S = 512 bytes	5864,53	5858,65	6107,75	6077,66

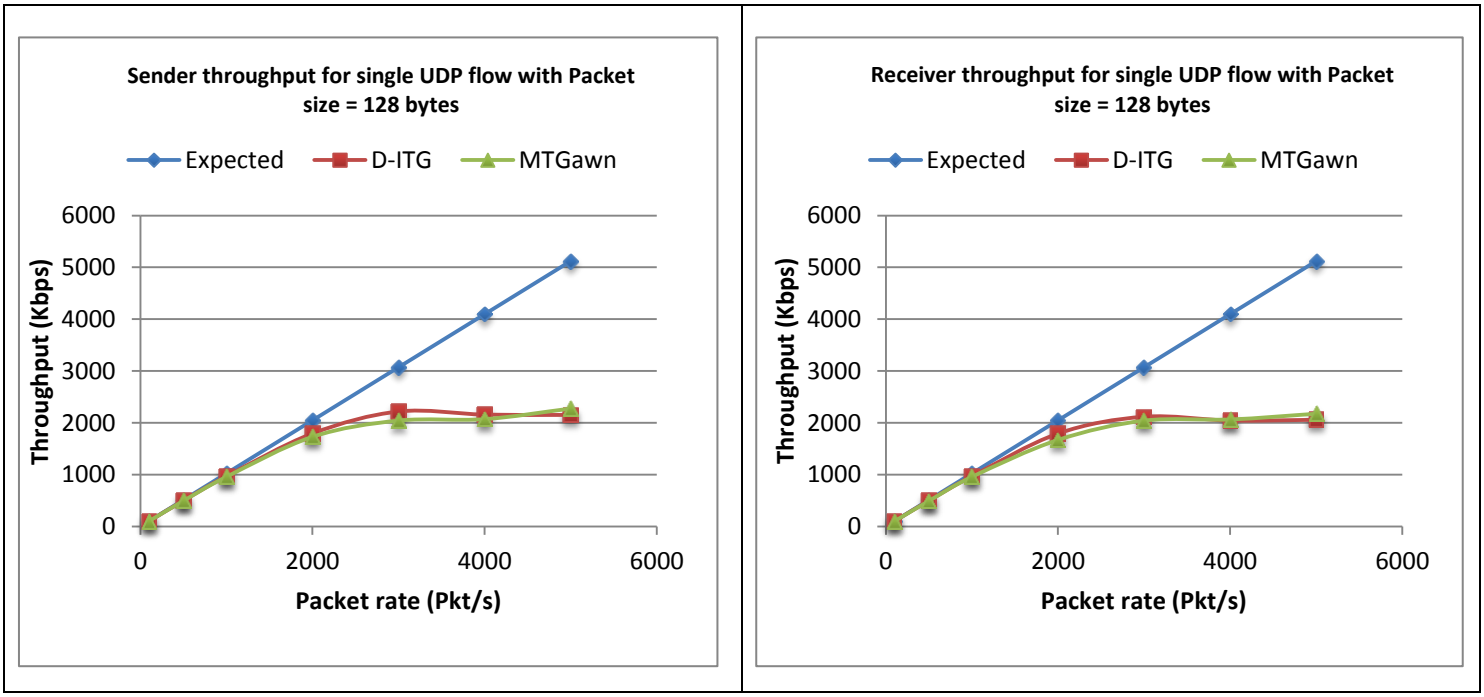


Figure 4.1: Throughput for single UDP flow at sender side (left) and receiver side (right) for packet size = 128 bytes

ii. Packet loss

In this section, packet loss and loss rate between the expected data rate and the actual data rate are reported for single UDP flows for different packet sizes. Figure 4.2 and Figure 4.3 illustrate the increase in loss rate as the packet size and the packet rate grow. MTGawn shows a lower loss rate when compared to D-ITG, in the case where packet rate is equal to 5000 Pkt/s and packet size = {128, 256, 512} bytes.

Both D-ITG and MTGawn show low loss rates when the packet rate is below 1000 Pkt/s. For a packet rate less or equal to 1000 Pkt/s, at sender side D-ITG reaches a loss rate less than 7,8 % while MTGawn reaches a loss rate less than 7,95 %. At receiver side, D-ITG reaches a loss rate less than 10,3 % and MTGawn reaches a loss rate less than 7,96%. As the packet rate exceeds 1000 Pkt/s, the percentage loss rate increases dramatically for both tools. In the case of R = 5000 Pkt/s and S= 1024 bytes, D-ITG reaches a 76,49 % loss rate and MTGawn reaches a 80,24%, at (s) sender side. Meanwhile, D-ITG reaches a 77,69 % loss rate and MTGawn reaches a 80,32% loss rate, at receiver side. A detailed analysis of single UDP loss rate can be found in Appendix C (see Figure C.4, Figure C.5, Figure C.6 and Figure C.7).

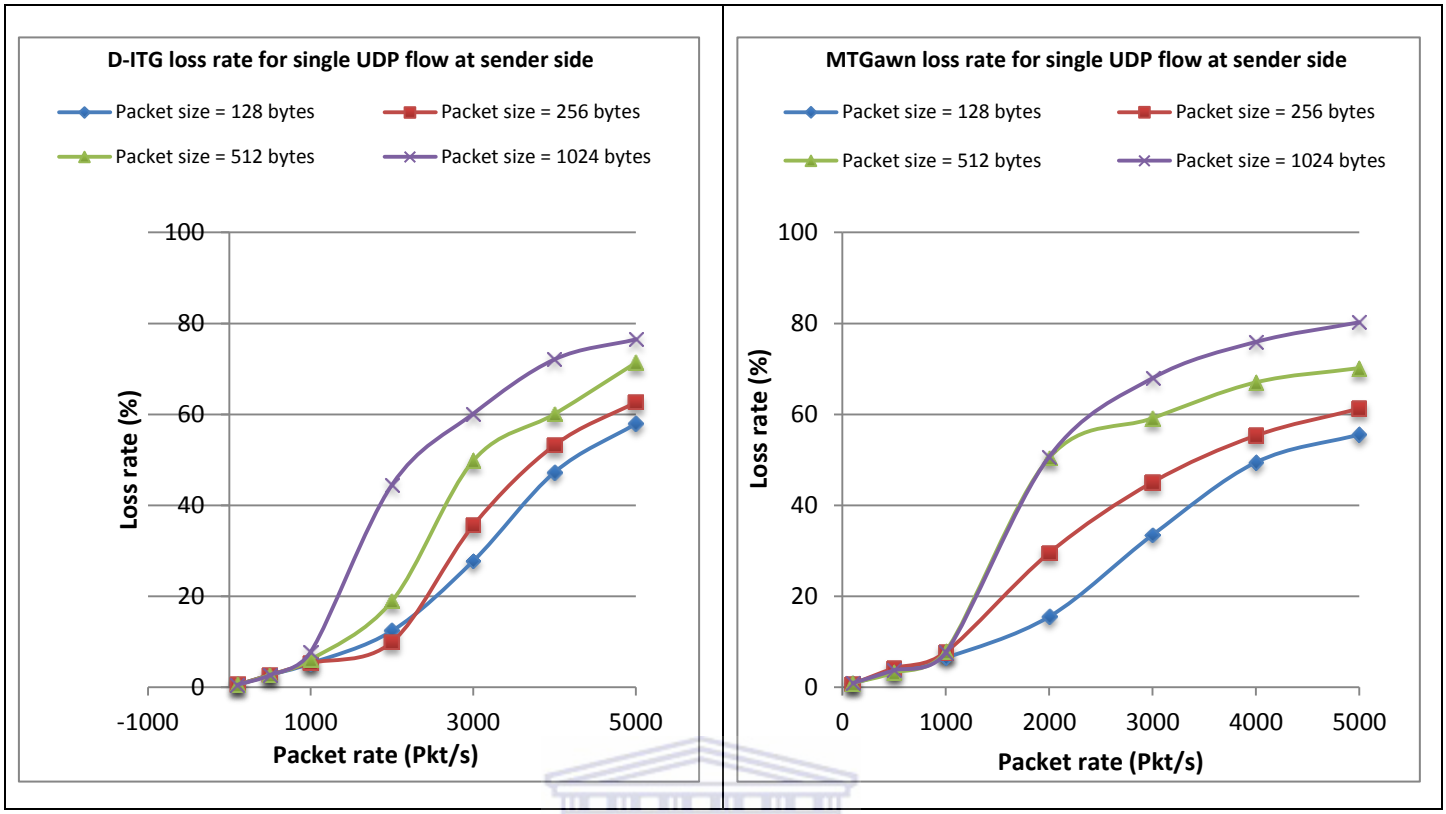


Figure 4.2: D-ITG loss rate (left) and MTGawn loss rate (right) for single UDP flow at sender side

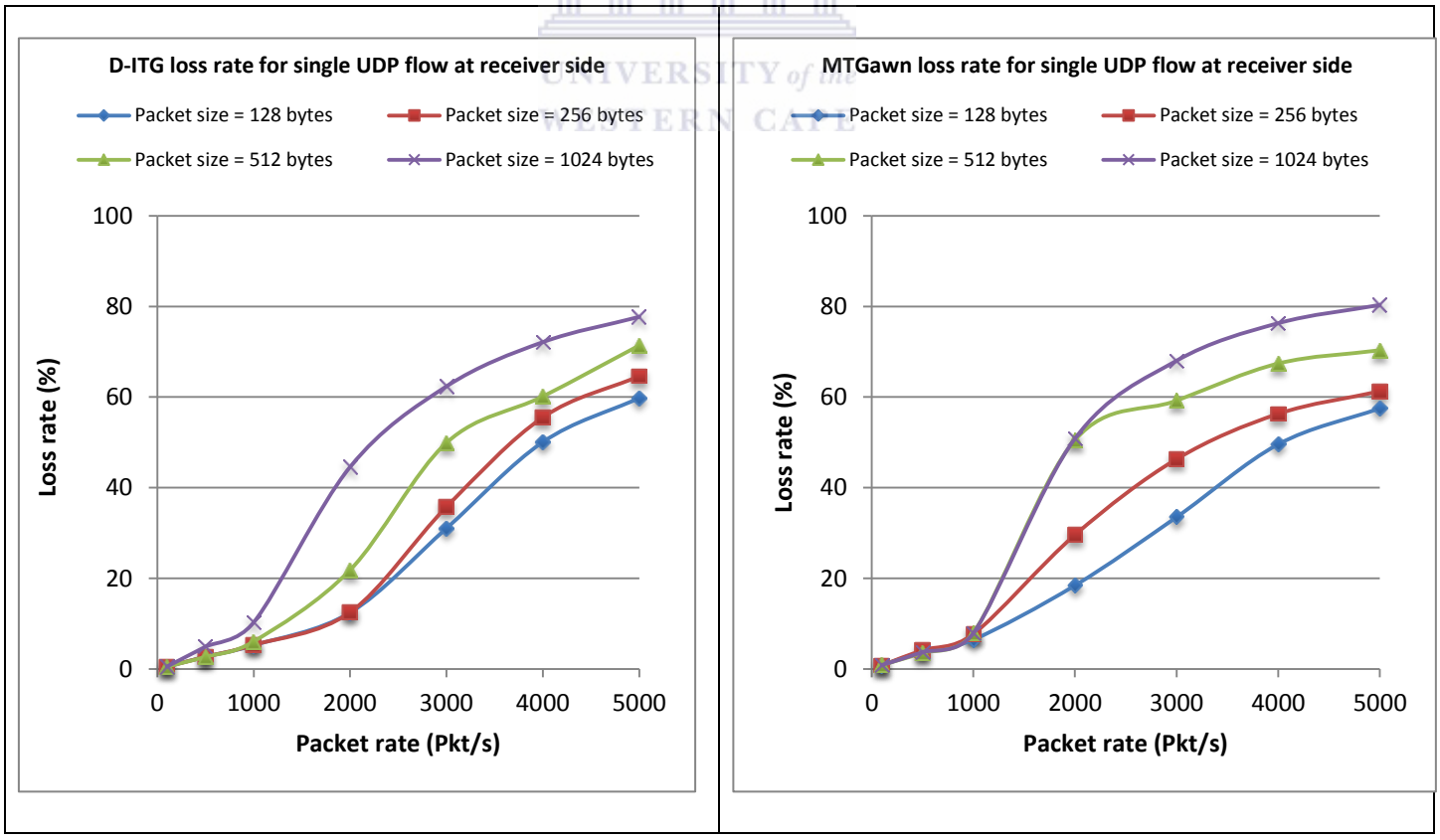


Figure 4.3: D-ITG loss rate (left) and MTGawn loss rate (right) for single UDP flow at receiver side.

Figure 4.4 presents the packet loss variation for single UDP flow in D-ITG and MTGawn. Packet loss is referred to as the number of packets that never reach their host destination due to numerous reasons such as insufficient bandwidth or overload of links or nodes in a transmission path.

For packet size = 100 Pkt/s, packet loss is below 1% for both tools but reaches a maximum of 7,07% in D-ITG and 8,42% in MTGawn as the packet rate increases. The highest packet loss in D-ITG is 7,07% and corresponds to packet size = 1024 bytes and packet rate = 4000 Pkt/s and the lowest is 0,18% which corresponds to packet size = 128 bytes and packet rate = 100 Pkt/s. In MTGawn, the lowest packet loss is 0,43% and corresponds to packet size = 128 bytes and packet rate = 100Pkt/s, while the highest is 8,42% and correspond to packet size = 1024 bytes and packet rate = 5000 Pkt/s

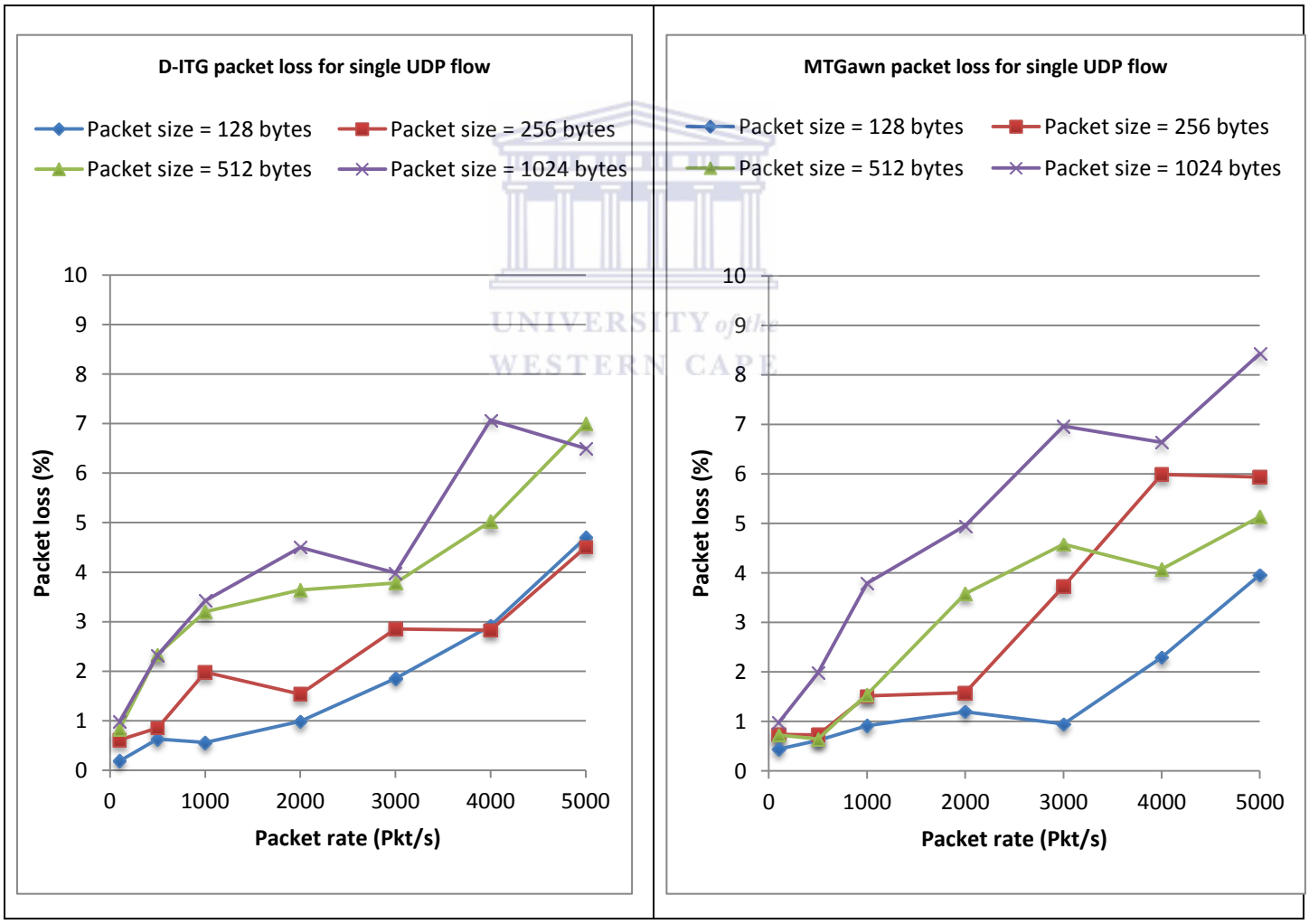


Figure 4.4: D-ITG packet loss (left) and MTGawn packet loss (right) for single UDP flow

iii. RTT delay

Figure 4.5 illustrates the RTT delay for single UDP flow. RTT delay varied between 44 ms and 90 ms in D-ITG. Meanwhile the delay varied between 20 ms and 86 ms in MTGawn. The highest delay in D-ITG is observed in the case of packet size = 128 bytes and packet rate = 3000 Pkt/s and the lowest delay is observed in the case of packet size = 1024 bytes and packet rate = 100 Pkt/s. However for MTGawn the lowest delay is observed in the case of packet size = 1024 bytes and packet rate = 2000 Pkt/s and highest delay is observed in the case of packet size = 512 bytes and packet rate = 5000 Pkt/s.

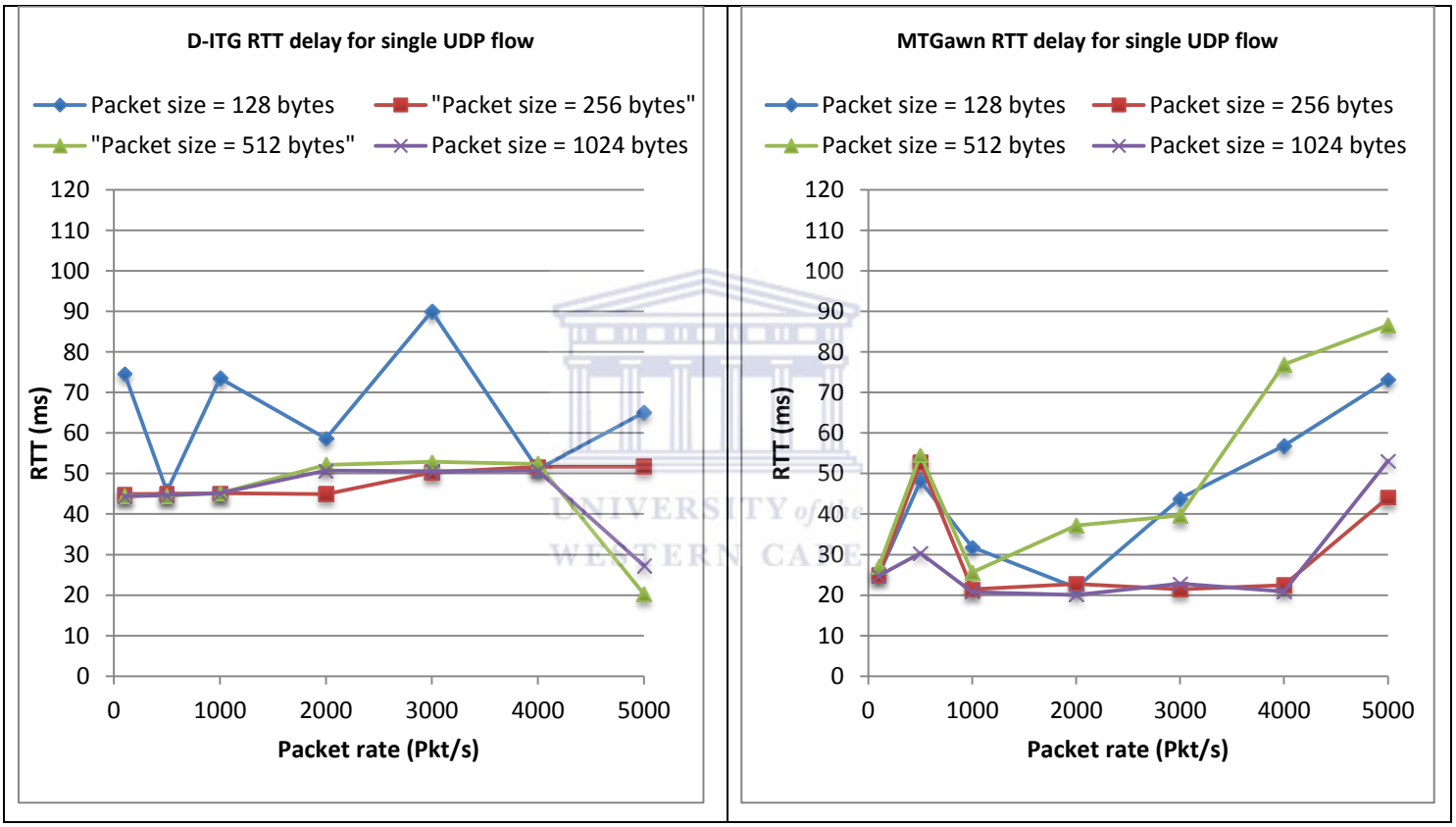


Figure 4.5: D-ITG RTT delay (left) and MTGawn RTT delay (right) for single UDP flow

iv. Jitter

Figure 4.6 illustrates the variation of delay also referred to as jitter for a single UDP flow. A high variation of jitter may be caused by various factors including bandwidth limitation; delay caused by network congestion and packets arriving out of order. For different packet sizes, jitter varies between 0,309 ms and 1,447 ms for D-ITG and between 0,616 ms and 1,156 ms for MTGawn. Both the highest and lowest variation of delay is observed in D-ITG and corresponds to the cases of S = 256 bytes and R = 100 Pkt/s and S = 128 bytes and R = 1000 Pkt/s, respectively.

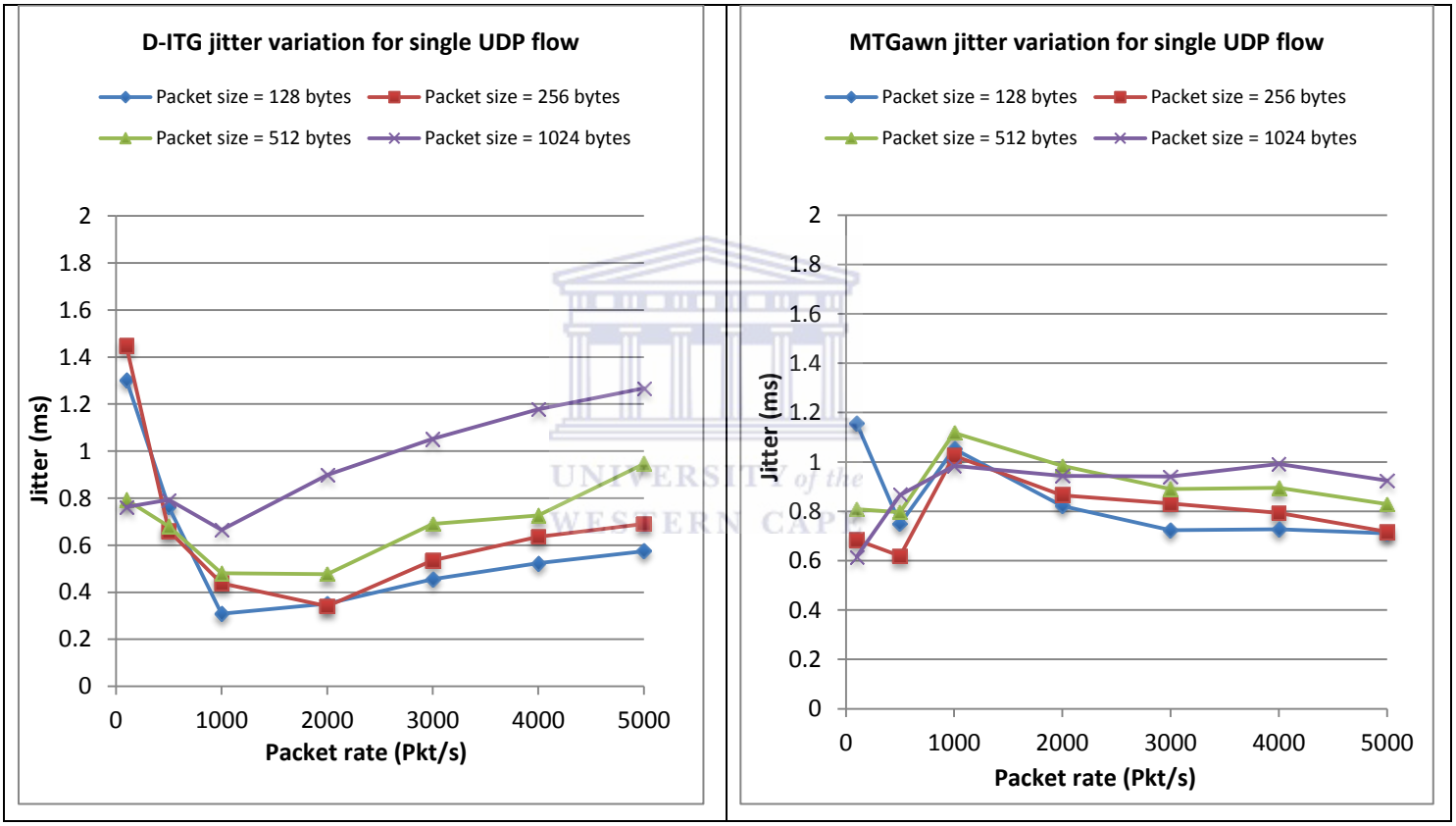


Figure 4.6: D-ITG jitter (left) and MTGawn jitter (right) for single UDP flow

4.1.2 Single TCP flow

i. Throughput

Similar to UDP traffic flow, numerous traffic conditions were represented using TCP. Figure 4.7 and Appendix C (see Figure C.8, Figure C.9 and Figure C.10) report the comparison between the expected throughput and the actual throughput at sender and receiver side for a single TCP flow with different traffic loads. Similar to single UDP flow, D-ITG and MTGawn show consistent similarity. However, MTGawn presents higher throughput compared to D-ITG in the case of packet rates less than 2000 Pkt/s for different packet sizes (except for packet size = 1024 bytes and packet rate = 1000 Pkt/s). For example, in the case of R= 100 Pkt/s and S=128 bytes, D-ITG reaches a throughput of 101,84 Kbps while MTGawn reaches a throughput of 102,16 Kbps at sender side. D-ITG reaches a throughput of 101,84 Kbps while MTGawn reaches a throughput of 102,28 Kbps at receiver side as shown in Table 4.5.

As packet rate reaches and exceeds 2000 Pkt/s (or more), D-ITG shows a better throughput in comparison with MTGawn but the performances of both tools reduce considerably with respect to throughput, as illustrated in Figure 4.7 and Appendix C (see Figure C.8, Figure C.9 and Figure C.10).

Table 4.5: Throughput for single TCP flow with packet size = 128 bytes.

Packet rate (Pkt/s)	**Expected throughput (Kbps)**	**Sender (Kbps)**		**Receiver (Kbps)**	
		D-ITG	**MTGawn**	**D-ITG**	**MTGawn**
R =100	102,4	101,84	102,16	101,84	102,28
R=500	512	496,26	505,17	496,26	496,34
R=1000	1024	955,13	994,25	955,14	992,49

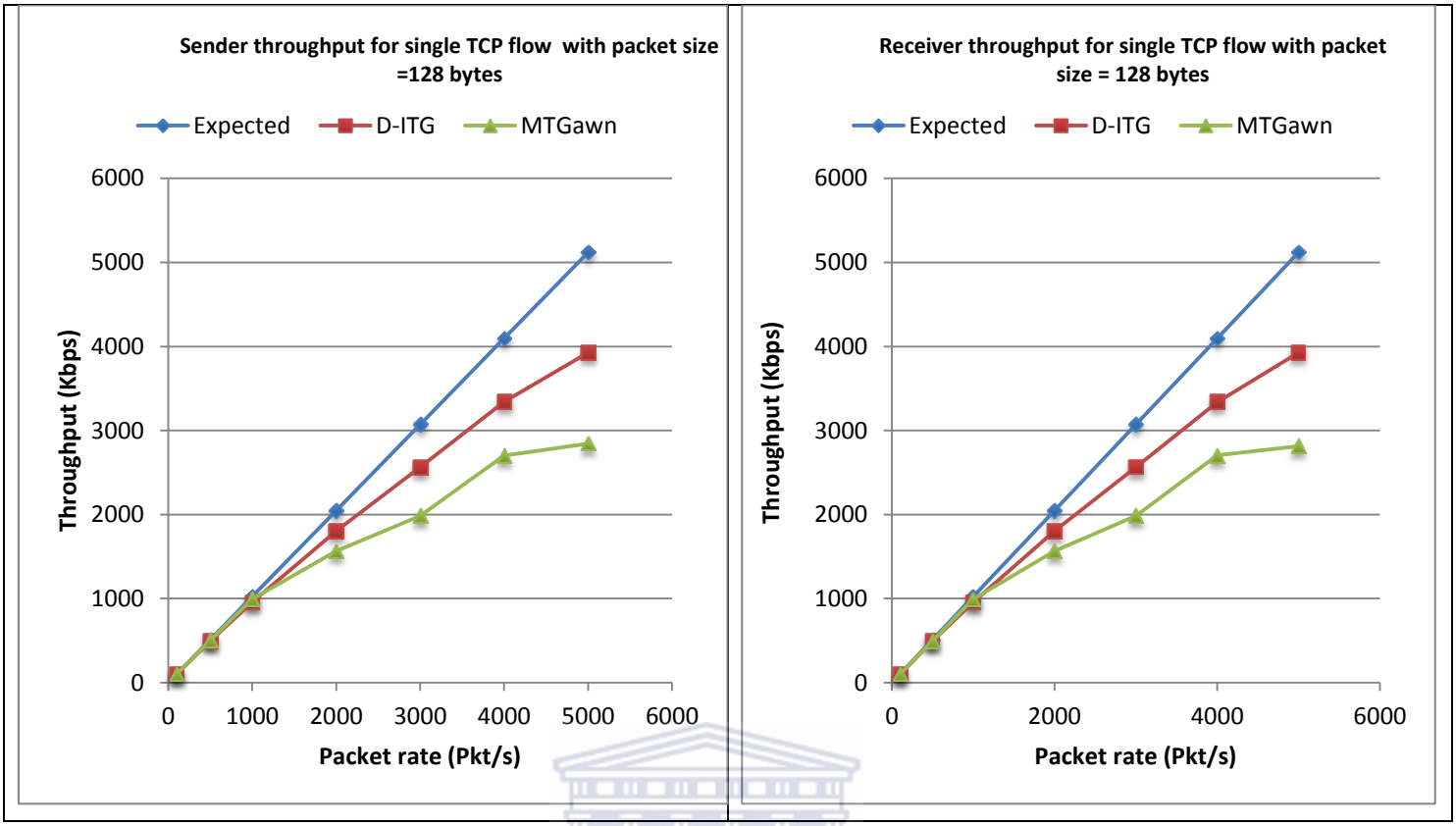


Figure 4.7: Throughput for single TCP flow at sender side (left) and receiver side (right) for packet size =128 bytes.

ii. Loss rate

TCP is a connection-oriented protocol, which uses a handshaking technique to ensure delivering, and reordering of packets at destination, thus packet loss is almost nonexistent with this protocol. Therefore, the focus is only on loss rate between the expected data rate and the actual data rate for this section. Figure 4.8 and Figure 4.9 illustrate the loss rate percentage at sender and receiver side for both D-ITG and MTGawn. For packet rates less than 2000 Pkt/s, MTGawn has the lowest loss rate, which is less than 4,17 % at sender side and less than 4,02 % at receiver side, while D-ITG loss rate is under 8,33% at both sender and receiver side for packet rates less than 2000 Pkt/s. When the packet rate is above 2000 Pkt/s, loss rate increases dramatically. At sender side, D-ITG reaches (up to) 83,26 % (which corresponds to R=4000 Pkt/s and S=1024 bytes) and MTGawn attains (up to) 88,11% (which corresponds to R=5000 Pkt/s and S=1024 bytes). At receiver side, D-ITG reaches (up to) 83,35 % (which corresponds to R=4000 Pkt/s and S=1024 bytes) and MTGawn attains (up to) 88,14% (which corresponds to R=5000 Pkt/s and S=1024 bytes). A detailed analysis of single TCP loss rate can be found in Appendix C (see Figure C.11, Figure C.12, Figure C.13 and Figure C.14).

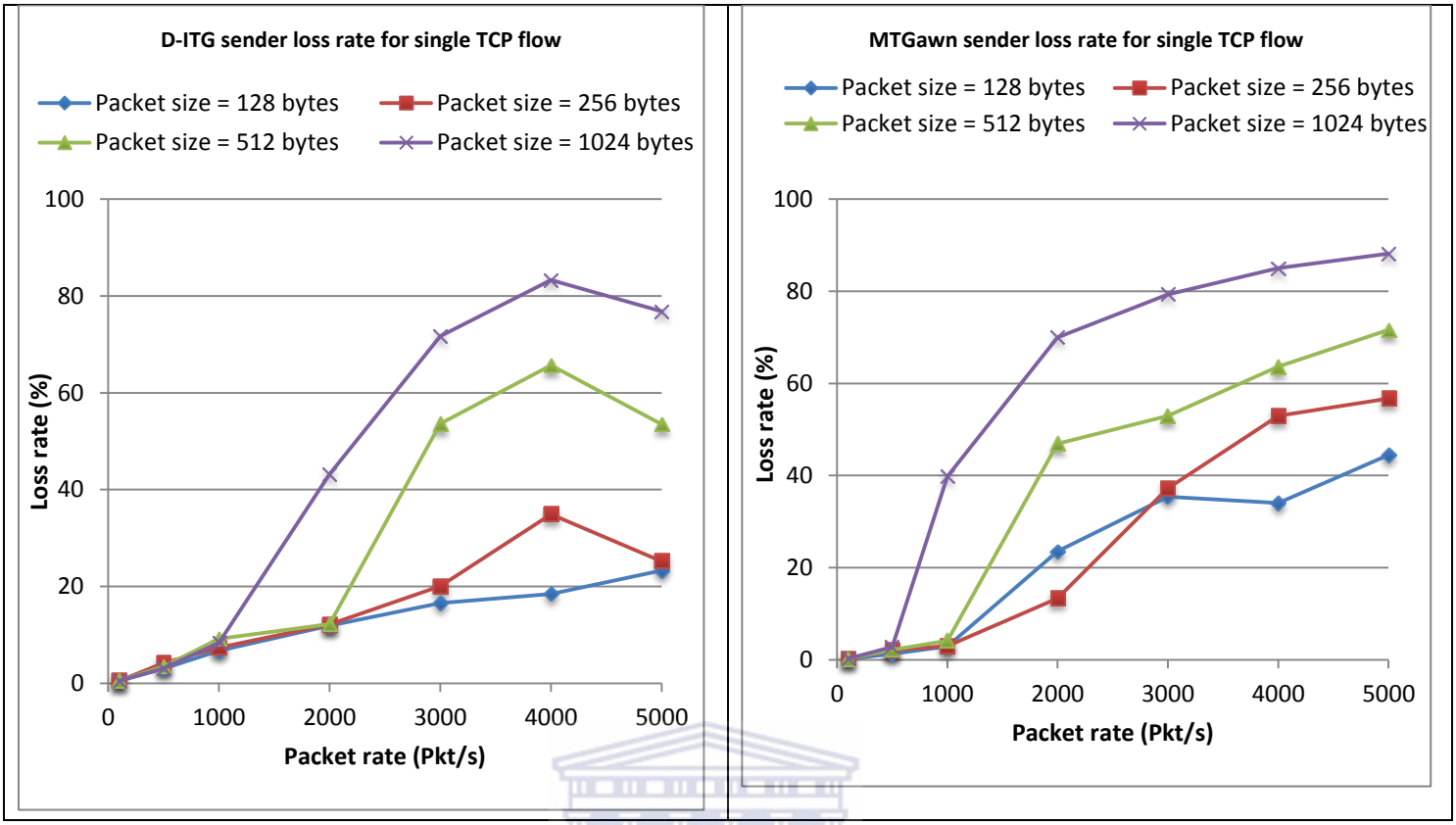


Figure 4.8: D-ITG loss rate (left) and MTGawn loss rate (right) for single TCP flow at sender side

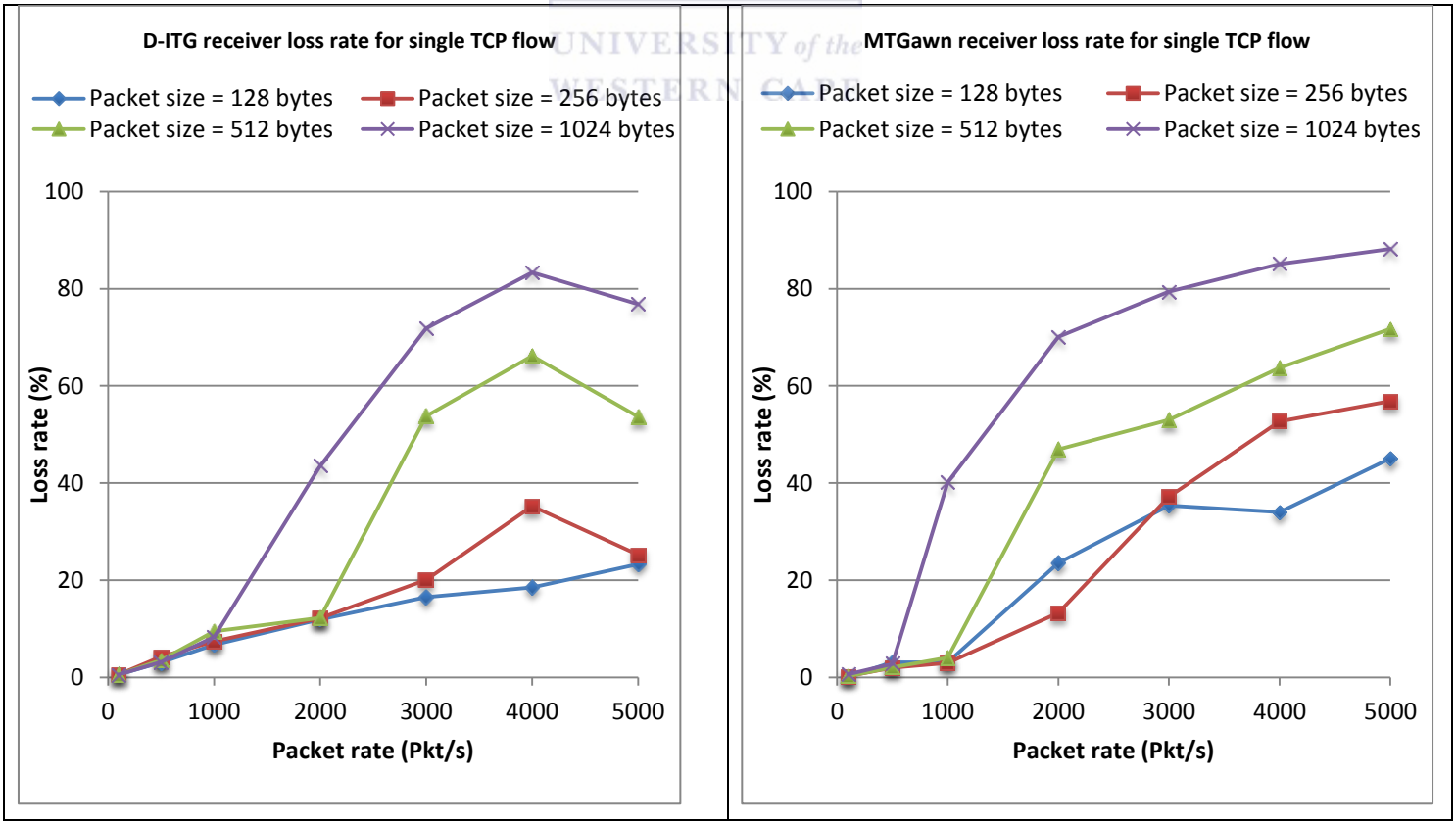


Figure 4.9: Loss rate for single TCP flow with D-ITG (left) and MTGawn (right) at receiver side

iii. RTT delay

Figure 4.10 illustrates the RTT delay for single TCP flow. Overall, the RTT delay increases as the packet size and the packet rate increase. In D-ITG the delay varied between 21 ms and 247 ms while in MTGawn the delay varied between 33 ms and 334 ms. Thus the highest delay in D-ITG and in MTGawn is observed in the case of packet size = 1024 bytes and packet rate = 5000 Pkt/s and correspond to 247 ms for D-ITG and 334 ms for MTGawn. The lowest delay in D-ITG is observed in the case of packet size = 128 bytes and packet rate = 100 Pkt/s. However for MTGawn the lowest delay is observed in the case of packet size = 512 bytes and packet rate = 100 Pkt/s.

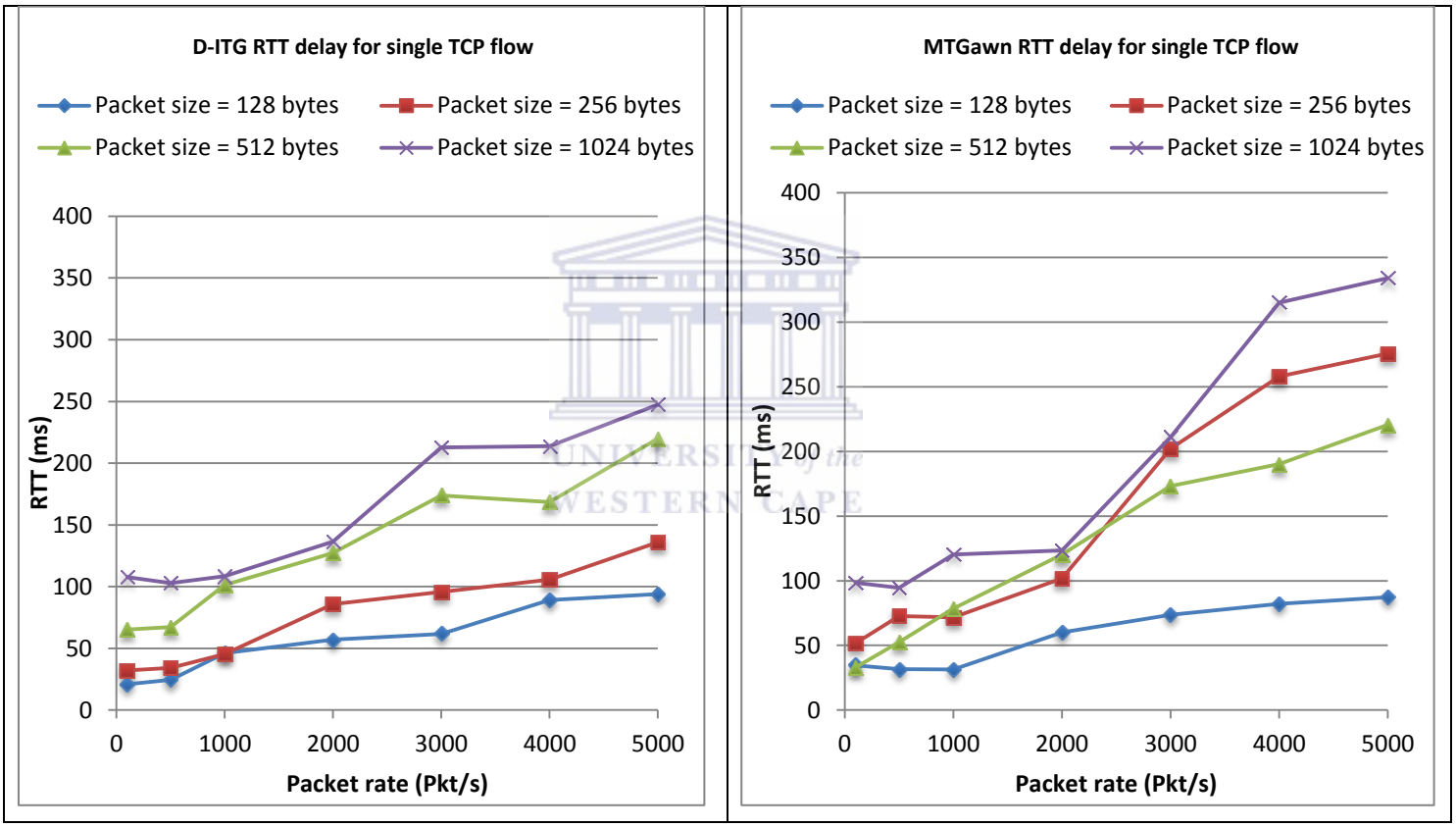


Figure 4.10: D-ITG RTT delay (left) and MTGawn RTT delay (right) for single TCP flow

iv. Jitter

Figure 4.11 illustrates the variation of delay for single TCP flow. For different packet sizes, jitter varies between 0,428 ms and 3,79 ms for D-ITG and between 1,644 ms and 5,12 ms for MTGawn. The lowest variation of delay is observed in D-ITG and corresponds to the case of S = 256 bytes and R = 5000 Pkt/s. The highest variation of delay is observed in MTGawn and corresponds to the case of S = 1024 bytes and R = 5000 Pkt/s.

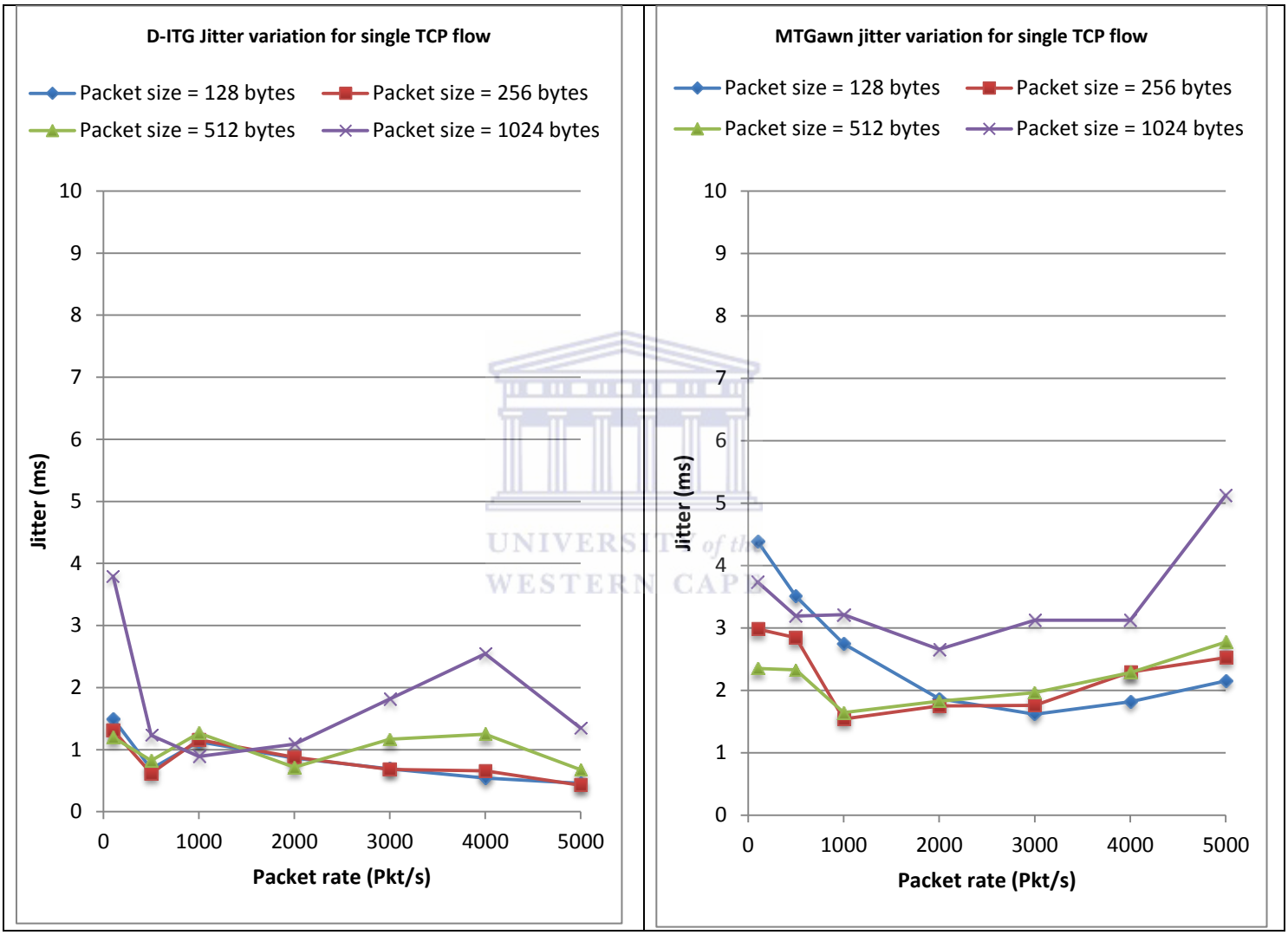


Figure 4.11: D-ITG jitter (left) and MTGawn jitter (right) for single TCP flow

4.1.3 Single VoIP flow

This section aims is to illustrate the analysis of VoIP performance through voice traffic generation and monitoring over a wireless network. In order to do so, single voice flows using different codecs were generated and transmitted over the network independently using both D-ITG and MTGawn as monitoring tools. Packet size and packet rate for each codec was predefined as shown in Table 4.3. Other voice parameters such as sample period and bandwidth used by the codec are of great importance since they have an impact on voice quality. The duration of each flow was set to 60 seconds, which is equivalent to a 1 minute VoIP call.

i. Throughput

Table 4.6 and Table 4.7 illustrate throughput using RTP and cRTP respectively for all the different codecs.

It can be seen that D-ITG and MTGawn have similar performances for a single voice flow with respect to throughput. The throughput using RTP protocol is higher compared to cRTP for all codecs for both D-ITG and MTGawn. For both tools, G.7.11.1 has the highest throughput, and the lowest throughput performance was that of G.723.1 codec. Using RTP, the performance of G.711.1 in MTGawn was 73,04 Kbps which was just lower than 73,18 Kbps of the same codec in D-ITG but comparable, and the throughput of G.723.1 was 8,72 Kbps for D-ITG and 8,83 Kbps for MTGawn.

Table 4.6: Throughput for single voice flow using RTP

Codec	D-ITG (Kbps)	MTGawn (Kbps)
G.711.1	73,182	73,043
G.711.2	68,566	68,570
G.723.1	8,727	8,830
G.729.2	12,765	12,758
G.729.3	11,065	11,168

Using cRTP, the performance of G.711.1 in MTGawn was 63,88 Kbps which was lower than 65,24Kbps of the same codec in D-ITG and the throughput of G.723.1 was 6,64 Kbps for D-ITG and 6,71 Kbps for MTGawn. From these two tables, it is noticeable that the MTGawn performs better with G.723.1 and G.729.3 codecs while D-ITG has better results with the other three codecs.

Table 4.7: Throughput for single voice flow using cRTP

Codec	D-ITG	MTGawn
G.711.1	65,241	63,882
G.711.2	64,591	64,580
G.723.1	6,649	6,715
G.729.2	8,776	8,768
G.729.3	8,434	8,516

ii. Packet loss

Packet loss has an important impact on VoIP since high packet loss ratios can cause degradation of voice quality. As mentioned in the literature, loss can occur due to many (for a variety of) reasons including insufficient bandwidth. In addition, packet loss variation is very unpredictable in wireless networks since it may be influenced, not only by the mobility of nodes but also by the distance between the hosts and the wireless access points in the network. When sending a single voice flow over the wireless mesh network, a different packet loss was experienced for each codec and for each protocol (RTP and cRTP) for both MTGawn and D-ITG. Figure 4.12 and Figure 4.13 illustrate the losses obtained for these experiments. For both tools, the percentage of loss was under 0,15%. The highest loss was observed on the G.729.2 codec using RTP in MTGawn with a value of 0,13%. The lowest loss was observed on the same G.729.2 codec using cRTP in D-ITG with a value of 0,003%. For RTP, D-ITG has a lower loss with G.711.1; G.723.1 and G.729.2 codecs while MTGawn has a lower loss with G.711.2 and G.729.3 codecs. Using cRTP, MTGawn has the lower loss for G.711.2 and G.723.1. D-ITG has the lower packet loss ratio using the others three codecs.

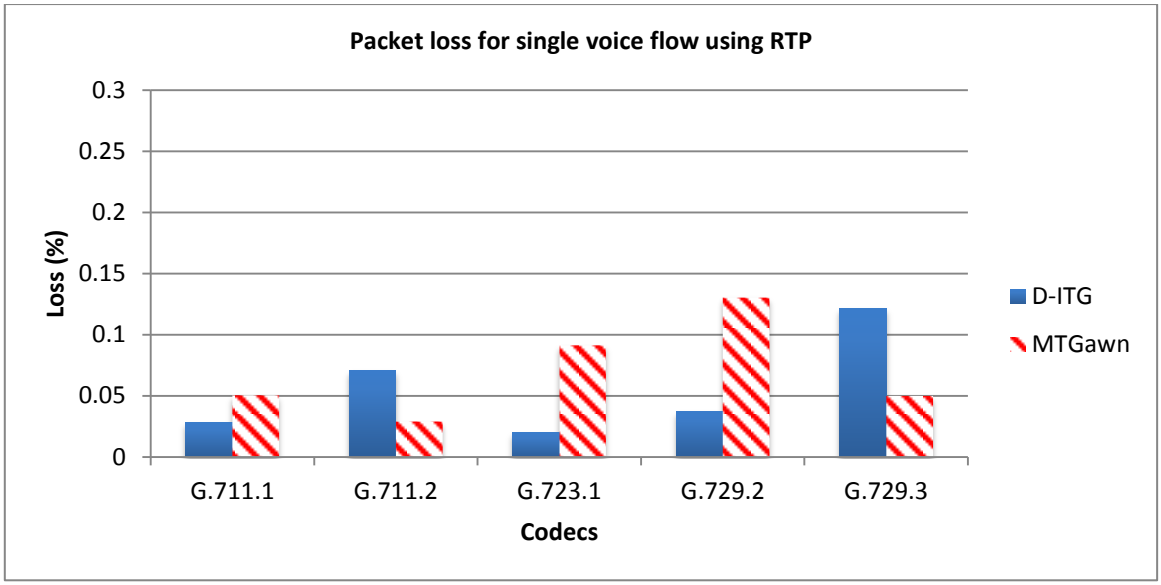


Figure 4.12: Packet loss for single voice flow using RTP

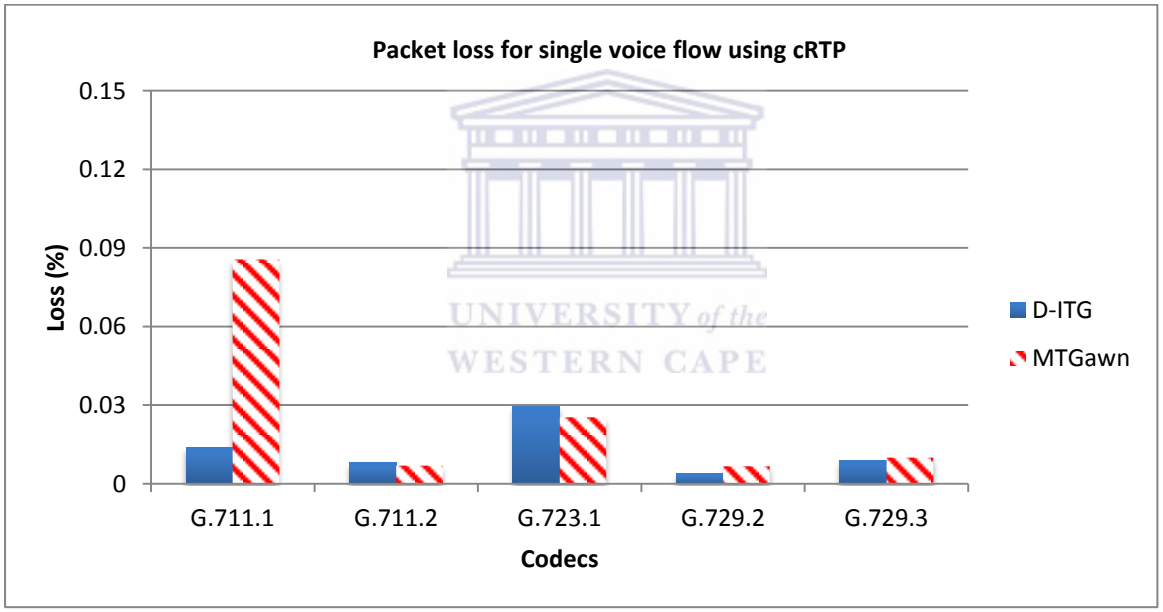


Figure 4.13: Packet loss for single voice flow using cRTP

iii. RTT delay

Figure 4.14 and Figure 4.15 illustrate the comparison between RTT delay in a single voice flow using different codecs in D-ITG and MTGawn. From this comparison, the delay was less for the D-ITG in almost all the cases. Therefore it will take less time to transmit and receive the voice from one destination to another across the network using D-ITG than using MTGawn.

For the RTP protocol, the highest delay in D-ITG was observed in G.711.1 codecs with a value of 49 ms and the lowest delay was observed with the G.729.2 codec at 16 ms, while the highest delay in MTGawn was observed in G.723.1 codec at 76 ms and the lowest delay was observed with the G.729.2 codec at 30 ms. For the cRTP protocol, the highest delay in D-ITG was observed in

G.723.1 codecs with a value of 66 ms and the lowest delay was observed with the G.729.3 codec at 11 ms while the highest delay in MTGawn was observed in G.711.1 codec at 74 ms and the lowest delay was observed with the G.729.2 codec at 24 ms.

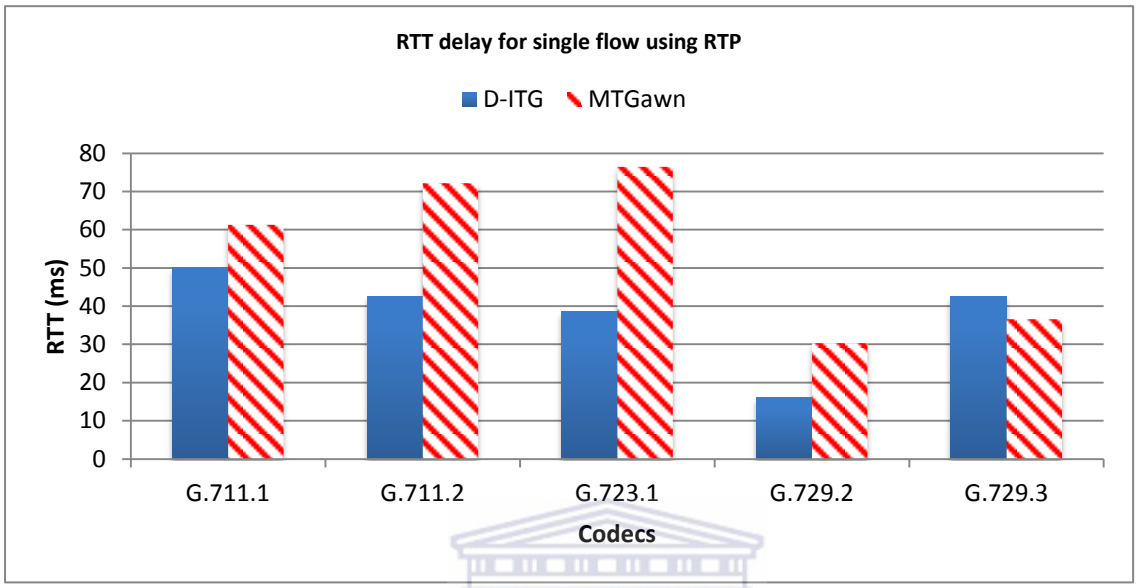


Figure 4.14: RTT delay for single flow using RTP

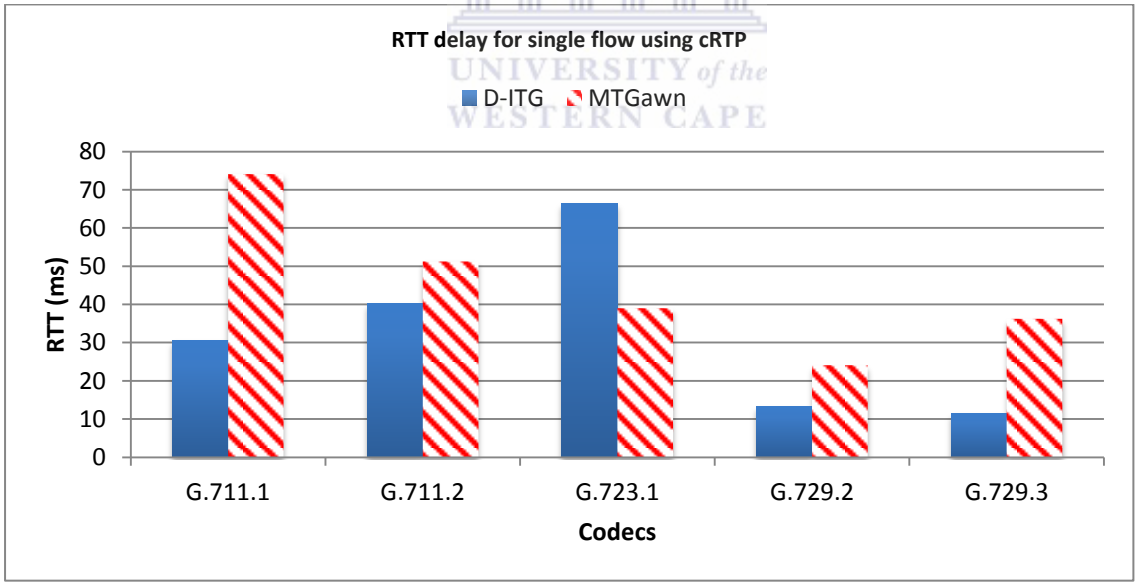


Figure 4.15: RTT delay for single flow using cRTP

iv. Jitter

Jitter is a very important metric especially when it comes to voice quality. A high variation in jitter causes damage to voice quality. Packet loss and jitter can be related since packets arriving too late to be of any use are discarded and this might cause a high variation of jitter. A jitter varies differently across all codecs. The highest jitter was obtained on the G.729.2 codec using RTP in MTGawn (it also corresponds to the highest packet loss) with a value of 3,89 ms. The lowest jitter was observed on the G.711.1 codec using cRTP in D-ITG with a value of 1,43 ms. For RTP, D-ITG had a lower jitter with G.711.1 and G.729.2 codecs while MTGawn had a lower jitter variation with G.711.2, G.723.1, G.729.3 codecs. Using cRTP, MTGawn had the lower jitter for all the codecs except for G.711.1 (see Figure 4.16 and Figure 4.17).

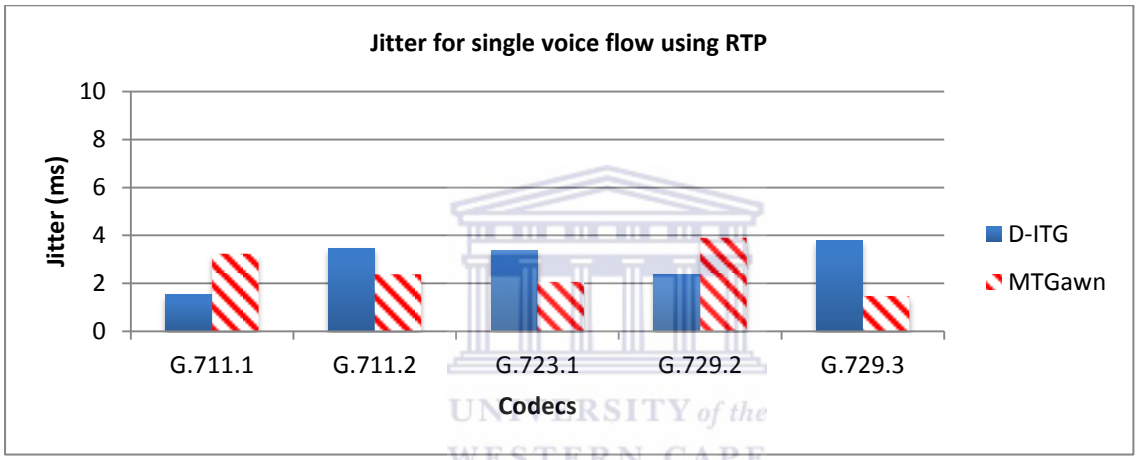


Figure 4.16: Jitter for single voice flow using RTP

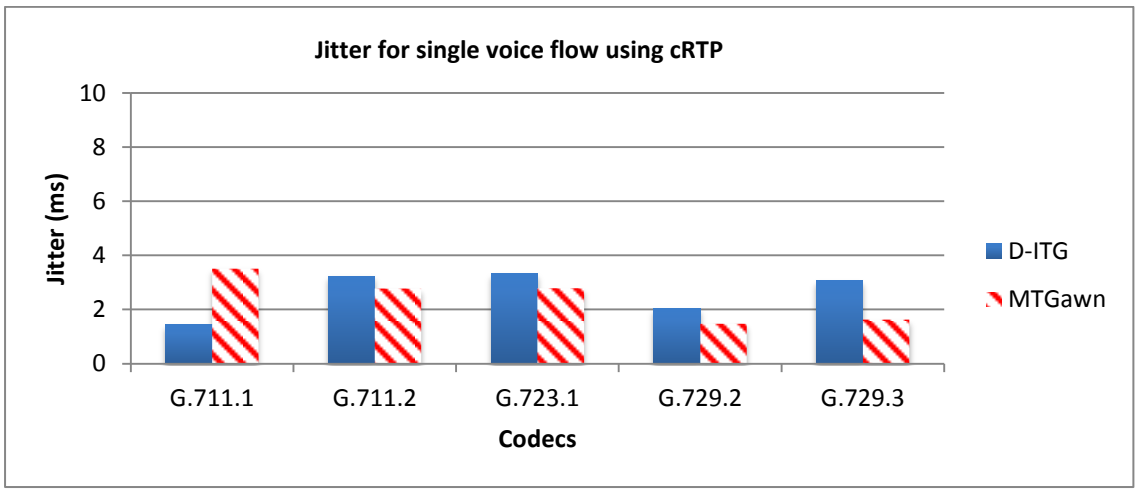


Figure 4.17: Jitter for single voice flow using cRTP

4.2 RESULTS FOR PROTOTYPE B: MULTIPLE FLOWS

The second DSR cycle lead to the design of MTGawn Prototype B, which allowed the sending, and receiving of multiple TCP, UDP and VoIP flows simultaneously in order to emulate multiple user activities concurrently. This DSR cycle was iterated in order to estimate the performance of the testbed WMN networks under more realistic conditions by giving the opportunity to the user to generate and transmit various traffic profiles. Each traffic profile was representative of real user behaviour. The purpose of this phase was to generate complex and varied TCP and UDP traffic sources representative of a wide range of traffic conditions. Different traffic patterns were produced to simulate different user activities. To test this prototype, two experiments were done in the laboratory testbed environment using different traffic patterns. The first experiment consisted of sending one UDP traffic flow and one VoIP traffic flow concurrently. The second experiment consisted of generating five voice flows concurrently over the network and collecting average results. Both experiments were done in D-ITG and MTGawn tools and the results obtained are reported in this section.

4.2.1 VoIP flow with UDP background traffic

The purpose of this experiment was to explore how background traffic could impact on VoIP QoS parameters. Two traffic patterns were defined:

- Single UDP traffic flow with constant packet size and constant packet rate distributions. Packet size was equal to 512 bytes and packet rate was 1000 Pkt/s.
- Single VoIP stream with five different codecs G.711.1, G.7.11.2, G.7.23.1, G.7.29.2 and G.7.29.3

Both traffics were simultaneously transmitted for the duration of 60 seconds and the performance of VoIP flow for each codec on both tools was monitored. Performance metrics of voice flow are reported below in terms of throughput, packet loss, delay and jitter.

i. Throughput

In Table 4.8, the throughput of the voice flow for each codec is given. From the obtained results, it is possible to note the decrease of throughput for each codec in comparison with single voice flow (as was discussed in Section 4.1.3). Background traffic loads had an impact on throughput for all three codecs on both D-ITG and MTGawn. From Table 4.8, it can be seen that D-ITG has both the highest bandwidth usage, which correspond to 71,51 Kbps using G.711.1 codec, and the lowest bandwidth usage with G.723.1 codec at 8,65 Kbps. From Table 4.9, it can be seen

MTGawn has both the highest bandwidth usage, which correspond to 63,33 Kbps using G.711.2 codec, and D-ITG has the lowest bandwidth usage with G.723.1 codec at 6,58 Kbps.

Table 4.8: Throughput for voice flow with UDP background traffic using RTP (in Kbps).

Codecs	D-ITG	MTGawn
G.711.1	71,514	70,415
G.711.2	64,932	66,999
G.723.1	8,650	8,715
G.729.2	12,637	12,042
G.729.3	10,924	11,001

Table 4.9: Throughput for voice flow with UDP background traffic using cRTP (in Kbps)

Codecs	D-ITG	MTGawn
G.711.1	61,227	60,629
G.711.2	60,920	63,333
G.723.1	6,589	6,653
G.729.2	8,665	8,600
G.729.3	8,024	8,403

ii. Packet loss

Figure 4.18 and Figure 4.19 show the packet loss variation using both RTP and cRTP. In this experiment, overall a higher variation of packet loss is observed compared to single voice flow. In fact, G.723.1 has the lowest loss with a percentage loss under 0,09 % for D-ITG and under 0,13 % for MTGawn. G.729.2 and G.729.3 loss is under 1%. Using RTP, the highest loss percentage is observed in D-ITG with G.711.2 codec at 1,88%. In MTGawn, a loss of 1,14% with G.711.2 and 1,27% with G.711.1 codec was observed. Using cRTP, the codec with the highest loss is G.711.1 in D-ITG with 1,33% and the codec with the lowest loss rate is G.723.1 in D-ITG at 0,09%.

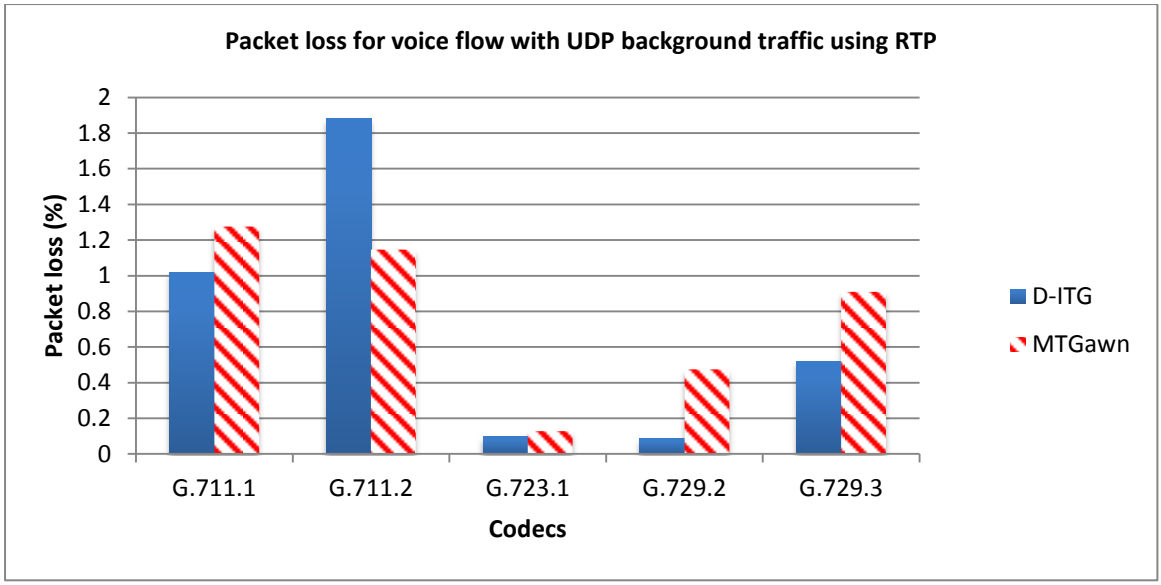


Figure 4.18: Packet loss for voice flow with UDP background traffic using RTP

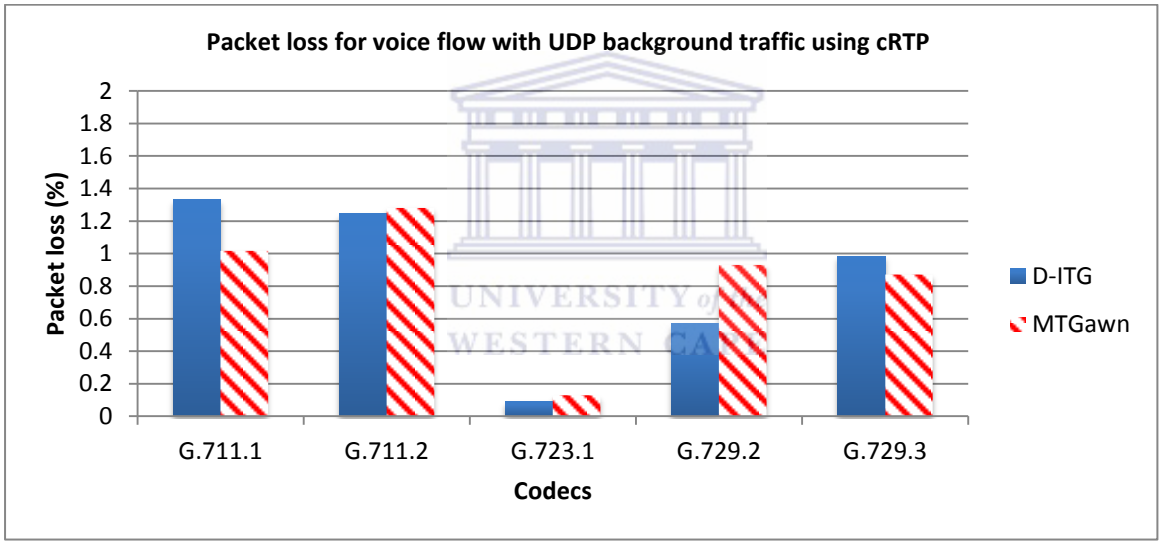


Figure 4.19: Packet loss for voice flow with UDP background traffic using cRTP

iii. RTT delay

Figure 4.20 and Figure 4.21 illustrate the RTT delay for VoIP flow with UDP background traffic using both RTP and cRTP. For MTGawn the RTT delay was under 123 ms for RTP, and under 94 ms for cRTP. For D-ITG the RTT delay was under 100 ms for RTP, and under 92 ms for cRTP. For both RTP and cRTP protocols, the highest delay is observed in MTGawn with G.711.1 codec. The lowest delay is observed in D-ITG using cRTP with the G.729.2 codec at 24 ms.

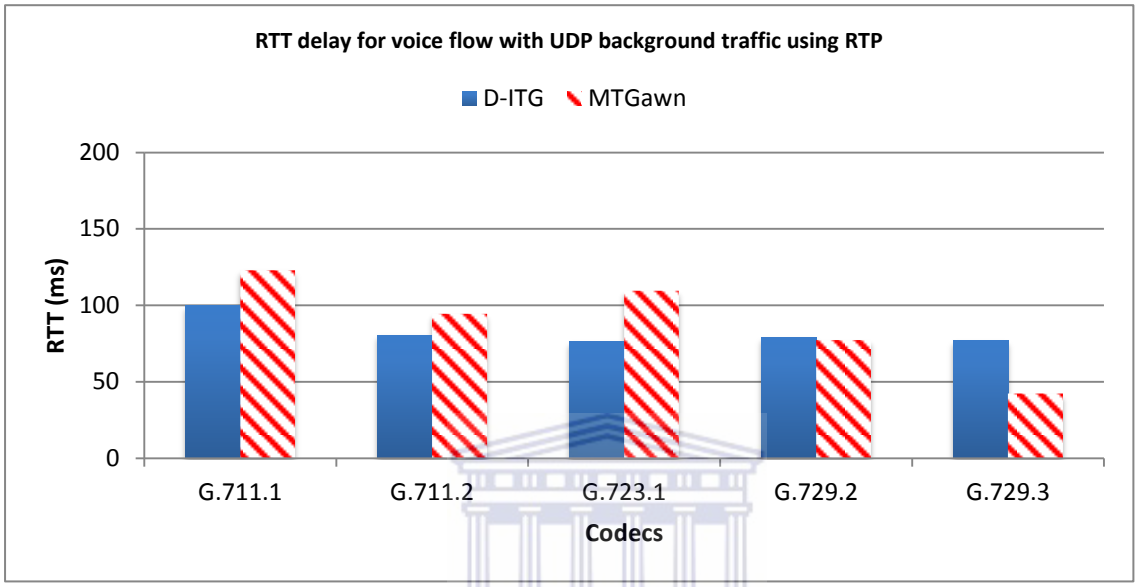


Figure 4.20: RTT delay for voice flow with UDP background traffic using RTP

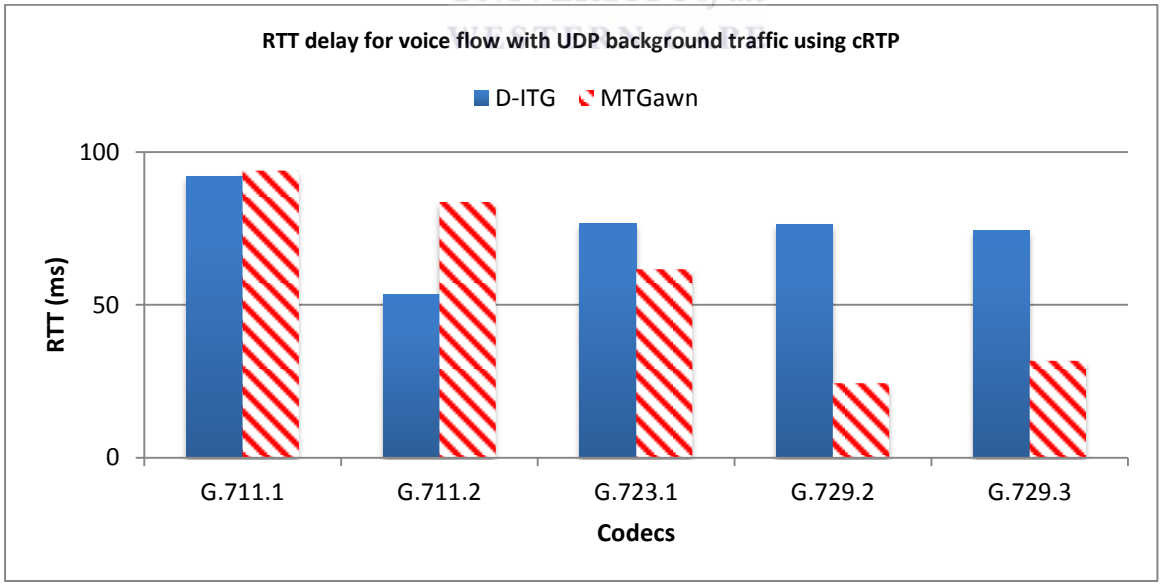


Figure 4.21: RTT delay for voice flow with UDP background traffic using cRTP

iv. Jitter

As in the previous section, the jitter varies differently across all codecs. The highest jitter was obtained on the G.723.1 codec using cRTP in D-ITG with a value of 5,7 ms (see Figure 4.23). The lowest jitter was observed on the G.711.1 codec using RTP in D-ITG with a value of 1,35 ms (see Figure 4.22). Using RTP, D-ITG has lower jitter with G.711.1 and G.711.2 codecs while MTGawn has lower jitter variation with G.729.2, G.723.1, and G.729.3 codecs. Using cRTP, D-ITG has the lower jitter for G.711.1 and G.729.2 and MTGawn has the lower jitter for the other codecs.

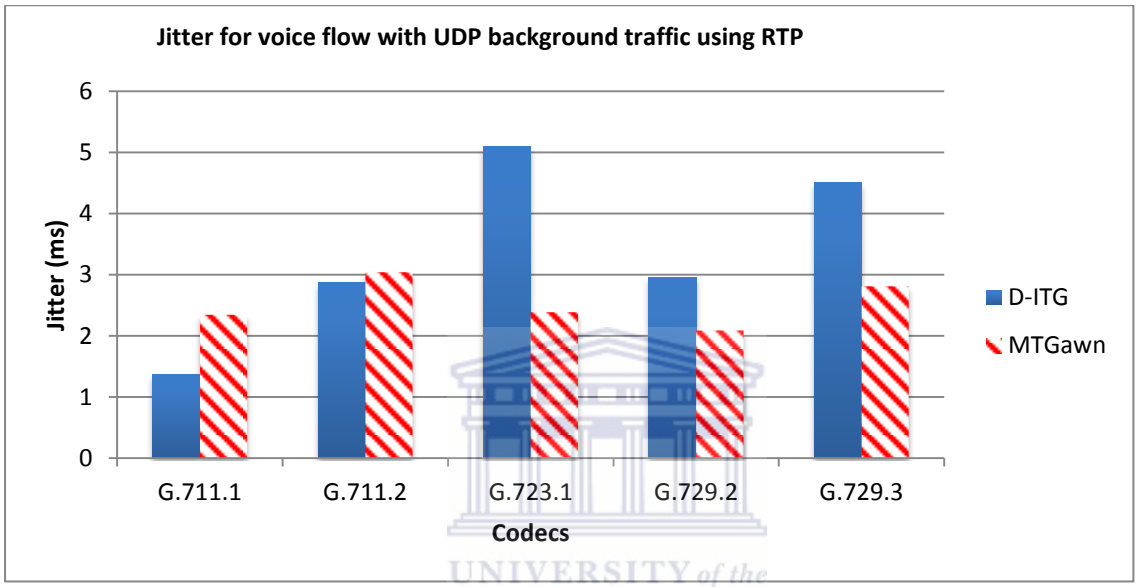


Figure 4.22: Jitter for voice flow with UDP background traffic using RTP

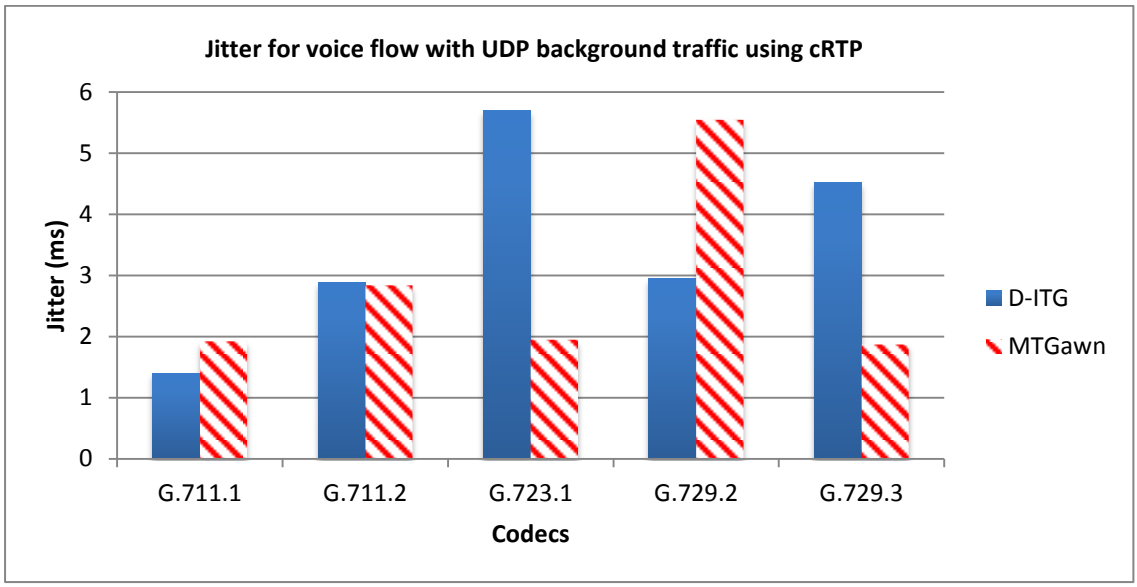


Figure 4.23: Jitter for voice flow with UDP background traffic using cRTP

4.2.2 Multiple VoIP flows

In this section, sending multiple flows simultaneously using voice traffic was tested. The aim was to discover the behaviour of the network when multiple voice flows are transmitted. The experiments entailed sending five flows simultaneously for every codec type, for D-ITG and MTGawn. Each voice flow lasted 60 seconds. For D-ITG, a script was used to send the five flows at the same time. The user interface of MTGawn allowed the selection of multiple flows and the sending of them simultaneously. For both tools, average results were obtained and compared in terms of throughput, packet loss, RTT delay and jitter.

i. Throughput

When sending multiple voice flows simultaneously, the same behaviour was observed as in section 4.1.3 meaning the codec with the highest frame rate has the highest throughput for both tools. From Table 4.10, it can be seen that D-ITG has the highest bandwidth usage for all the codecs especially for G.711.1 and G.711.2 codecs and that the highest throughput has a value of 360,21 Kbps and is obtained using G.711.1 codec in D-ITG. Moreover, the lowest throughput is obtained using G.723.1 codec in MTGawn, at approximately 42,81 kbps.

Table 4.10: Throughput for multiple VoIP flows using RTP (in Kbps)

Codecs	D-ITG	MTGawn
G.711.1	360,210	339,916
G.711.2	342,561	334,141
G.723.1	43,576	42,812
G.729.2	62,990	62,192
G.729.3	54,752	54,039

In Table 4.11 the highest throughput has a value of 325,44 Kbps and is obtained using G.711.1 codec with D-ITG. Moreover, the lowest throughput is obtained using G.723.1 codec with MTGawn, at approximately 32,61 kbps.

Table 4.11: Throughput for multiple VoIP flows using cRTP (in Kbps)

Codecs	D-ITG	MTGawn
G.711.1	325,447	304,947
G.711.2	322,810	300,284
G.723.1	33,051	32,6133
G.729.2	43,116	42,795
G.729.3	42,122	41,275

ii. Packet loss

Similar to section 4.2.1, packet loss is higher when multiple flows are sent at the same time. Overall, MTGawn has the higher packet loss using RTP for all codecs and lower packet loss for G.729.2 and G.729.3 codecs using cRTP. With the G.711.1 codec, it is observed that up to 5,04 % (which is the highest packet loss) packets are lost using RTP and up to 4,45% are lost when using cRTP in MTGawn. The lowest loss is approximately 0,31% and is experienced with D-ITG using RTP and G.723.1 codec. The next lowest is 0,42% using G.723.1 codec and cRTP in D-ITG.

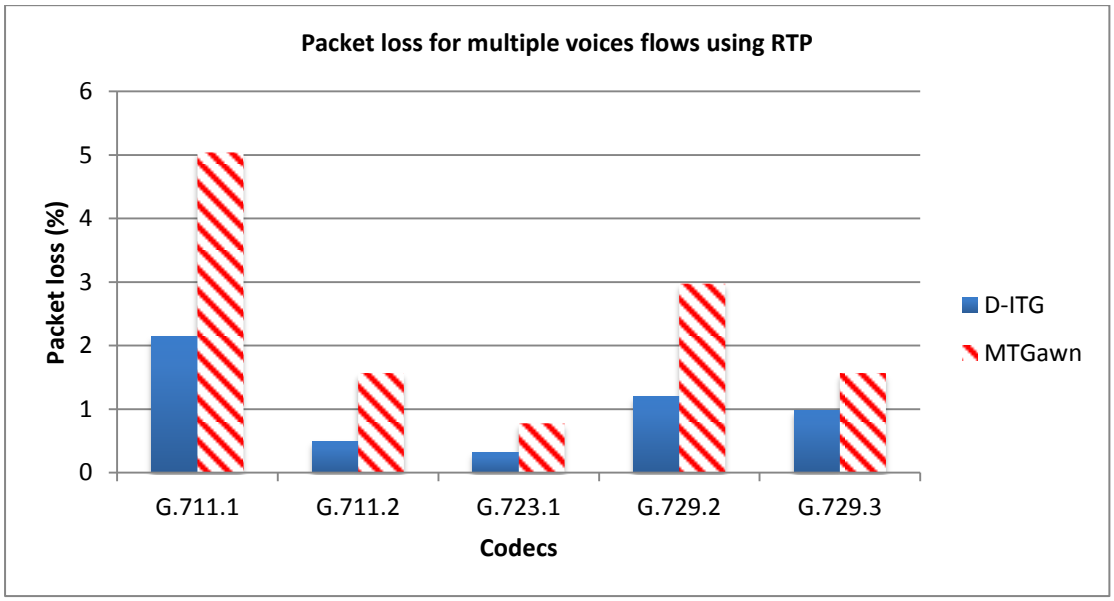


Figure 4.24: Packet loss for multiple VoIP flows using RTP

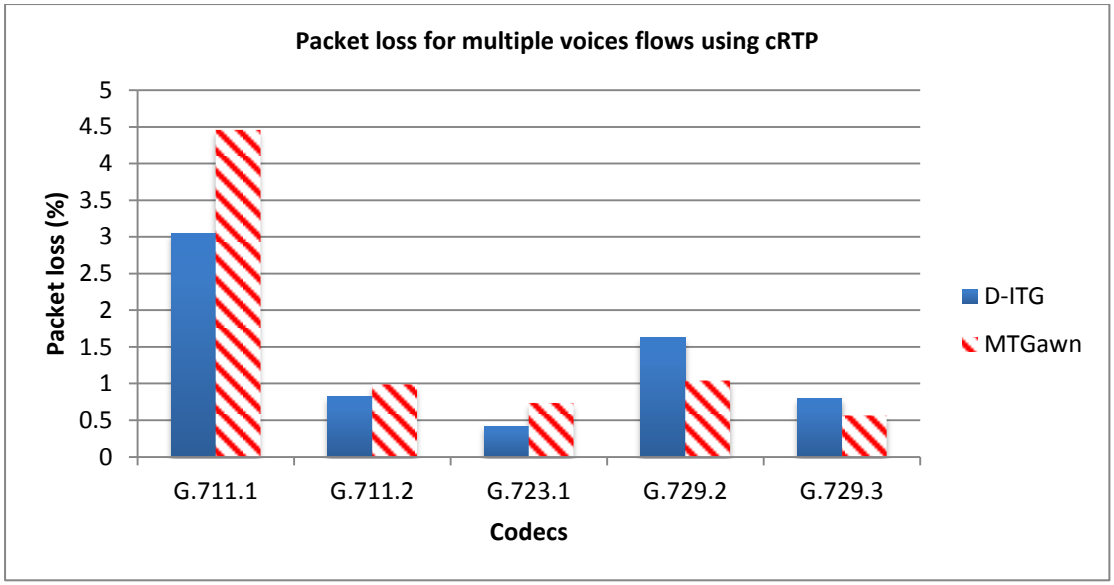


Figure 4.25: Packet loss for multiple VoIP flows using cRTP

iii. RTT delay

This section summarizes RTT delay variation for multiple voices. For MTGawn the RTT delay was under 114 ms for RTP and under 88 ms for cRTP. For D-ITG the RTT delay was under 95 ms for RTP and under 61 ms for cRTP. For RTP, the highest delay is observed in MTGawn with G.711.1 codecs and the lowest delay is observed in D-ITG with the G.711.2 codec at 32 ms (see Figure 4.26). For cRTP, the highest delay is observed in MTGawn with G.729.2 codecs and the lowest delay is observed in MTGawn with the G.723.1 codec at 21 ms (see Figure 4.27).

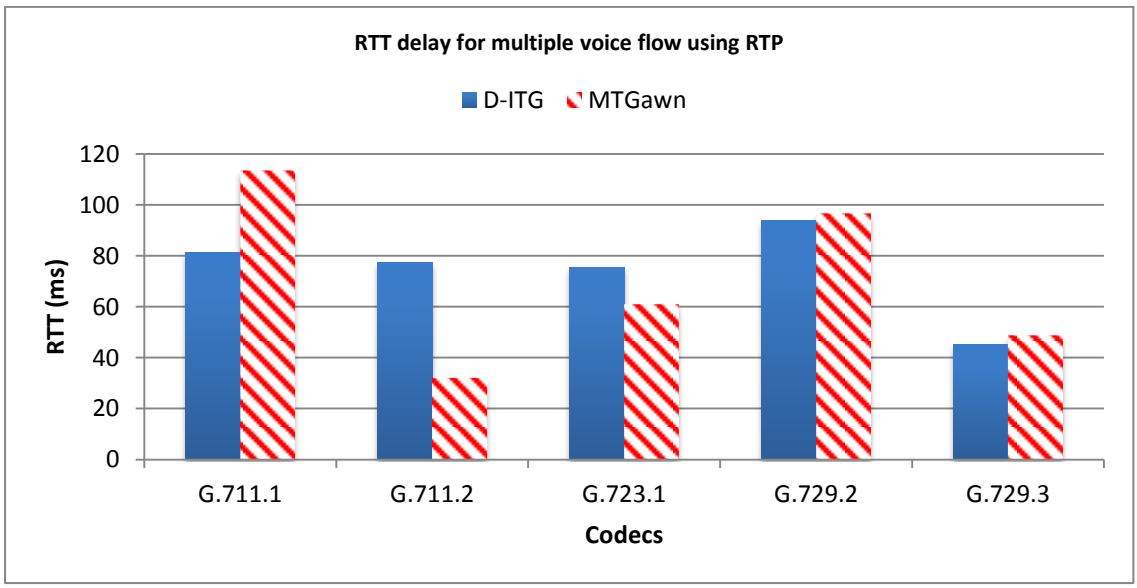


Figure 4.26: RTT delay for multiple VoIP flow using RTP

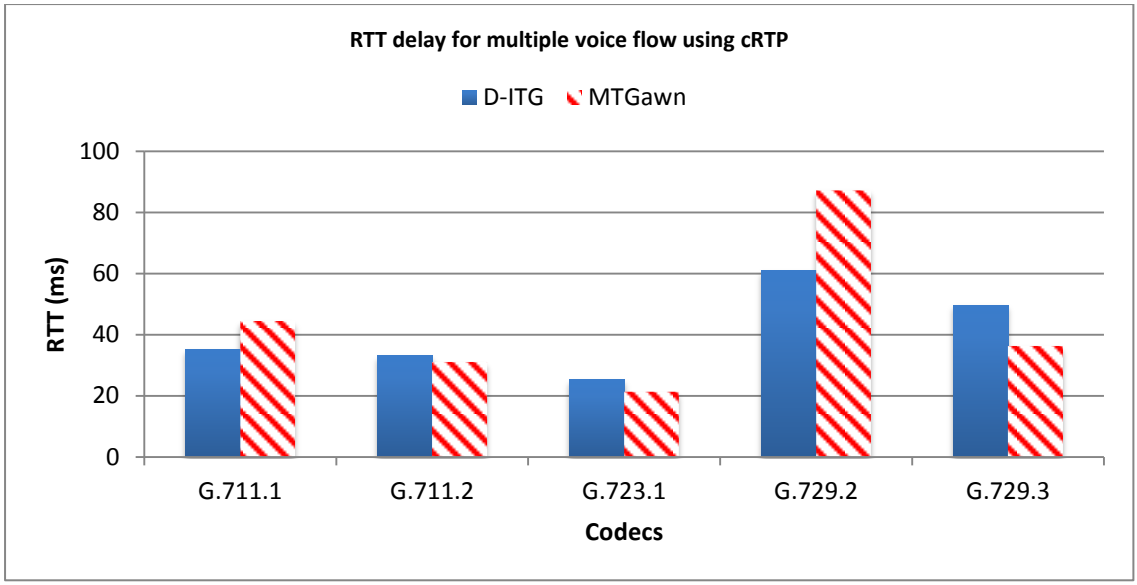


Figure 4.27: RTT delay for multiple VoIP flow using cRTP

iv. Jitter

This section summarizes jitter variation when multiple voice flows are transmitted at the same time over the wireless testbed network. The main observation is that both D-ITG and MTGawn present highest jitter with G.711.2 codec which is up to 5,77 ms for D-ITG and up to 5,38 ms for MTGawn. The lowest jitter is 1,46 ms and observed with G.723.1 codec using cRTP in MTGawn (see Figure 4.28 and Figure 4.29).

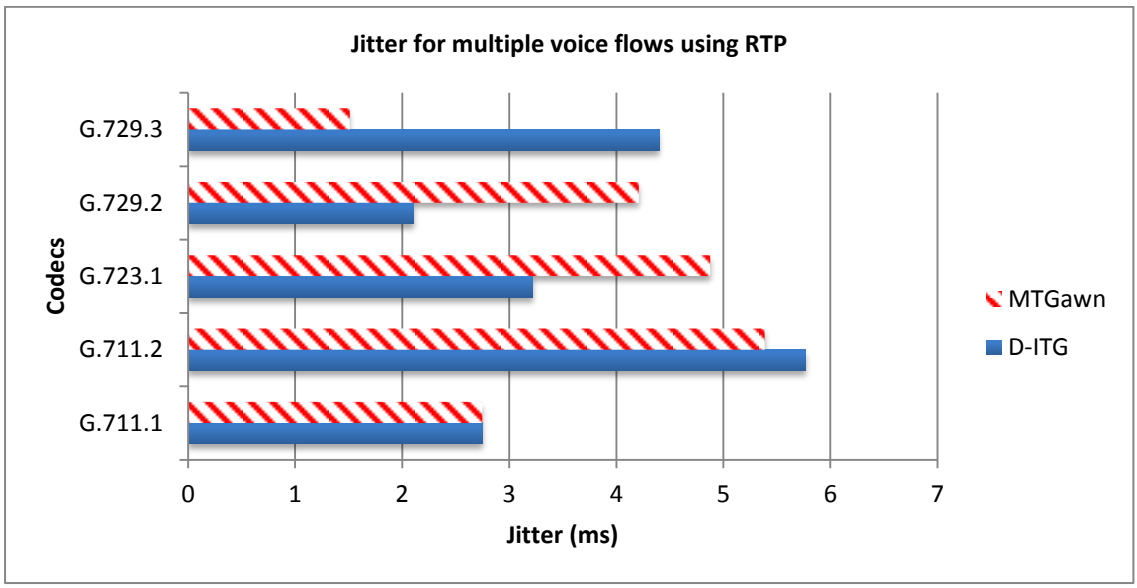


Figure 4.28: Jitter for multiple VoIP flows using RTP

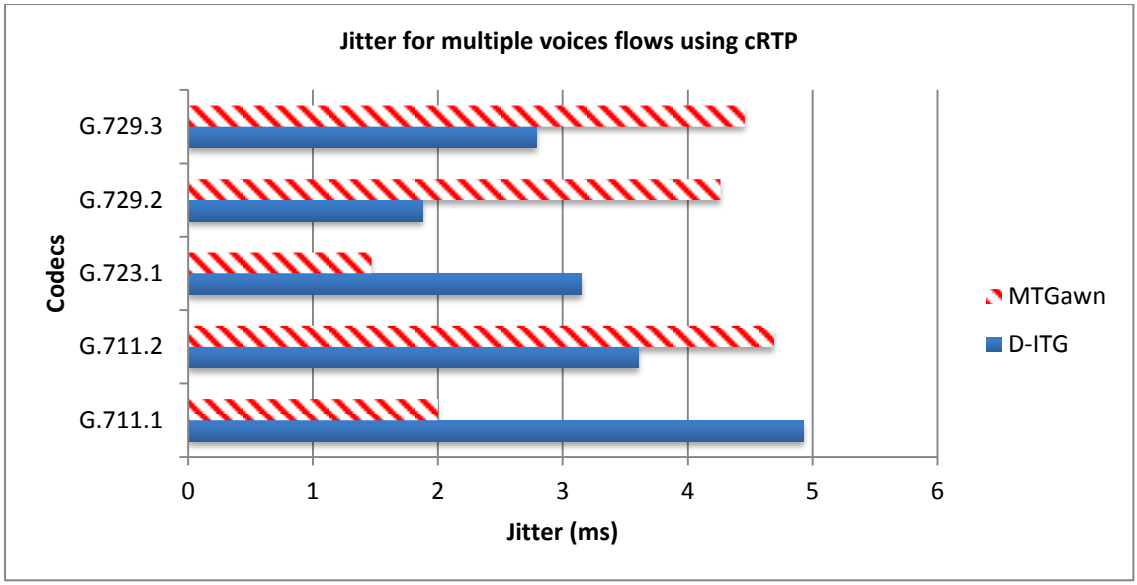


Figure 4.29: Jitter for multiple VoIP flows using cRTP

4.3 SUMMARY

In this chapter, results obtained while using the DSR methodology to design each prototype were summarized. The research was done in an iterative manner where results obtained during each phase were used in the next phase to solve the research question. The next chapter discusses the results and elaborate the main conclusions drawn from these results.

Chapter 5

DISCUSSION AND CONCLUSION

The aim of this research was to develop and implement a cost-effective tool that generates typical network traffic for the purpose of performance analysis. The tool was deployed on affordable and easily accessible devices such as mobile phones or tablets in order to ease performance measurement of Wi-Fi networks in poorly resourced areas. To do so, a mobile application or tool (MTGawn) that allows the generation of traffic flows for measurement, was developed. The research question addressed in this thesis was "How should a mobile traffic generator be designed to be feasible in order to analyse and evaluate the performance of wireless networks?"

This overarching question was unpacked into four sub-questions:

a. What techniques should be used for network monitoring and performance evaluation?
b. How should an appropriate traffic model be developed to emulate realistic traffic flows for mobile wireless networks?
c. How should the emulated traffic, over the Wi-Fi interface of a mobile device, be generated?
d. What performance metrics should be taken into consideration?

The previous chapter presented the results obtained while testing MTGawn. Several experiments were conducted to evaluate UDP, TCP and VoIP performance using both MTGawn and a PC-based tool, D-ITG. The results achieved in both cases were compared using the same parameters (in laboratory testbed experiments). In this chapter, these findings are discussed in terms of the research questions. To conclude, difficulties encountered during the design and testing of MTGawn are discussed as well as the limitations of the tool are mentioned. Some suggestions on how the tool can be improved, in order to be more useful, are made.

5.1 FINDINGS IN TERM OF THE RESEARCH QUESTIONS

What techniques should be used for network monitoring and performance evaluation?

The technique used to address the first sub-question was active monitoring, which involved modelling and generation of network traffic flows using traffic generator software. A traffic

generator was used to emulate synthetic but realistic traffic flows. The flows were then injected into the network using a mobile device and captured at the end of the network using another mobile device, to extract QoS metrics of interest. The main difficulty with this approach was the choice of appropriate traffic models that accurately represented realistic traffic flows. Numerous traffic models were proposed in the literature review and the choice of the suitable model depended upon many parameters, such as the type of traffic to be generated (UDP, TCP, VoIP…. etc.).

How should an appropriate traffic model be developed to emulate realistic traffic flows for mobile wireless networks?

Probability distributions and a Markov process (in particular the ON–OFF model) were used for the emulation of realistic traffic flows, thus addressing the second sub-question. Packet size and packet inter-arrival time (packet rate) were the two important parameters that needed to be emulated. In this thesis, three types of flow were emulated: UDP, TCP and VoIP. For UDP and TCP, these two parameters were represented as random variables and were characterized by a probability distribution (see Appendix A). For VoIP, the size of the packets generated and the rate at which the packets were transmitted were related to the codec in use, since each codec has its predefined standard parameters.

How should the emulated traffic, over the Wi-Fi interface of a mobile device, be generated?

For the third sub-question a Java implementation of the UDP and TCP transport protocols was used to generate realistic flows and transmit and capture the packets on an Android mobile device. TCP packets were transmitted and received using Sockets. UDP packets were created using the DatagramPacket and sent using a DatagramSocket. VoIP packets were encapsulated in UDP packets and transmitted using a DatagramSocket.

What performance metrics should be taken into consideration?

For the measurement, the final sub-question, the performance metrics chosen were throughput, packet loss, delay and jitter.

The results shown in Chapter 4 indicate that MTGawn was comparable to D-ITG in terms of achieved throughput for all the experiments. Overall, D-ITG shows a slightly better performance than MTGawn in respect of achieved throughput. As far as packet loss was concerned, both tools for the most part, presented an increase of packet loss as the packet rate and packet size increased.

It was also observed that the higher the packet rate the higher the loss rate for both UDP and TCP in both D-ITG and MTGawn. The loss rate also increases as the packet size increased.

When comparing achieved throughput for a single VoIP flow with different codecs, the best throughput of the codecs was for G.711.1 and the worst throughput was observed in G.723.1 across the two tools. Similar results were obtained when sending a VoIP flow with UDP background traffic and when transmitting five voices flows simultaneously. An impact of background traffic load on QoS parameters of voice services was observed: overall, the achieved throughput of each codec was lower but the packet loss, the delay and the variation in delay (Jitter) were increased for both tools.

Delay and jitter varied for each experiment. The smaller the delay, the quicker it was for a packet to reach its destination. Delay and jitter are very important especially in UDP based applications (such as VoIP), which are very sensitive to high delay variation. Overall, and for both tools, when sending a single TCP flow, a higher delay was observed than for a single UDP flow. This high variation of delay for TCP as compared to UDP might be due to the packet fragmentation, retransmission and reordering typical of TCP.

Similarly to UDP and TCP flows, delay and jitter were very different and variable across all the codecs for all VoIP flows in both D-ITG and MTGawn. In general delay and variation in delay are caused by network congestion and high packet drop ratio. However, encapsulating each packet at the sender side and de-capsulating the packet at the receiver side also caused an increase in the delay. In addition, delay and jitter were also affected by the protocol used for voice transmission (RTP or cRTP) thus the choice of appropriate codecs varied for different situations. Based on the results obtained, it might be concluded that sending multiple flows at the same time increased the packet loss, delay and jitter as compared to single flows. This observation is reasonable even in a real-life scenario, where the growth of users using the network leads to the growth of traffic flowing through the network. This growth in traffic load generally results in non-negligible service delay and packet loss.

5.2 DIFFICULTIES ENCOUNTERED

Some problems encountered during the design and testing of MTGawn include:

- Network failure and difficulties to recover from errors within the application: if an error occurred during the transmission, it resulted in system crashes and the application closing

automatically. Some of these errors included: one of the hosts loses its connection; the port is already in use; connection exception; failure to connect to Wi-Fi interface…. etc.)

- For reasons of simplicity and because of time constraints, time synchronisation was not considered within the MTGawn application. To address synchronisation problems, the same network provider was used for the SIM card (inserted in the phones) with the assumption that the time (determined by the SIM card network provider) would be the same on both the mobile sender and receiver phones. However, this did not work since most of the one-way delays were still either too high or negative, the experiments were repeated several times and averaged to get useful results and the round trip time was considered.

- Fragmentation and retransmission of packets in TCP scenario: TCP is not a packet protocol but rather a stream based protocol. Thus, it combines many packets to maximize the size of the IP packet for the given maximum transmission unit (MTU). As the packet traverses each Ethernet device, it might be fragmented if its size is bigger that the MTU of that Ethernet device. Consequently, even though no losses were experienced (since most packet losses are retransmitted), the number of packets sent and received differed considerably and this had an impact on the achieved results.

5.3 RESEARCH CONTRIBUTION

The tool is intended to ease feasibility testing and performance evaluation of a rural wireless network by implementing a 'stripped down' version of a packet generation and monitoring system to a mobile platform with the functionality found in common open source tools. The tool was compared to an open source tool D-ITG. Overall, MTGawn showed similar results to D-ITG for the same traffic profiles. This suggests it can work in the field, on a mobile device.

The fundamental contribution of this work was the design of a traffic generator that is able to run on a mobile device and has the ability to emulate realistic traffic scenarios and provide standard QoS metrics such as throughput, delay, jitter and packet loss. Hence, the tool was deployed with an easy to use interface and it could allow network administrators to keep track of their network even in disastrous scenarios, such as load shedding, since mobile phones can be charged using solar panels or a car battery. Moreover, mobile phones are essential tools for worldwide communications due to their ultra-portable size, their convenience and their relative affordability, which makes them suitable in a variety of scenarios because of the ease and feasibility of mobile applications.

5.4 CONCLUSIONS

This research involved the development of a prototype for a mobile traffic generator called MTGawn. The research was completed in three phases: The first phase involved the *problem identification* in which the research questions were formulated and the objectives of the research were clarified. The second phase involved the *prototype design and presentation* in which DSR methodology was applied to guide the design and the third phase involved the *prototype evaluation and report*. This evaluation was performed in a testbed wireless mesh network and the results obtained were compared to a well known PC-based tool, namely D-ITG.

Different UDP, TCP and VoIP traffic flows were transmitted over a wireless mesh network under diverse traffic conditions—many packet sizes and packet transmission rate values were combined to create different traffic conditions and in some case, many flows were transmitted simultaneously. This was done to highlight the scenario of multiple applications running concurrently over a wireless link. The pieces of equipment used in the test-bed were different in terms of software and hardware. This situation needs to be taken into account when clarifying the conditions in which MTGawn performs better than D-ITG or vice versa.

Even though D-ITG shows slightly better performance in most cases, MTGawn has proven to be useful since it provides similar and comparable results. The difference between the results in the two tools might be based on several factors including:

- The programming language: MTGawn was programmed in Java while D-ITG was programmed in C. Both programming languages have their own ways to handle threads, memory, resources allocation, files…etc., which might have influenced the results.

- The hardware and software equipment used for the experiments: D-ITG was deployed on a computer and MTGawn was deployed on a mobile device with limited computing power. The mobile phones and the computers used during the experiments were distinctive in term of hardware and software**.** D-ITG traffic sender was a computer running Ubuntu 14.04 (trusty) and D-ITG traffic receiver was a computer running Ubuntu 12.04 (precise). MTGawn traffic sender was a mobile device running Android OS v4.2.2 (Jelly Bean) and MTGawn traffic receiver was a mobile device running Android OS v4.0.3 (Ice Cream Sandwich). (This could also have had an influence on the results)

5.5 FUTURE WORK

MTGawn is just a subset of D-ITG thus it does not have all the functionalities nor the capabilities that D-ITG has. Furthermore, MTGawn has limitations: the tool was only deployed for mobile phones running Android OS and it offers three types of traffic flows (TCP, UDP and VoIP) as well as their mixture.

However, in future research the tool can be improved by adding more functionalities and looking at techniques to improve the performance of MTGawn considering the limited computing power of mobile devices. The following are some suggestions to extend and improve MTGawn:

- Testing the tool using different Android devices with different screen sizes and different OS versions to see how the tool behaves under different hardware requirements. By sending a large number of flows and repeating the same experiments several times, more accurate results could be achieved. And comparing these statistically.

- Testing the tool considering cases of heavy and light load conditions on the network and their performance measurements. This will be useful to determine how the quality of service is affected with increasing number of mobile device connections.

- Emulating voice traffic using other types of codecs, and including more samples for each codec.

- Modelling and generating other types of flow(s) such as games, DNS, HTTP, video, etc. using the implementation of UDP and TCP traffic flows and the probability distributions of this thesis.

- The tool could be expanded to take into consideration other important network parameters such as the time to live (TTL), type of service (TOS), etc.

- Consideration could be given to ways to improve the scalability and portability of the tool so that it could be used on any mobile device independently of the OS (iOS, windows phone, etc.)

- In-the-field testing on an actual rural wireless network, would be advisable, to assess how the mobile prototype behaves under real physical conditions, and comparing these results against other standard PC-based performance tool such as D-ITG, to see the usefulness of the tool in a real life scenario.

- Considering the problem of interference on normal traffic when flows are injected in the network, in-the-field testing will also be useful in order to study the effect or impact of synthetic traffic on normal traffic and provide helpful suggestions to address this issue.

BIBLIOGRAPHY

Achmed, I., Venter, I., & Eisert, P. (2013). *Independent hand-tracking from a single two-dimensional view and its application to South African sign language recognition.* University of the Western Cape.

Adas, A. (1997, July). Traffic Models in Broadband Networks. *IEEE Communications Magazine* , pp. 82-89.

Aktas, I., Schmidt, F., Weingärtner, E., Schnelke, C.-J., & Wehrle, K. (2012). An Adaptive Codec Switching Scheme for SIP-based VoIP. *Internet of Things, Smart Spaces, and Next Generation Networking*, 347-358.

Alvarion. (2006). *Implementing VoIP Service Over Wireless Network.* Alvarion Technologies.

Andreev, S., Anisimov, A., Koucheryavy, Y., & Turlikov, A. (2010). Practical Traffic Generation Model for Wireless Networks. *Fourth ERCIM Workshop on Emobility*, (p. 61).

Avallone, S., Guadagno, S., Emma, D., Pescape, A., & Ventre, G. (2004). D-ITG Distributed Internet Traffic Generator. *First International Conference on the Quantitative Evaluation of Systems* (pp. 316-317). IEEE Computer society.

Avallone, S., Pescape, A., & Giorgio, V. (2004). Analysis and experimentation of internet traffic generator. *proceedings of NEW2AN*, (pp. 70-75).

Avallone, S., Pescape, A., & Ventre, G. (2003). Distributed Internet Traffic Generator (D-ITG): analysis and experimentation over heterogeneous networks. *International Conference on Network Protocols (ICNP Poster Proceedings).* Atlanta, Georgia.

Bahari, S. F. (2010). Qualitative versus Quantitative research strategies: contrasting epistemological and ontological assumptions. *Jurnal Teknologi, 52*(1), 17-28.

Balakrishnan Chandrasekaran. (2009). *Survey of Network Traffic Models.* wustl.edu.

Barbosa, R., Kamienski, C., Mariz, D., Callado, A., Fernandes, S., & Sadok, D. (2007, June). Performance evaluation of P2P VoIP applications. *ACM NOSSDAV, 7*.

Bitweaver. (2003). Retrieved 2015, from VOIP-Info.org: http://www.voip-info.org/wiki/view/RTP+Silence+Suppression

Bourne, M. (2014, March 23). *Counting and Probability* . Retrieved from Interactive mathematics: http://www.intmath.com/counting-probability/13-poisson-probability-distribution.php

Brocke, J. V., & Buddendick, C. (2006). Reusable conceptual models–requirements based on the design science research paradigm. *First International Conference on Design Science Research in Information Systems and Technology(DESRIST 2006)*, (pp. 576-604). Claremont.

Cai, Y., Wolf, T., & Gong, W. (2011, May). Delaying Transmissions in Data Communication Networks to Improve Transport-Layer Performance. *Journal on selected areas in communications, 29*(5), 916-927.

Camacho, A. A., Laner, M., Svoboda, P., & Rupp, M. (2011). *Analysis and interpretation of emulated data traffic in Android platform.* Barcelonna: Vienna University of Technology.

Cecil, A. (2012). A Summary of Network Traffic Monitoring and Analysis Techniques. *Conference on Instruction & Technology (CIT)*, (pp. 10-25).

Chen, T. M. (2007). Network traffic modeling. In H. Bidgoli, *The handbook of computer networks.*

Cheng, X., Mohapatra, P., Lee, S.-J., & Banerjee, S. (2008). Performance evaluation of video streaming in multihop wireless mesh networks . *Proceedings of the 18th International Workshop on Network and Operating Systems Support for Digital Audio and Video* (pp. 57-62). New York: ACM.

Cheng, Y., Cetingaya, E., & Sterbenz, J. (2013). Transactional traffic generator implementation in ns-3. *Information and telecommunication technology centre*.

Choi, H. K., & Limb, J. O. (1999). A Behavioral Model of Web Traffic. *Proceedings on the 7th IEEE International Conference on Network Protocols (ICNP'99)* (pp. 327-334). IEEE.

Christiansen, T., Giotis, I., & Mathur, S. (n.d.). *Performance Evaluation of VoIP in Different Settings.*

Cisco . (2006, February 02). *Voice Quality Voice Over IP - Per Call Bandwidth Consumption*. Retrieved January 30, 2014, from Cisco Support Community : http://www.cisco.com/c/en/us/support/docs/voice/voice-quality/7934-bwidth-consume.html

Clark Science Center. (n.d.). Computer Networks and Internet - Overview. In *Computer Networking and Management* (pp. 1-44).

Collis, J., & Hussey, R. (2009). *Business research: A practical guide for undergraduate and postgraduate students*. Palgrave Macmillan.

Cottrell, L. (2001, March 11). *Passive vs Active Monitoring*. Retrieved June 20, 2014, from http://www.slac.stanford.edu/comp/net/wan-mon/passive-vs-active.html

Crotty. (1998). *The foundations of social research: Meaning and perspective in the research process*. Sage Publications Ltd.

Crotty, M. (1998). *The foundations of social research: Meaning and perspective in the research process*. Sage Publications Ltd.

Deng, S. (1995). Traffic characteristics of packet voice. *International Conference on Communications (ICC '95) . 3*, pp. 1369 - 1374 . Seattle, WA : IEEE .

Deri, L. (2004). Improving passive packet capture: Beyond device polling. *Fourth International System Administration and Network Engineering Conference (SANE 2004)*, (pp. 85-93). Amsterdam.

Eindhoven University of Technology. (2012-2014). Retrieved from http://chi.se.wtb.tue.nl/reference-manual/distributions.html

Fang, Y. (2005). Performance evaluation of wireless cellular networks under more realistic assumptions. *Wireless Communications and Mobile Computing, 5*(8), 867-885.

Flowers, P. (2009). *Research Philosophies – Importance and Relevance*. Cranfield School of Management.

Fras, M., Mohorko, J., & Cucej, Z. (2012). Modeling and Simulating the Self-Similar Network Traffic in Simulation Tool. *Telecommunications Networks - Current Status and Future Trends*.

Friesen, J. (2013). *Learn Java for Android Development*. Springer.

Garcia, L. M. (2010). *TCPDump & LIBPCAP*. Retrieved 06 20, 2014, from http://www.tcpdump.org/

Geerts, G. L. (2011, June). A design science research methodology and its application to accounting information systems research. *International Journal of Accounting Information Systems, 12*(2), 142–151.

Golder, M., & Golder, S. (n.d.). *Random Variables and Probability Functions*. Pennsylvania State University OBJ OBJ OBJ .

Hasib, M. (2006). *Analysis of Packet Loss Probing in Packet Networks*. Queen Mary, University of London, Department of Electronic Engineering.

Hassan, H., Garcia, J.-M., & Brun, O. (2005). *Generic modeling of multimedia traffic sources* . Laboratory for Analysis and Architecture of Systems .

Heo, J. (2011). *Voice over IP (VoIP) Performance Evaluation on VMware vSphere®5* . Palo Alto CA: VMware.

Husanu, A., & Trifan, A. (2014). Labelling spam through the analysis of protocol patterns. *Virus Bulletin Conference (VB2014)* (pp. 189-194). Seattle: Virus Bulletin.

Huston, G. (2003). Measuring IP Network Performance. *The Internet Protocol Journal, 6*(1), 2-19.

Iperf. (2008). Retrieved from http://iperf.sourceforge.net/

JSDU. (2006). *VoIP overview*. JSDU uniphase corporation.

Landauer, J., & Rowlands, J. (2001). *Subjectivism*. Retrieved November 7, 2014, from Importance of philosophy: http://www.importanceofphilosophy.com/Irrational_Subjectivism.html

Lee, I. W., & Fapojuwo, A. O. (2005). Stochastic processes for computer network traffic modeling. *Computer Communications, 29*(1), 1-23.

Lee, J. J., & Gupta, M. (2007). *A New traffic model for current user web browsing behavior*. Intel Corporation.

Loadfocus. (2015). *What is Throughput in Performance Testing?* Retrieved 2015, from Cloud Testing Platform – Load Testing, API Monitoring and Website Speed Testing: https://loadfocus.com/blog/2013/07/what-is-throughput-in-performance-testing/

Marsic, I. (2010). *computer networks:performance and quality of service*. New Jersey: Rutgers University (The State University of New Jersey).

Mohammed, A. M., & Agamy, A. F. (2011). A Survey on the Common Network Traffic Sources Models. *International Journal of Computer Networks (IJCN), 3*(2), 103-115.

Mundra, S., & Hernandez, C. E. (2004). *Patent No. 20040032860*. USA.

Newport Networks . (2005). *VoIP Bandwidth Calculation*. Newport Networks Ltd .

Nicola, B., Giordano, S., Procissi, G., & Secchi, R. (2005). BRUTE: A High Performa n ce and Extensible Traffic Generator. *In Proceedings of SPECTS*, (pp. 839-845). Philadelphia, USA.

Oracle. (2009). *Getting started*. Retrieved November 3, 2014, from docs.oracle.com: http://docs.oracle.com/cd/E14269_01/em.451/e12487/getting_started_reworked.htm

Orebaugh, A. (2007). *Wireshark & Ethereal Network Protocol Analyzer Toolkit*. canada: Syngress publishing inc.

Orebaugh, A., Ramirez, G., Burke, J., Morris, G., Pesce, L., & Wright, J. (2007). *Wireshark & Ethereal Network Protocol Analyzer Toolkit*. Canada: Syngress Publishing.

Ostermann, S. (2000). *Tcptrace official homepage*. Retrieved from Tcptrace: http://www.tcptrace.org/

Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2008). A design science research methodology for information systems research. *Journal of Management Information Systems , 24*(3), 45-78.

Rajasekar, S., Philominathan, P., & Chinnathambi, V. (2006). *Research methodology*.

Rey-Moreno, C., Roro, Z., Tucker, W. D., Siya, M. J., Bidwell, N. J., & & Simo-Reigadas, J. (2013). Experiences, challenges and lessons from rolling out a rural WiFi mesh network. *Proceedings of the Third ACM Symposium on Computing for Development* (p. 11). ACM.

Rubin, H. J., & Rubin, I. S. (2012). Research philosophy and qualitative interviews. In H. J. Rubin, & I. S. Rubin, *Qualitative Interviewing: The Art of Hearing Data* (pp. 13-24). Sage.

Siegrist, K. (1997-2014). *Special Distributions: Bernouilli Trials*. Retrieved October 20, 2014, from Virtual Laboratories in Probability and Statistics: http://www.math.uah.edu/stat/bernoulli/Geometric.html

Siegrist, K. (1997-2014). *Special Distributions: The Normal Distribution*. Retrieved October 20, 2014, from Virtual Laboratories in Probability and Statistics: http://www.math.uah.edu/stat/special/Normal.html

Sommers, & Barford. (2004).

Sommers, Kim, & Barford. (2004).

Sriram, K., & Whitt, W. (1986, September). Characterizing superposition arrival processes in packet multiplexers for voice and data. *Journal on Selected Areas in Commun , 4*(6), 833-846.

Stefano Avallone, A. P. (2004). Analysis and experimentation of Internet Traffic. *NEW2AN*, 1-6.

Techopedia. (2010). *Home>Dictionary>Tags>Hardware*. Retrieved 2015, from techopedia.com: http://www.techopedia.com/definition/5573/throughput

TechTarget. (2000). *Home > Network Design > Computing fundamentals Glossary > throughput definition*. Retrieved 2015, from SearchNetworking.com: http://searchnetworking.techtarget.com/definition/throughput

TechTarget. (2008). Retrieved 2015, from searchunifiedcommunications.techtarget.com: http://searchunifiedcommunications.techtarget.com/definition/voice-activation-detection

TechTarget. (2009). *Home > Bandwidth and capacity planning > Data transmission Glossary > bandwidth definition*. Retrieved 2015, from SearchEnterpriseWAN.com: http://searchenterprisewan.techtarget.com/definition/bandwidth

Telchemy. (2006). *VoIP Performance Managemen:Impact of Delay in Voice over IP Services*. Telchemy.

Varga, P., & Olaszi, P. (2013). LTE core network testing using generated traffic based on models from real-life data. *7th IEEE International Conference on Advanced Networks and Telecommunication Systems (ANTS)* (pp. 1-6). Chennai, India: IEEE.

Veiga, H., Pinho, T., Oliveira, J. L., Valadas, R., Salvador, P., & Nogueira, A. (2005). Active traffic monitoring for heterogeneous environments. *Fourth International Conference on Networking (ICN 2005)* (pp. 603-610). Springer-verlag Berlin Heidelberg.

Walsh, T. J., & Kuhn, R. D. (2005). Challenges in Securing Voice over IP. *IEEE Security & Privacy*, 44-49.

Weisstein, E. (1999-2014). *Gamma Distribution*. Retrieved October 20, 2014, from Wolfram MathWorld : http://mathworld.wolfram.com/GammaDistribution.html

Weisstein, E. (1999-2014). *Normal Distribution*. Retrieved October 20, 2014, from Wolfram Mathworld: http://mathworld.wolfram.com/NormalDistribution.html

Weisstein, E. (1999-2014). *Uniform Distribution*. Retrieved October 16, 2014, from Wolfram Mathworld: http://mathworld.wolfram.com/UniformDistribution.html

Westerveld, R. (2011). *Inverse Telecommunications: The Future for Rural Areas in Developing countries?*

Wilson, M. (2006). A Historical View of Network Traffic Models.

Wyk, A. d. (2006). *Modelling of Ethernet traf fic with self-similar processes* . Rand Afrikaans University, Dept. of Electrical and Electronic Engineering.

Zangrilli, M., & Wekamp, B. B. (2003). Comparing Passive Network Monitoring of Grid Application Traffic with Active Probes. *Proceedings Fourth IEEE International Workshop In Grid Computing* (pp. 84-91). IEEE.

Constant distribution

The constant distributions have very predictable samples, which makes them ideal for testing functioning of a system before adding stochastic behaviour (Eindhoven University of Technology, 2012-2014). It is a distribution that always returns the same value i.e. a constant distribution with parameter p will always return p and the mean is p.

$$f_X(x) = \begin{cases} 1, & x = p \\ 0, & otherwise \end{cases} \quad (4)$$

Uniform distribution

The uniform distribution (often abbreviated U (a, b)) is defined by the two parameters, a and b, which are its minimum and maximum values. A random variable X follows a uniform distribution if the probability density function $f_X(x)$ and cumulative distribution function $F_X(x)$ on the interval [a, b] are defined as follow (Weisstein, Uniform Distribution, 1999-2014):

$$f_X(x) = \begin{cases} 0 & for\ x < a \\ \frac{1}{b-a} & for\ a \le x \le b \\ 0 & for\ x > b \end{cases} \quad (5)$$

$$F_X(x) = \begin{cases} 0 & for\ x < a \\ \frac{x-a}{b-a} & for\ a \le x \le b \\ 1 & for\ x > b \end{cases} \quad (6)$$

The mean and the variance of the uniform distribution are equalled to $\frac{a+b}{2}$ and $\frac{(b-a)^2}{12}$ respectively.

Poisson distribution

The Poisson distribution is the model commonly used for analyzing traffic in traditional telephony networks. In order to apply the Poisson distribution, the arrivals should be from a large number of independent sources, referred to as Poisson sources i.e. poisson processes are common in traffic

applications scenarios that comprise of a large number of independent traffic streams (Mohammed & Agamy, 2011). A discrete random variable X is said to be a Poisson random variable with parameter λ if the probability distribution of a given number of events occurring in a fixed interval of time and/or space is given by the formula (Bourne, 2014):

$$f_X(x) = \frac{(e^{-\lambda})\lambda^x}{x!} \tag{7}$$

$$F_X(x) = e^{-\lambda} \sum_{i=0}^{\lfloor k \rfloor} \frac{\lambda^i}{i!} \quad for\ k \geq 0 \tag{8}$$

X represents the number of occurrences within a given time interval or a specified region of space. λ = Mean or average number of occurrences in the given time interval or region of space. In a Poisson process the inter-arrival times are exponentially distributed with a rate parameter λ and both the mean and the variance of a Poisson distribution correspond to λ.

Exponential distribution

This distribution is commonly used to model waiting times between occurrences of events in a poisson process. Suppose that the number of events occurring in any time interval of length t has a Poisson distribution with parameter αt and the numbers of occurrences in non-overlapping intervals are independent of one another. Then the distribution of elapsed time between the occurrences of two successive events is exponential with parameter $\lambda = \alpha$.

The exponential distribution is a continuous probability distribution, which has the memoryless property. This means: a random variable with an exponential distribution forgets about its past. A random variable X has an exponential distribution with parameter λ if the PDF and the CDF is defined as follow (Weisstein, Gamma Distribution, 1999-2014):

$$f_X(x) = \begin{cases} \lambda e^{-\lambda x} & x \geq 0 \\ 0 & otherwise \end{cases} \tag{9}$$

$$F_X(x) = \begin{cases} 1 - e^{-\lambda x} & x \geq 0 \\ 0 & x < 0 \end{cases} \tag{10}$$

λ is called the rate parameter. The mean and the variance correspond to $\frac{1}{\lambda}$ and $\frac{1}{\lambda^2}$ respectively.

Normal distribution

The normal distribution (also referred to as the *Gaussian distribution)* is a fundamental distribution in probability and statistics, mostly because of the central limit theorem, which states that the distribution of the sum (or average) of a large number of independent, identically distributed variables will be approximately normal, regardless of the underlying distribution (Siegrist, Special Distributions: The Normal Distribution, 1997-2014).

A random variable X with mean μ and variance σ^2 has the normal distribution if it has the PDF and CDF given by (Weisstein, Normal Distribution, 1999-2014):

$$f_X(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad (11)$$

$$F_X(x) = \frac{1}{2} + \frac{1}{2} erf\left[\frac{x-\mu}{\sigma\sqrt{2}}\right] \quad (12)$$

Where erf is the so-called error function. The normal distribution with parameter values μ = 0 and σ = 1 is called a standard normal distribution.

Lognormal distribution

The Lognormal distribution is a continuous probability distribution in which the logarithm of a variable has a normal distribution (Weisstein, 1999-2014). In this sense, a random variable X has a Lognormal distribution if $Y = \log X$ is normally distributed and vice versa Y has a normal distribution if $X = e^Y$ is log-normally distributed. The Lognormal distribution is defined by the two parameters, μ and σ, which are respectively the mean and standard deviation of the random variable's natural logarithm. The variable's logarithm is normally distributed i.e $X = e^{\mu+\sigma Z}$ with Z a standard normal variable.

The probability density and cumulative distribution functions for the lognormal distribution are given as (Weisstein, 1999-2014):

$$f_X(x) = \frac{1}{\sigma x\sqrt{2\pi}} e^{-\frac{(\ln x-\mu)^2}{2\sigma^2}} \quad (13)$$

$$F_X(x) = \frac{1}{2} + \frac{1}{2} erf\left[\frac{\ln x - \mu}{\sigma\sqrt{2}}\right] \quad (14)$$

The distribution has a mean and variance equal to the parameter $e^{\mu+\sigma^2/2}$ and $\left(e^{\sigma^2} - 1\right)e^{2\mu+\sigma^2}$ respectively.

Gamma distribution

A gamma distribution arises naturally in processes for which the waiting times between Poisson distributed events are relevant (i.e. particularly used to model the arrival times in the poisson process). Gamma distributions have two parameters, the shape parameter α and scale parameter θ. The reciprocal $\lambda = \frac{1}{\theta}$ is known as the rate parameter in the poisson process. If $\alpha = 1$ the gamma distribution is called the exponential distribution with scale parameter θ (or rate parameter λ).

The PDF and CDF of a random variable X that has the gamma distribution are defined as follow (Weisstein, Gamma Distribution, 1999-2014):

$$f_X(x) = \frac{x^{\alpha-1}e^{-\frac{x}{\theta}}}{\boldsymbol{\Gamma(\alpha)\theta^{\alpha}}} \quad (15)$$

$$F_X(x) = P\left(\alpha, \frac{x}{\theta}\right) = \frac{\boldsymbol{\gamma\left(\alpha, \frac{x}{\theta}\right)}}{\boldsymbol{\Gamma(\alpha)}} \quad (16)$$

Where $P\left(\alpha, \frac{x}{\theta}\right)$, $\gamma\left(\alpha, \frac{x}{\theta}\right)$ and $\Gamma(\alpha)$ are regularized gamma function, incomplete gamma function and complete gamma function respectively. The mean is $\alpha\theta$ and the variance corresponds to $\alpha\theta^2$

Geometric distribution

It is discrete probability distribution used to model the trial number of the first success in a sequence of Bernoulli trials. The parameter $P\epsilon\,(0,1]$ represents the success parameter. When performing a sequence of Bernoulli trials $X = (X_1 + X_2 + \cdots)$ with success parameter P, the random variable $N = min\{n \,\epsilon\, \mathbb{N}_+ : X_n = 1\}$ that gives the trial number of the first success has a geometric distribution with the PMF and CDF defined as follow (Siegrist, Special Distributions: Bernouilli Trials, 1997-2014):

$$f_N(n) = p(1-p)^{n-1} \quad n \in \mathbb{N}_+ \tag{17}$$

$$F_N(n) = 1 - (1-p)^n \quad n \in \mathbb{N} \tag{18}$$

The mean is computed as $\frac{1}{p}$ and the variance is $\frac{1-p}{p^2}$.

Cauchy distribution

The Cauchy distribution is part of the family of distributions for which the expected value and other moments do not exist since both its mean and its variance are undefined. A continuous random variable X has the Cauchy distribution with location parameter $a \in (-\infty, +\infty)$ and scale parameter $b \in (0, +\infty)$ if X has its PDF and CDF given by (Siegrist, Special Distributions: Bernouilli Trials, 1997-2014):

$$f_X(x) = \frac{b}{\pi[(x-a)^2 + b^2]} \tag{19}$$

$$F_X(x) = \frac{1}{2} + \frac{1}{\pi}\tan^{-1}\left(\frac{x-a}{b}\right) \tag{20}$$

If $\mathrm{a} = 0$ and $\mathrm{b} = 1$ then X has the standard cauchy distribution. This distribution is related to the normal distribution. If X and Y are two independent random variables with standard normal distribution, then the random variable $Z = X/Y$ has the standard cauchy distribution.

Weibull distribution

A continuous random variable X is said to have a Weibull distribution with parameters λ and κ if it has the PDF and CDF defined as:

$$f_X(x) = \begin{cases} \frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{k-1} e^{-(x/\lambda)^k} & x \geq 0 \\ 0 & x < 0 \end{cases} \tag{21}$$

$$F_X(x) = \begin{cases} 1 - e^{-(x/\lambda)^k} & x \geq 0 \\ 0 & x < 0 \end{cases} \tag{22}$$

With k > 0 is the shape and λ > 0 is the scale of the distribution. The mean and variance are respectively associated to $\lambda\Gamma\left(1+\frac{1}{k}\right)$ and $\lambda^2\left[\Gamma\left(1+\frac{2}{k}\right)-\left(\Gamma\left(1+\frac{1}{k}\right)\right)^2\right]$

Pareto distribution

The Pareto distribution is applied to model self-similar arrival in packet traffic.

The Pareto distribution is a continuous power-law probability distribution that is defined by the following functions (Siegrist, 1997-2014):

$$f_X(x) = \frac{ab^a}{x^{a+1}} \quad b \le x \le +\infty \tag{23}$$

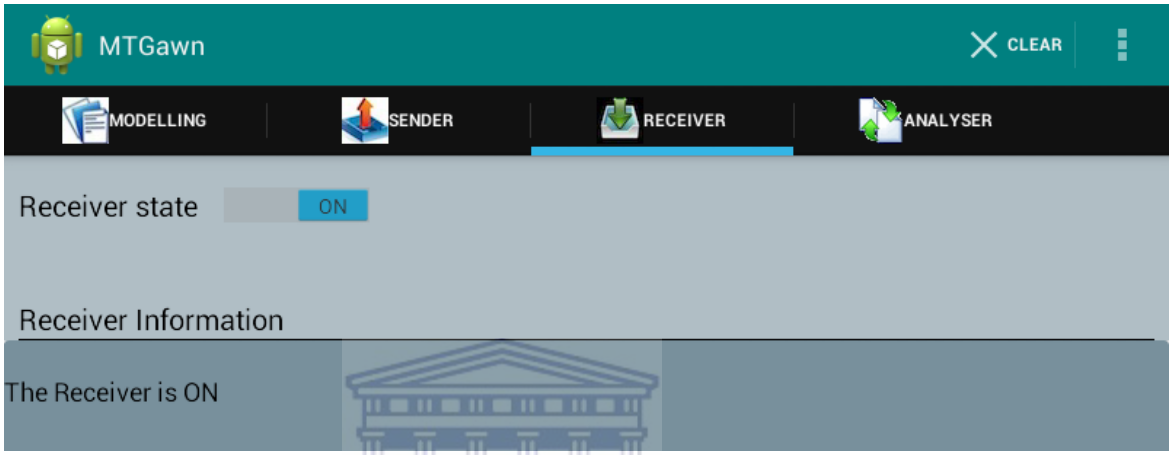
$$F_X(x) = 1 - \left(\frac{b}{x}\right)^a \quad b \le x < +\infty \tag{24}$$

It is said that the random variable X has the Pareto distribution with shape parameter $a > 0$ and scale parameter $b > 0$. The Pareto Distribution is heavy-tailed distribution as the convergence $(f(x) \to 0\ as\ x \to \infty)$ is at a power rate rather than an exponential rate.

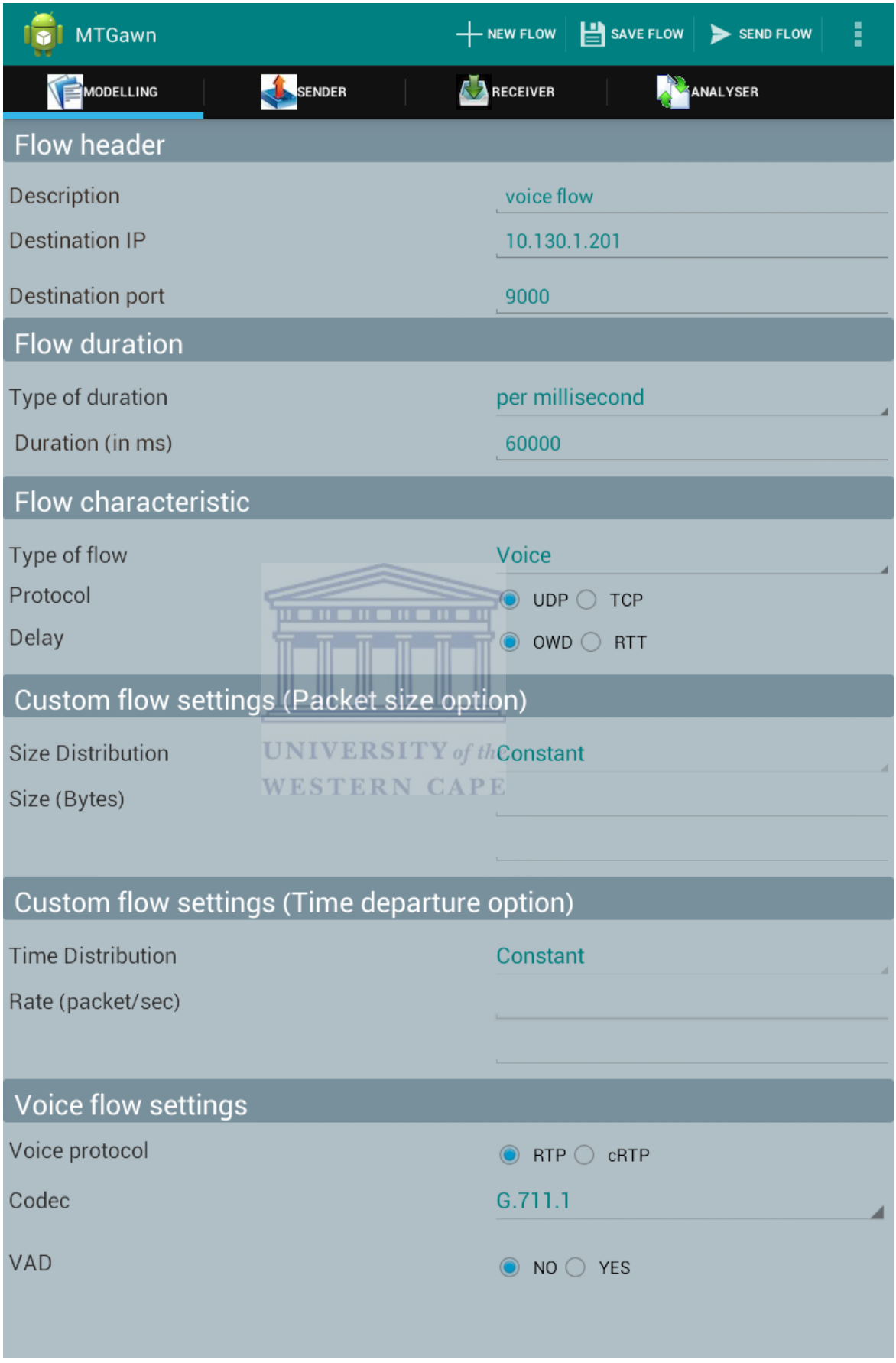
The mean is $\frac{ba}{a-1}\ , a > 1$ and the variance is $\frac{b^2a}{(a-1)^2(a-2)}\ , a > 2$

Single flow

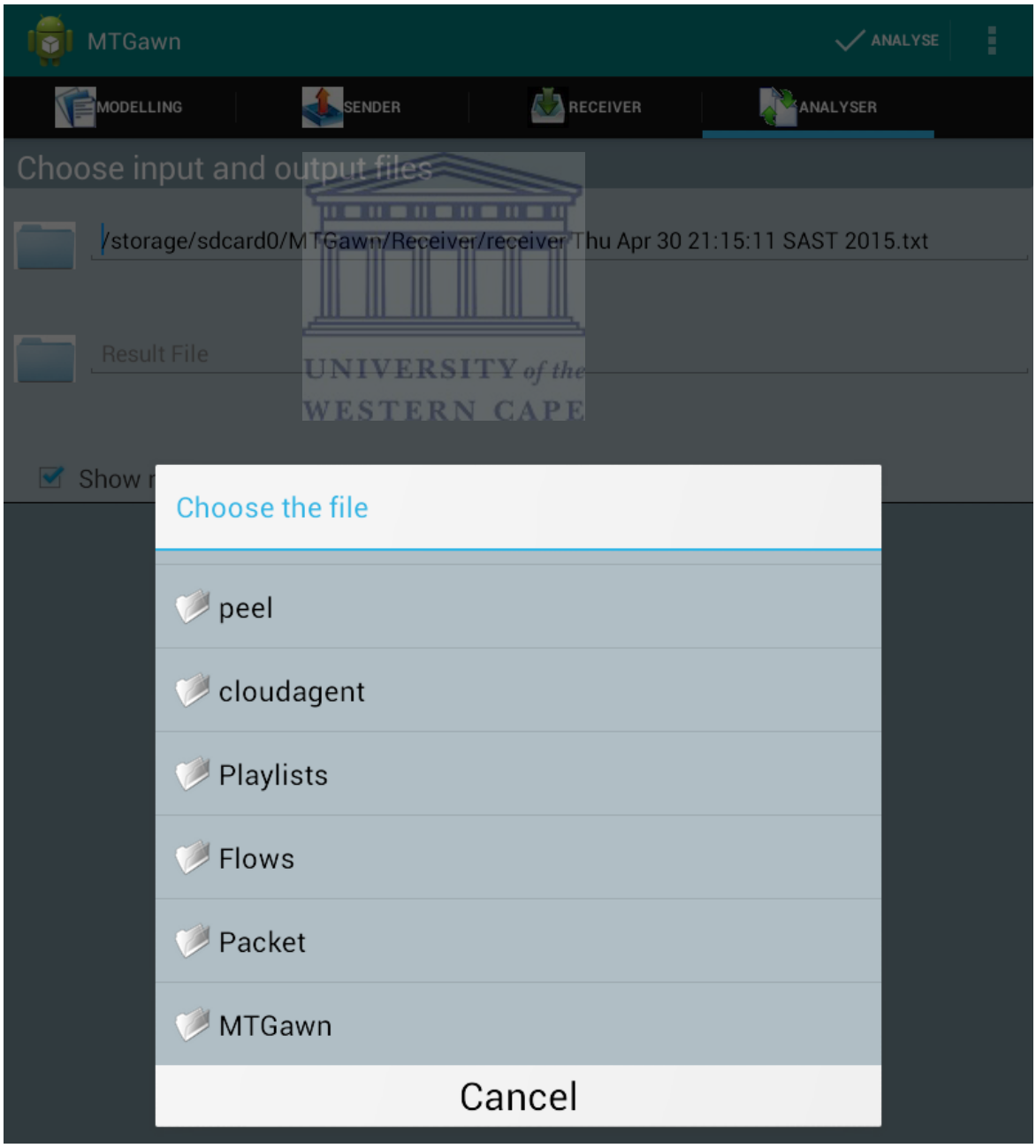
- ✓ On the mobile receiver: go to the *'Receiver fragment'* and start the receiver by pressing the *'Receiver state'* switch button. The receiver is now *"on"*.



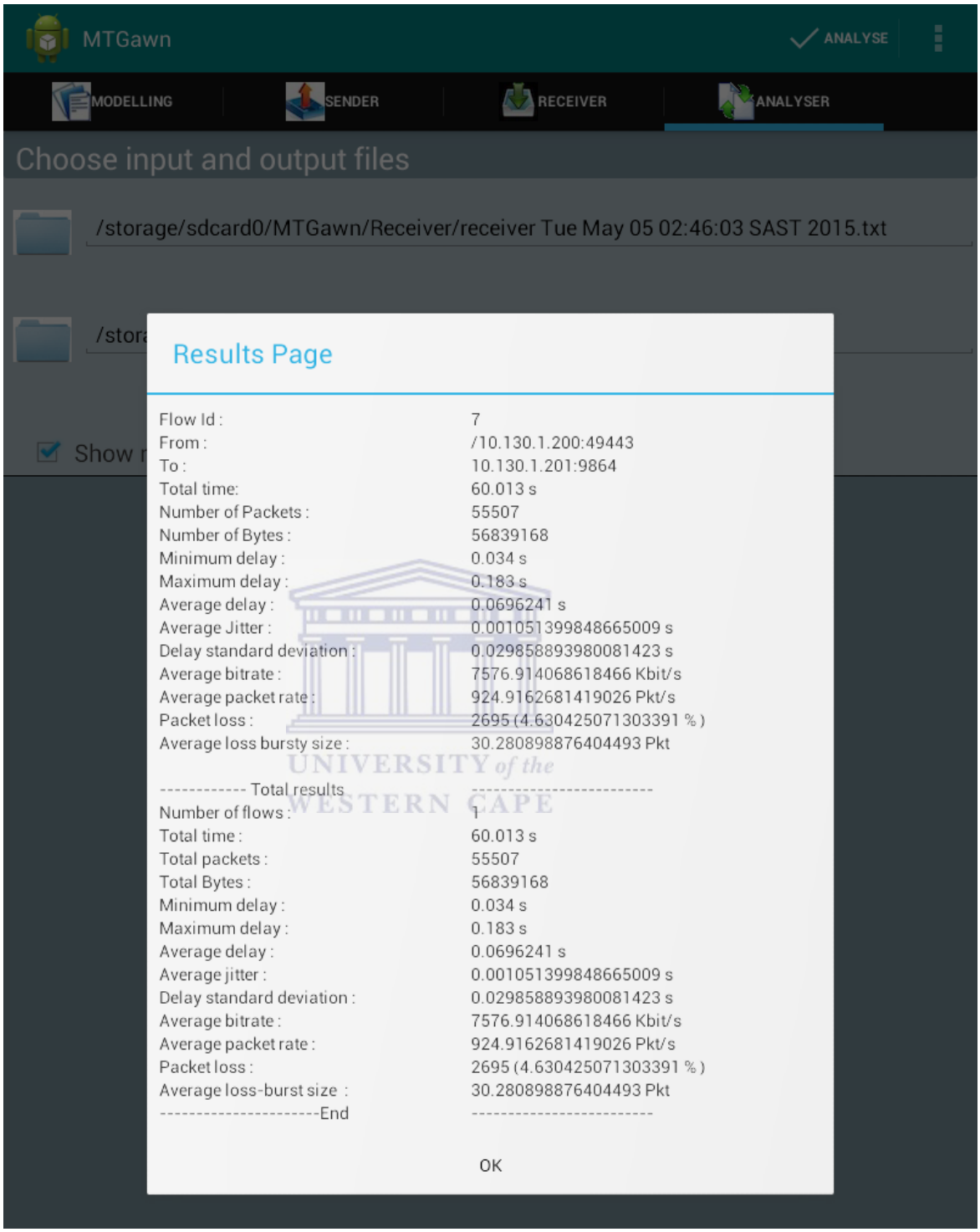
- ✓ On the mobile sender: go to the *'Modelling fragment'* and fill form providing all the information needed to generate a new traffic flow. Then validate by clicking on the menu item *'send flow'*.



- ✓ On the mobile receiver: terminate the generation by turning off the *'Receiver state'* switch button. The receiver is now *"off"*.
- ✓ During the generation, two log files are automatically created: one file is created on the mobile sender and another file on the mobile receiver. Both the sender and receiver log files are named according to the date and time when the flow was generated: e.g.: *'Sender Sun Apr 19 00/51/55 SAST 2015.txt'* and *'Receiver Sun Apr 19 00/51/55 SAST 2015.txt'*. Both files are located in the path: 'storage/sdcard0/MTGawn/'.
- ✓ In case the user wants to evaluate the *'OWD'*, he/she needs to analyse the receiver file located on mobile receiver. In case the user wants to evaluate the *'RTT'*, he/she needs to analyse the sender file located on mobile sender.
- ✓ To analyse a file: go to the *'Analyser fragment'* and choose the file to analyse by giving its location. Then validate by clicking on the menu item *'analyse'*.



✓ The results are shown in a *'dialog box'* screen and can also be saved in a file.

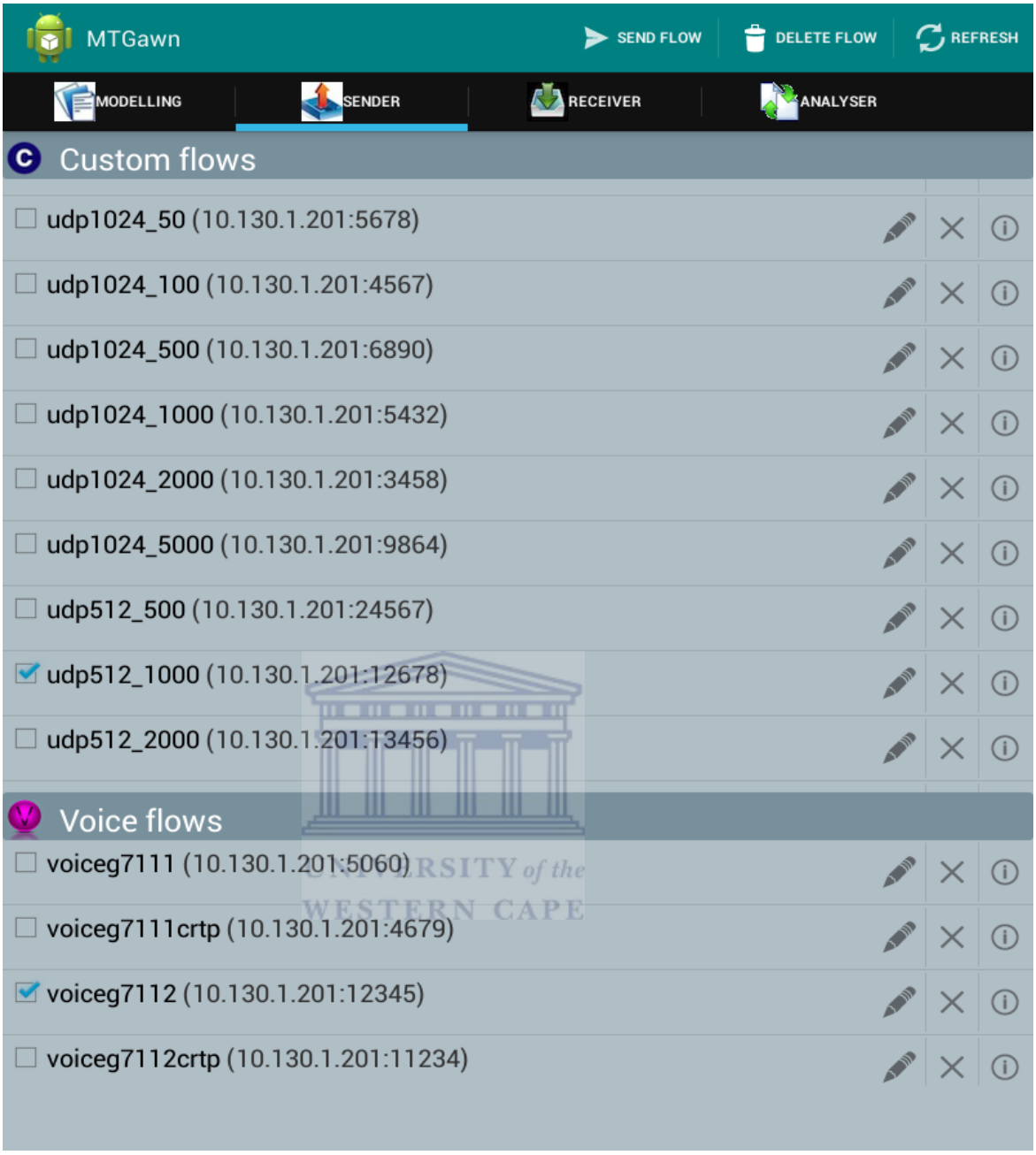


Multiple flows

- ✓ On the mobile receiver: go to the *'Receiver fragment'* and start the receiver by pressing the *'Receiver state'* switch button. The receiver is now *"on"*.

- ✓ On the mobile sender: go to the *'Modelling fragment'* and fill form providing all the information needed to generate a new traffic flow. Then validate by clicking on the menu item '*save flow'*. Repeat this same step until all the flows to be sent are created and saved.

- ✓ On the mobile sender: go to the *'Sender fragment'*, choose all the flows to be sent and press on the menu item *'send flow'*. The sender interface shows the list of flows grouped by category. The list of all the saved flows is displayed and classified depending on the type of flow (either '*Custom'* or *'Voice'*). One or many flows can be chosen (checked) and sent at the same time. The user can checked multiple flows to be sent simultaneously. Each flow is defined by its description followed by the receiver IP address and the port where the flow should be sent in a bracket. The *"edit"* button allows changing the setting of a flow. The *"delete"* button is used to delete a flow and the *"info"* button is used to see the characteristic of a specific flow.



- ✓ On the mobile receiver: terminate the generation by turning off the *'Receiver state'* switch button. The receiver is now *"off"*.
- ✓ The analysis is done as previously described for the single flow.

Appendix C : Throughput and loss rate

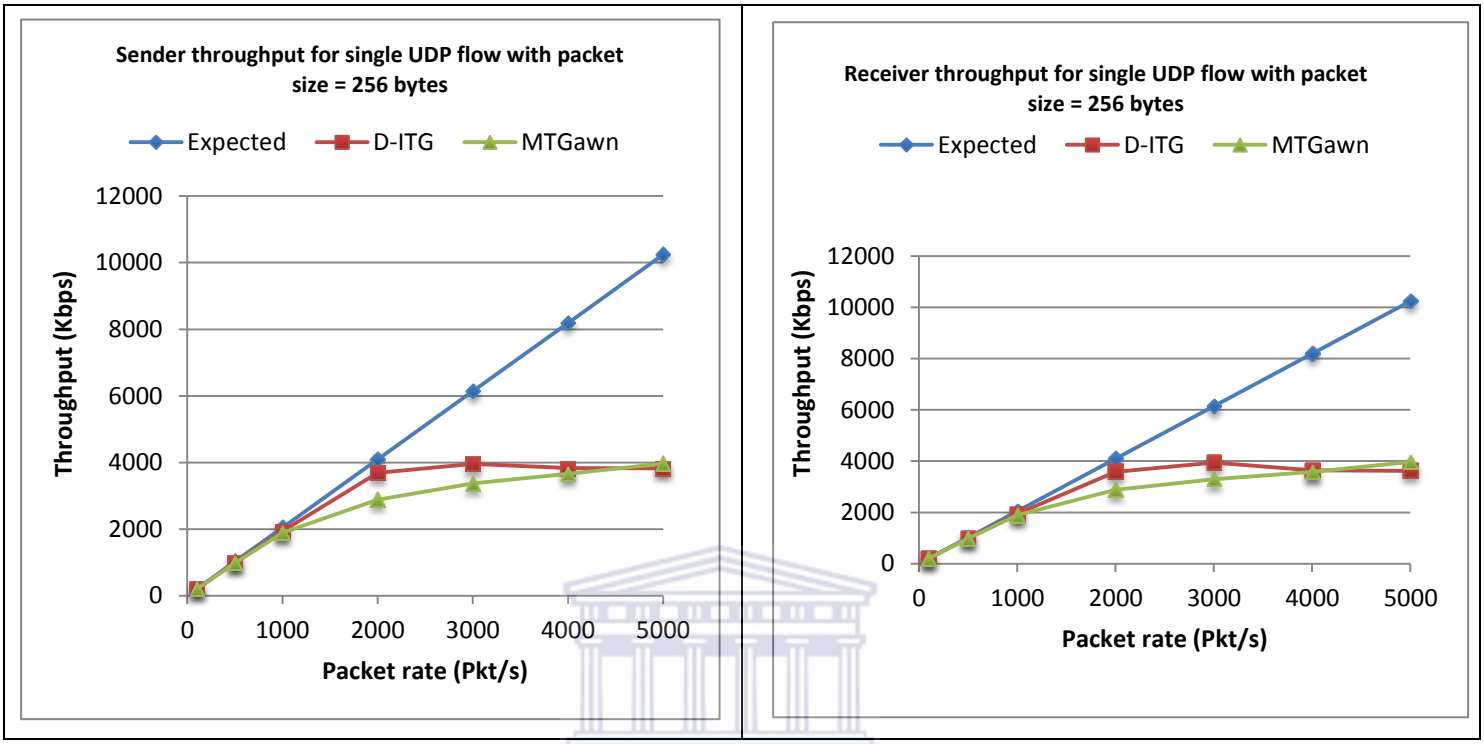


Figure C.1: Throughput for single UDP flow at sender side (left) and receiver side (right) for packet size = 256 bytes

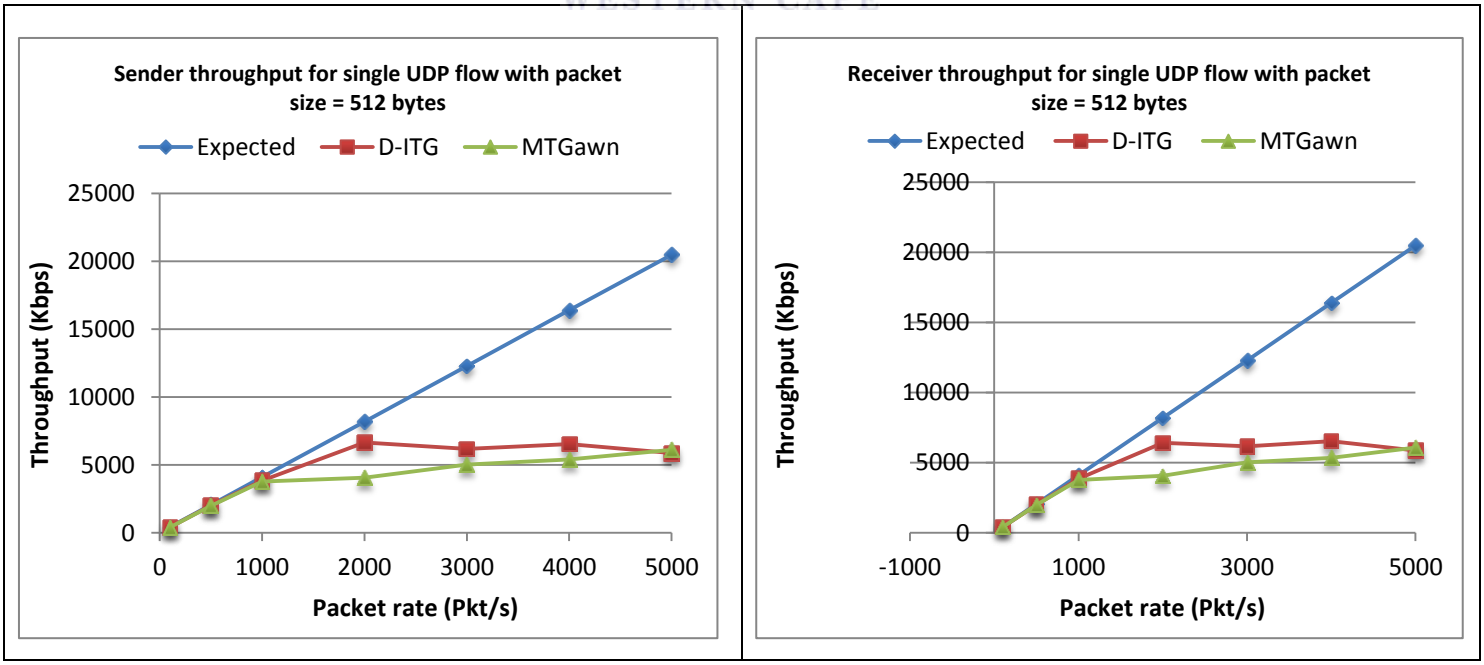


Figure C.2: Throughput for single UDP flow at sender side (left) and receiver side (right) for packet size = 512 bytes

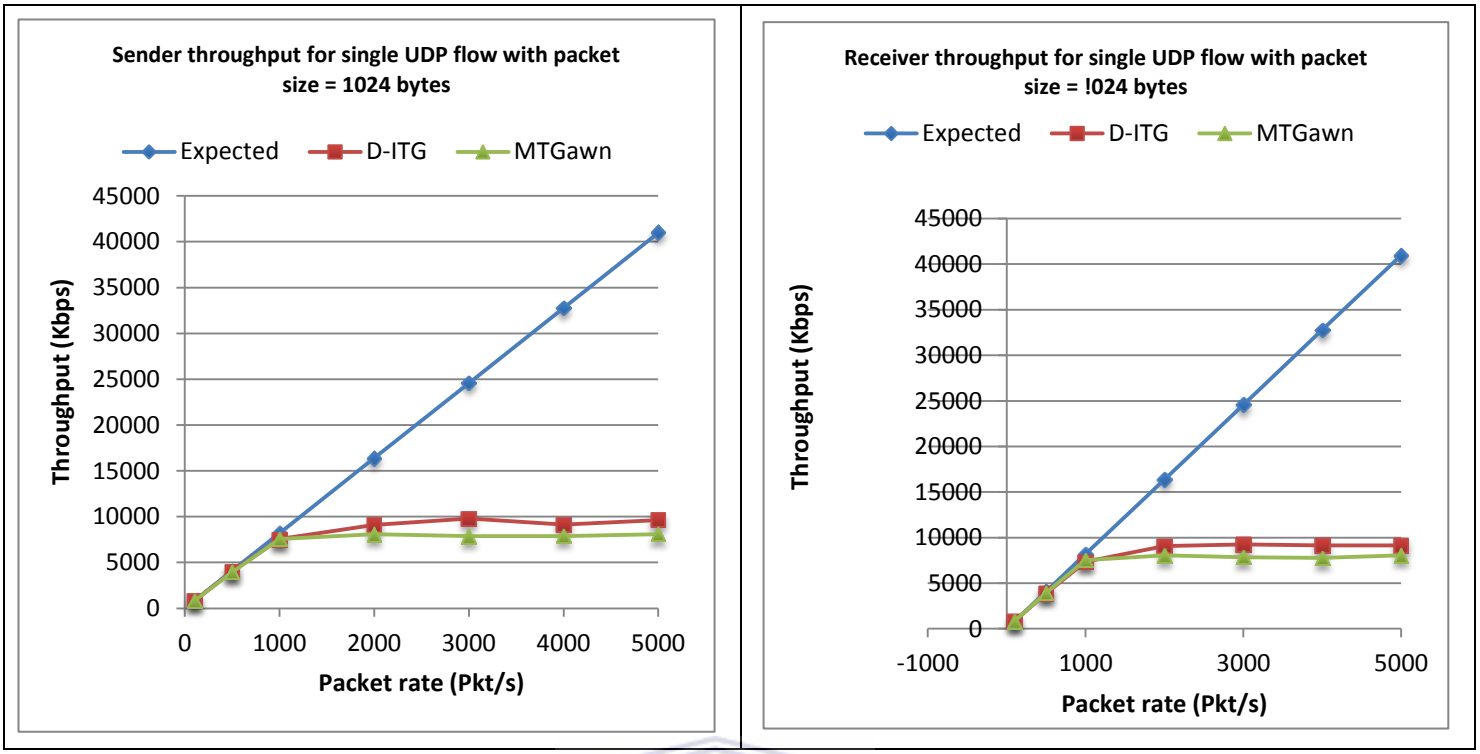


Figure C.3: Throughput for single UDP flow at sender side (left) and receiver side (right) for packet size = 1024 bytes

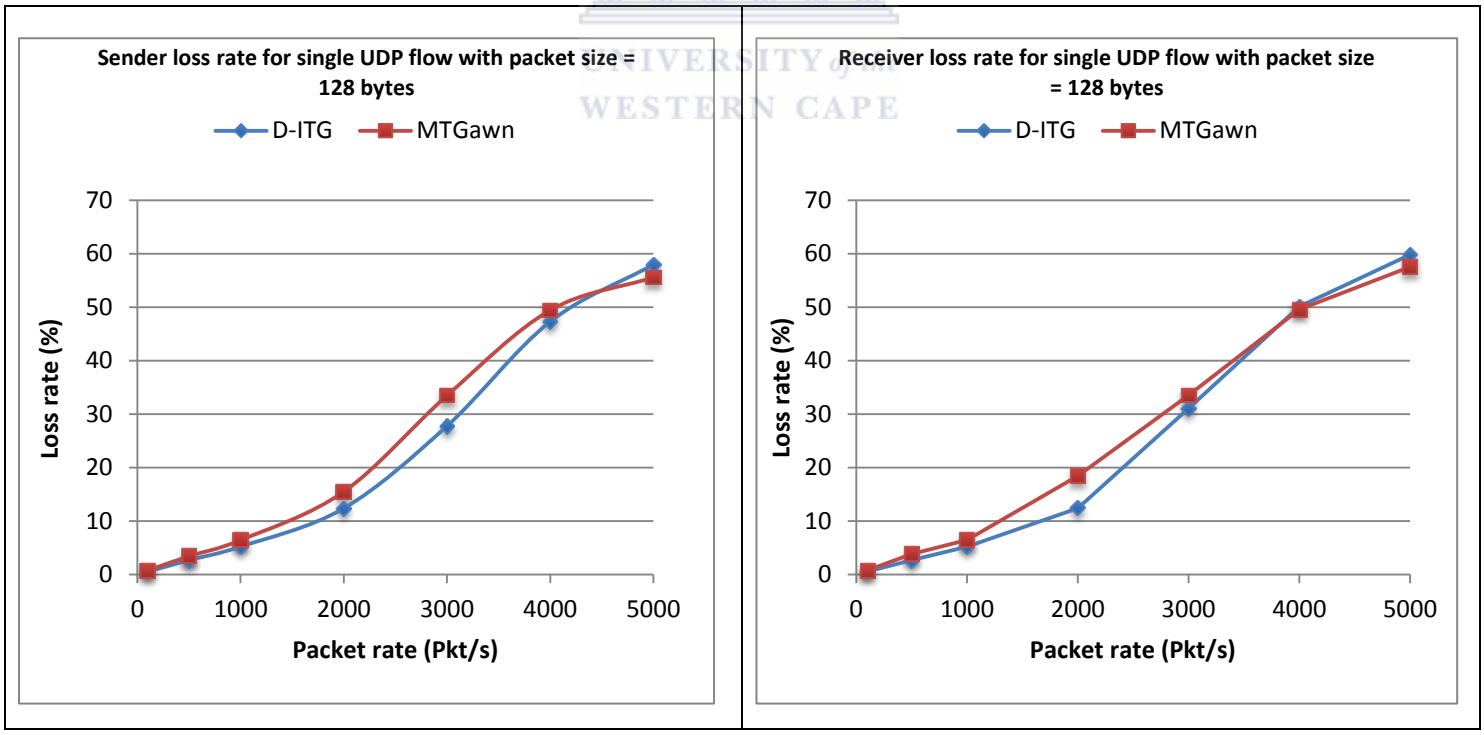


Figure C.4: Loss rate for single UDP flow at sender side (left) and receiver side (right) for packet size = 128 bytes

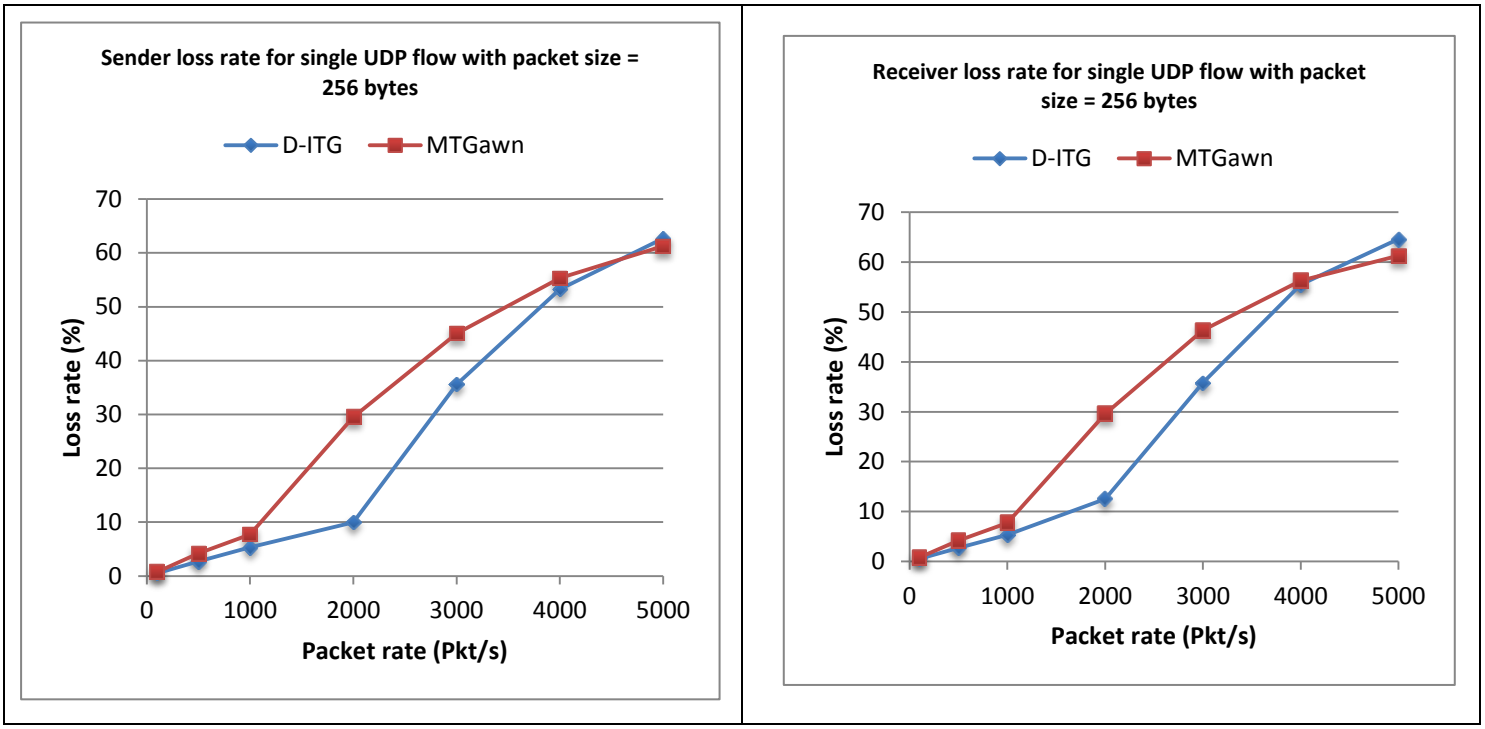


Figure C.5: Loss rate for single UDP flow at sender side (left) and receiver side (right) for packet size = 256 bytes

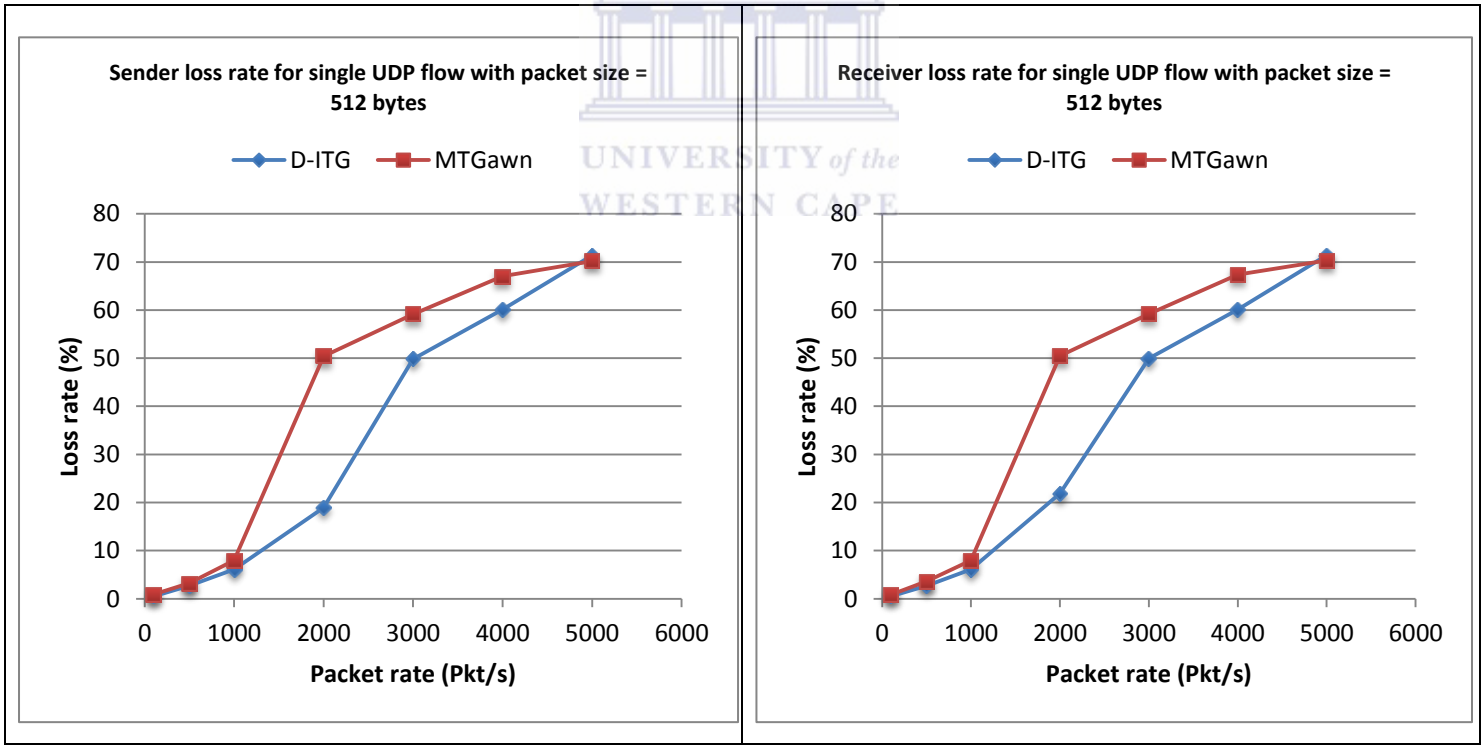


Figure C.6: Loss rate for single UDP flow at sender side (left) and receiver side (right) for packet size = 512 bytes

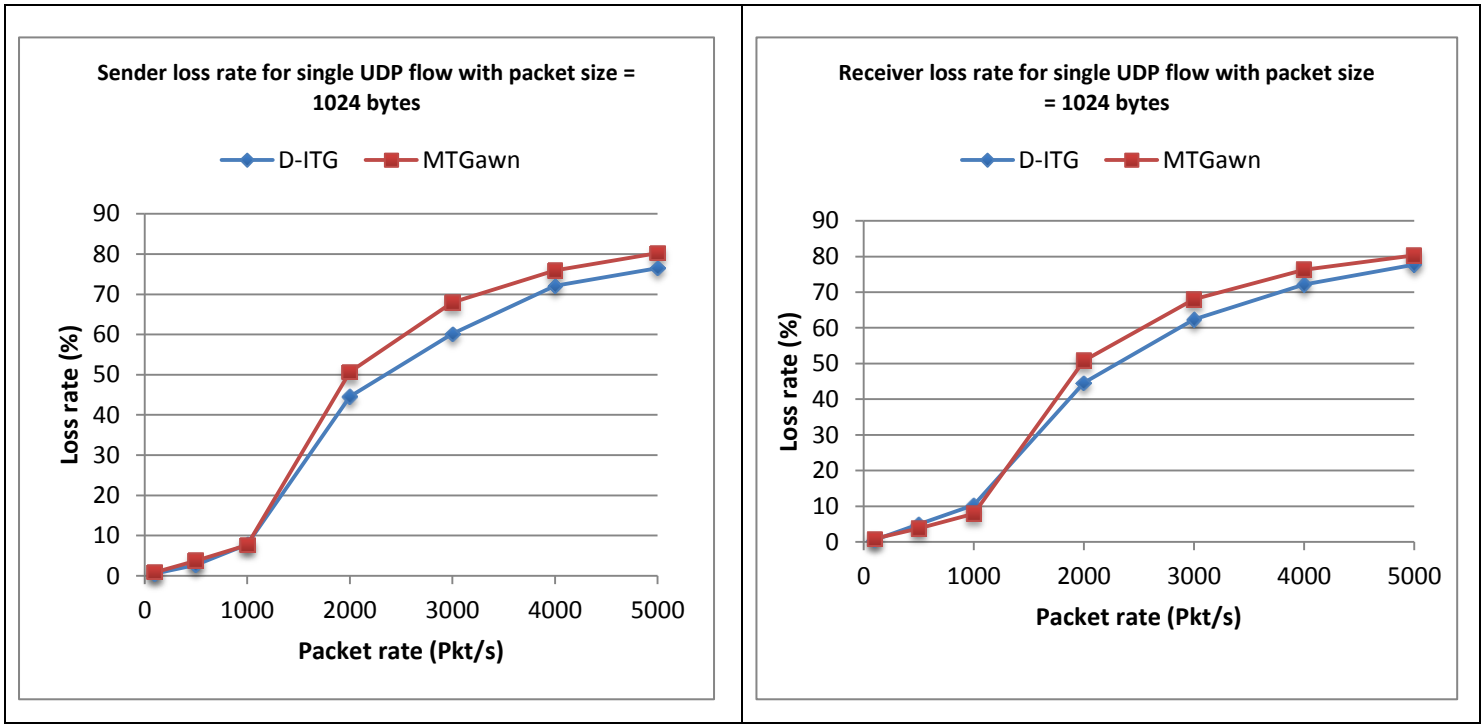


Figure C.7: Loss rate for single UDP flow at sender side (left) and receiver side (right) for packet size = 1024 bytes

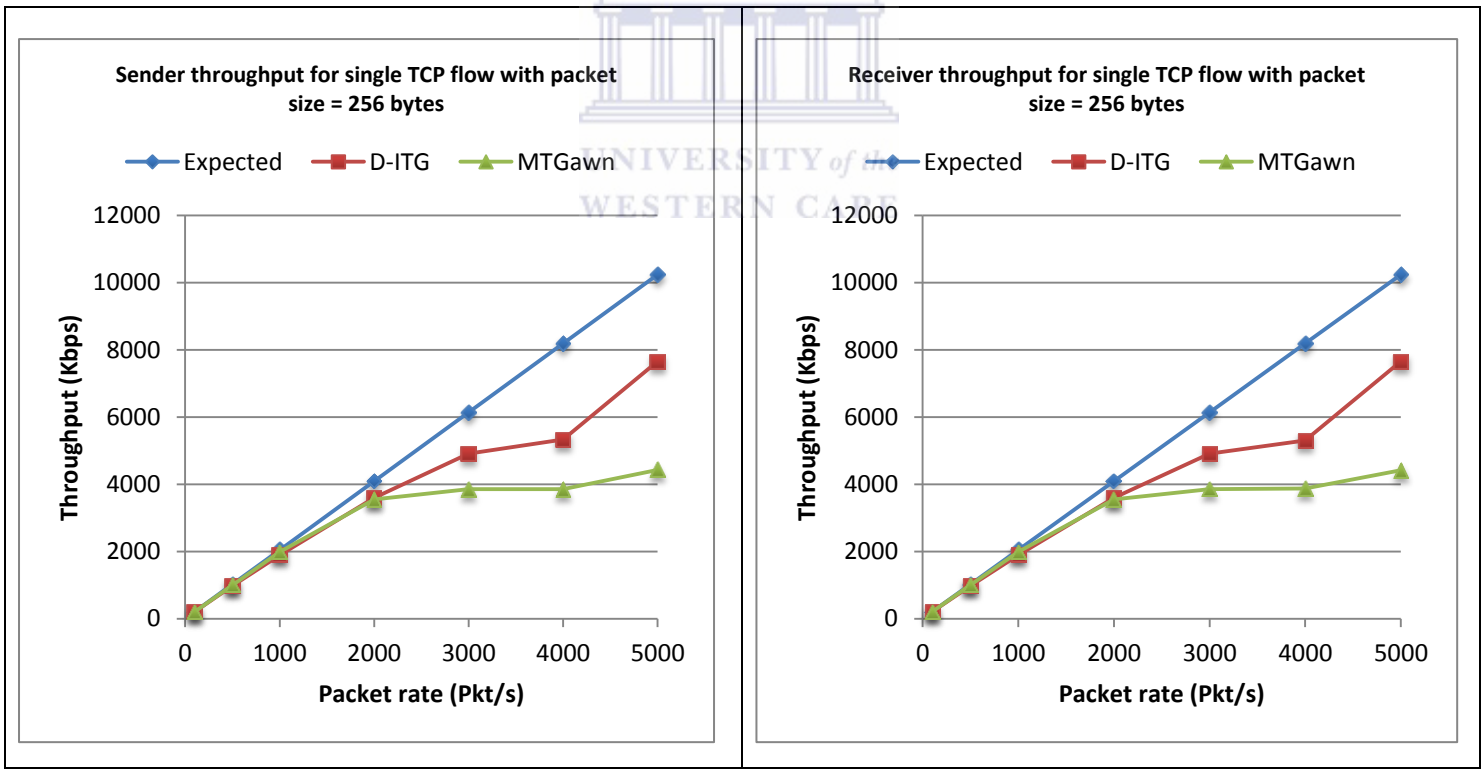


Figure C.8: Throughput for single TCP flow at sender side (left) and receiver side (right) for packet size = 256 bytes

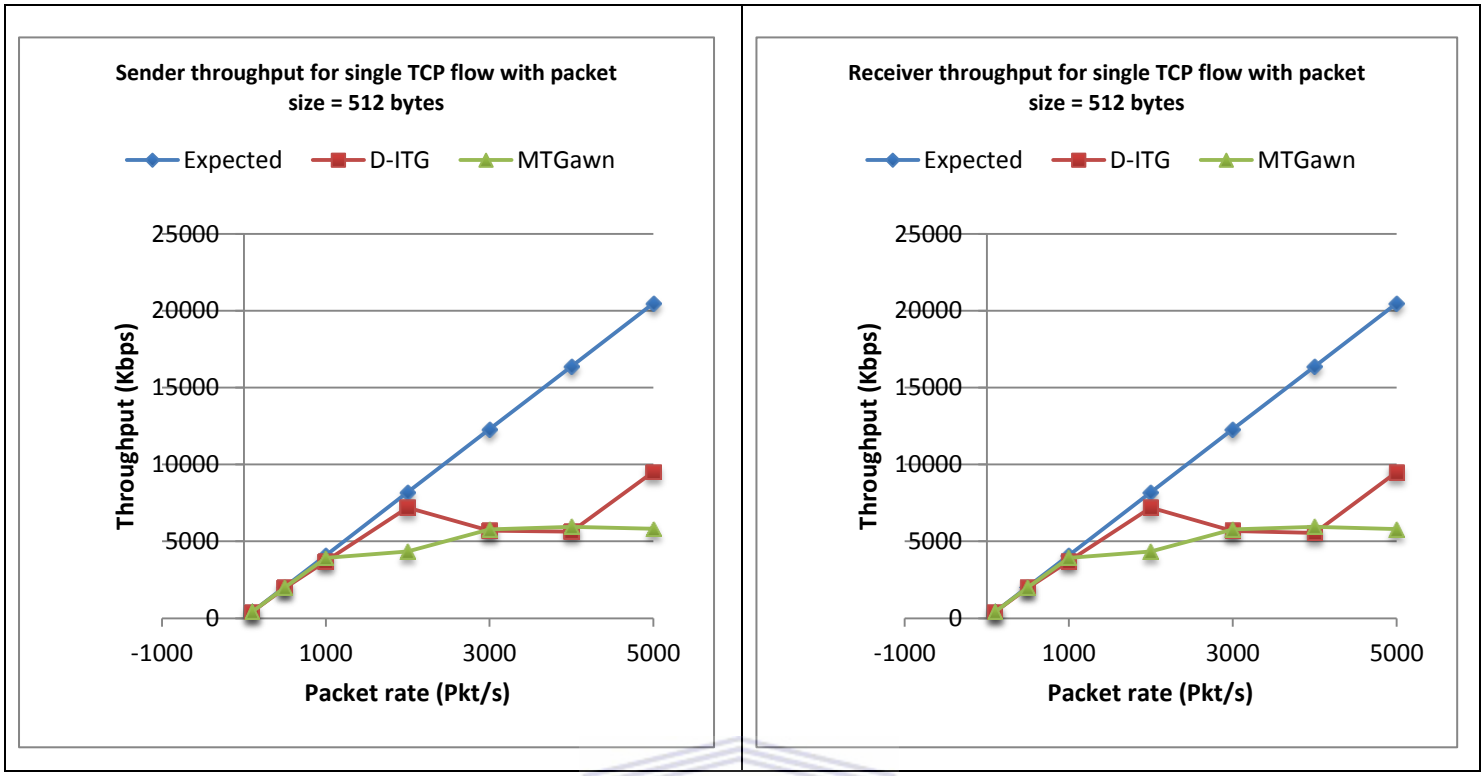


Figure C.9: Throughput for single TCP flow at sender side (left) and receiver side (right) for packet size = 512 bytes

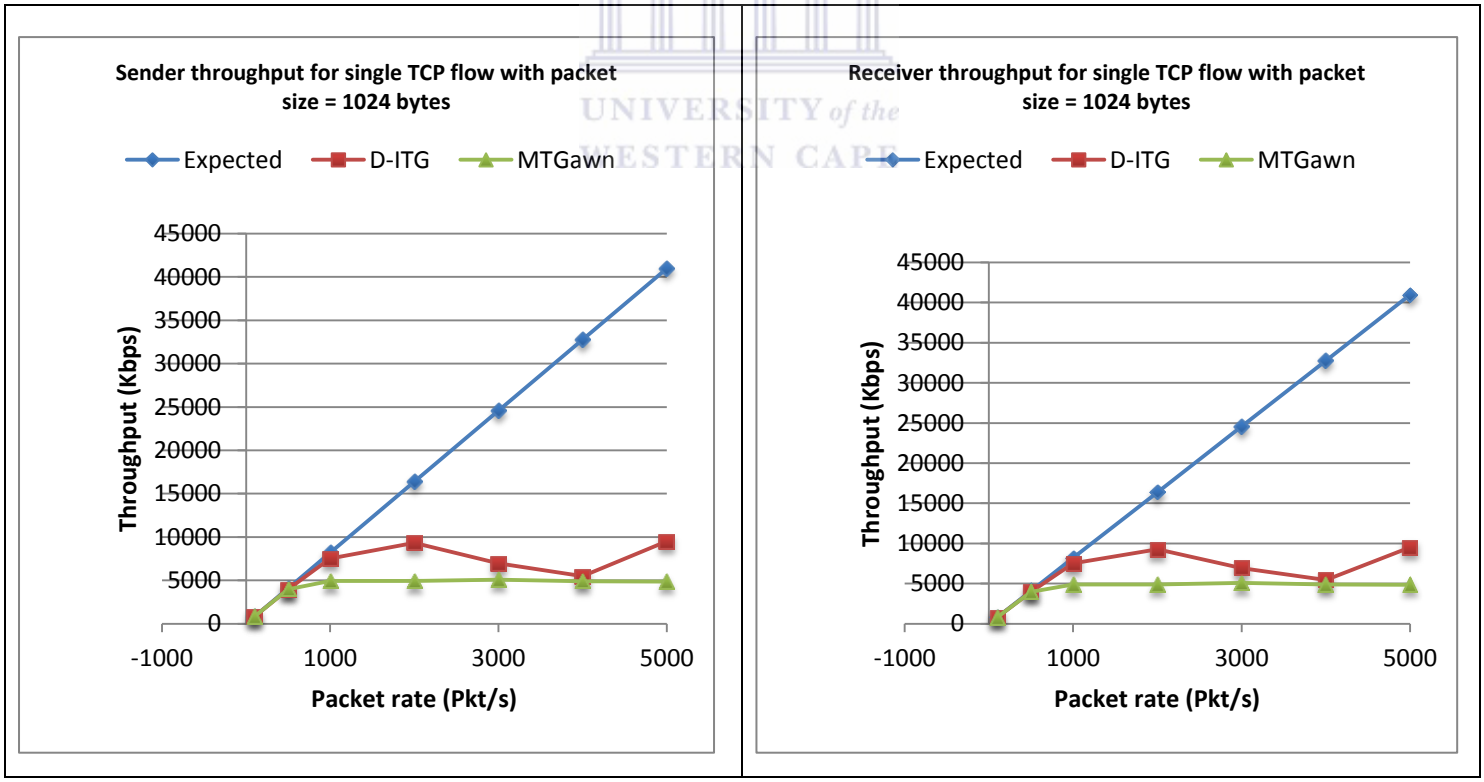


Figure C.10: Throughput for single TCP flow at sender side (left) and receiver side (right) for packet size = 1024 bytes

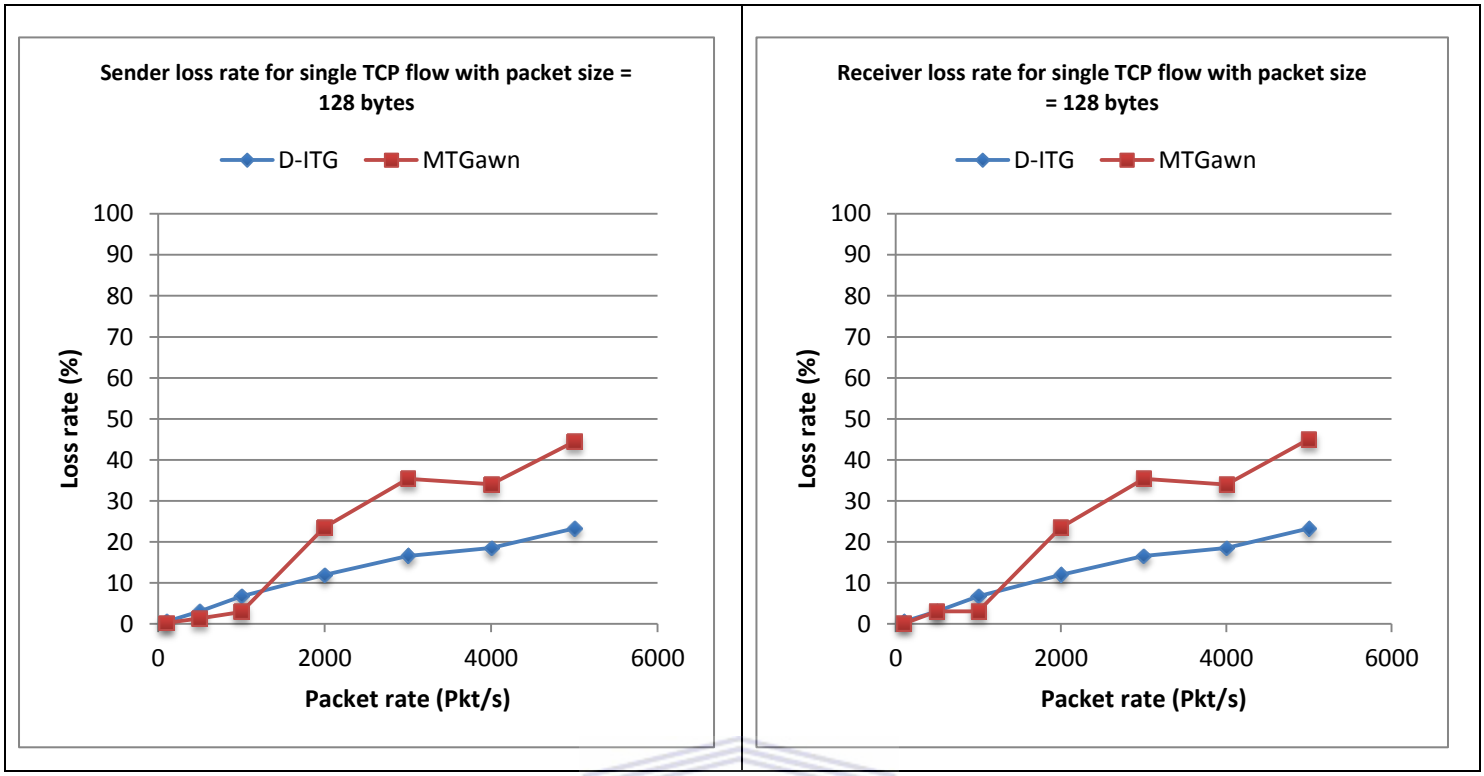


Figure C.11: Loss rate for single TCP flow at sender side (left) and receiver side (right) for packet size = 128 bytes

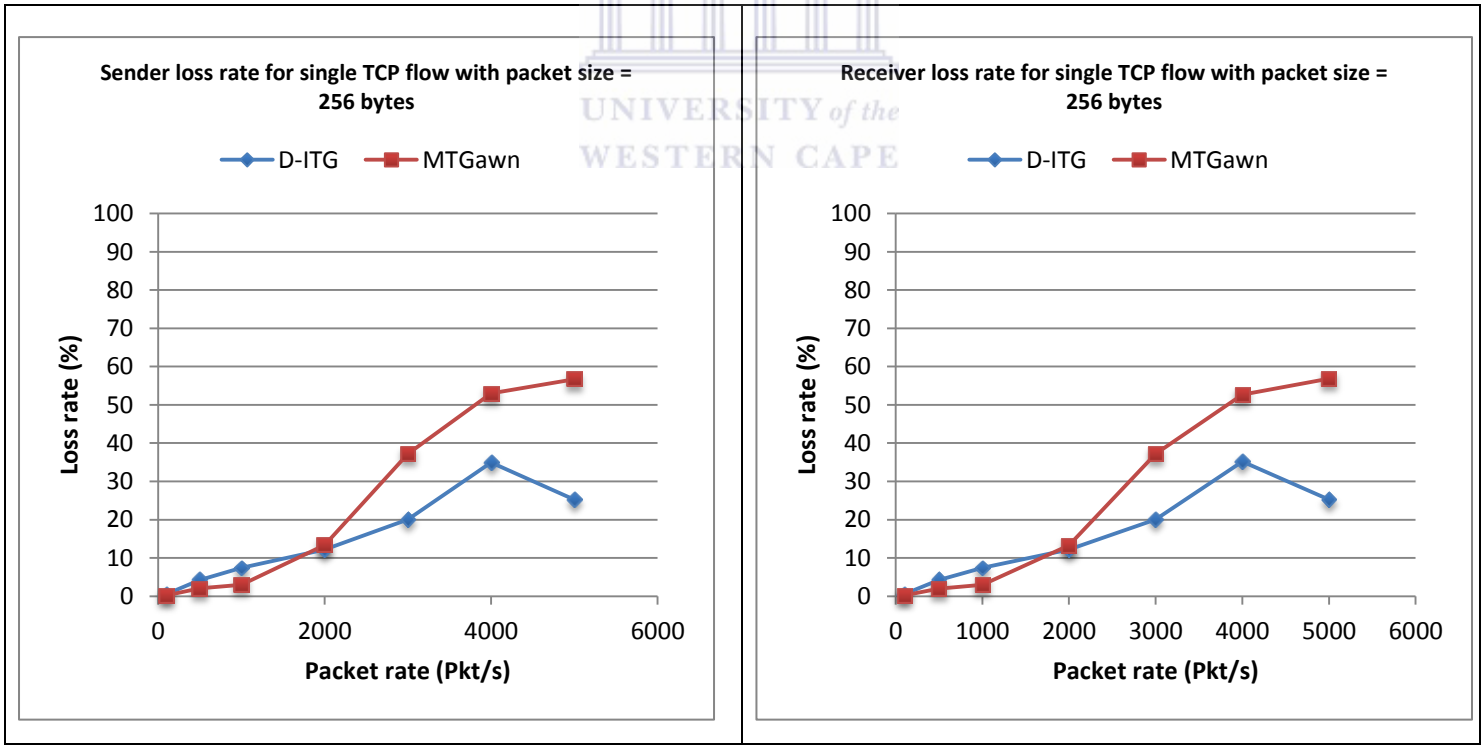


Figure C.12: Loss rate for single TCP flow at sender side (left) and receiver side (right) for packet size = 256 bytes

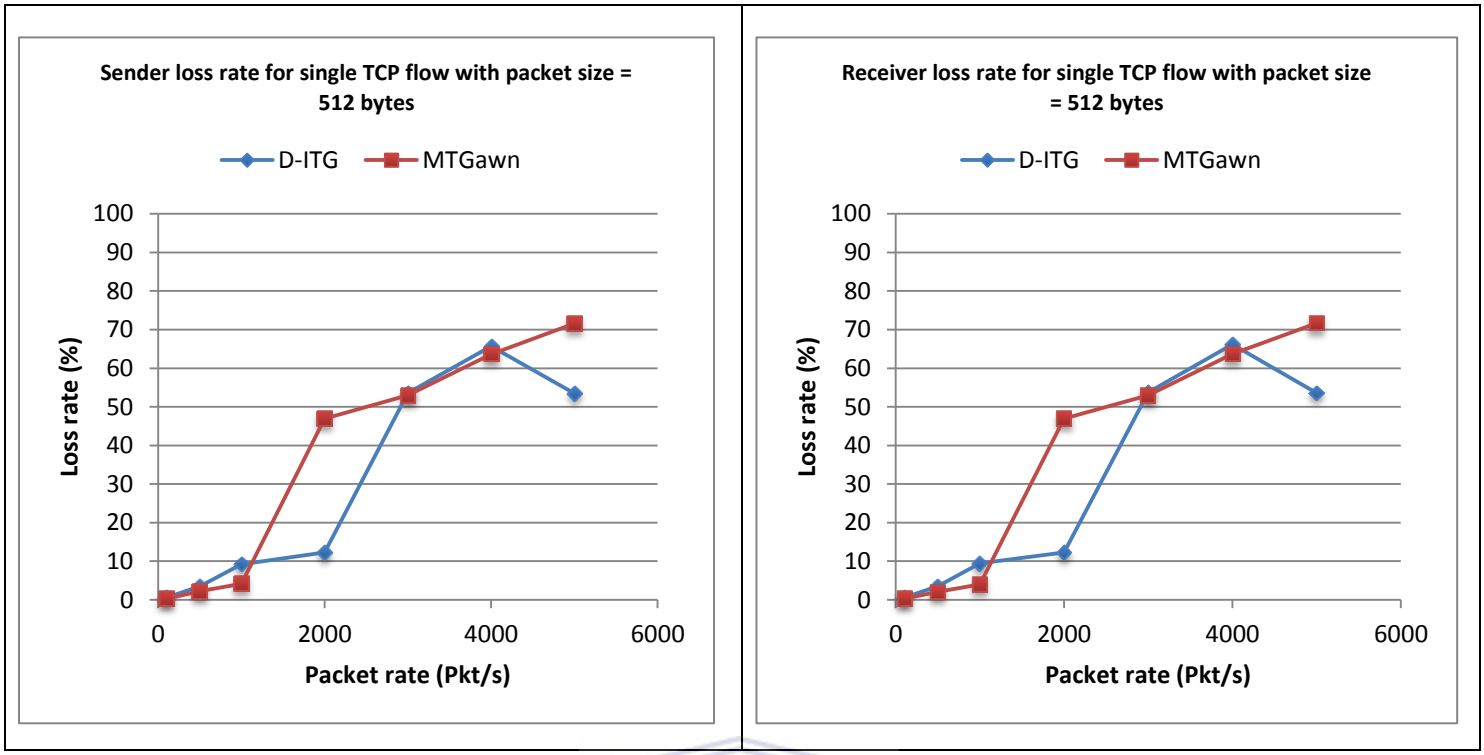


Figure C.13: Loss rate for single TCP flow at sender side (left) and receiver side (right) for packet size = 512 bytes

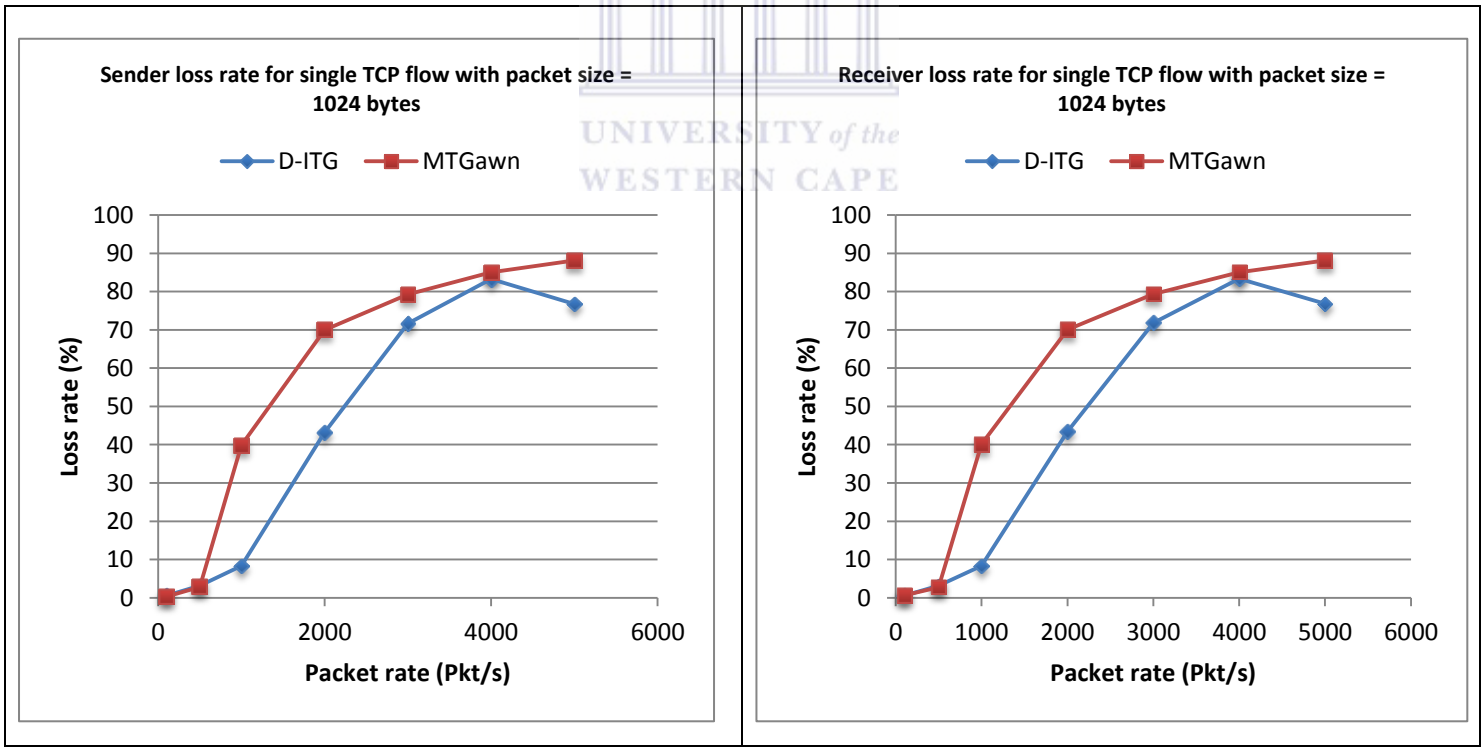


Figure C.14: Loss rate for single TCP flow at sender side (left) and receiver side (right) for packet size = 1024 bytes