UNIVERSITY OF THE WESTERN CAPE

Packet aggregation for Voice over Internet Protocol on wireless mesh networks

by

Docas Dudu Zulu

Supervisor: Dr William D Tucker

A thesis submitted in fulfillment of the degree of Masters of Computer Science

in the
Department of Computer Science

March 2012

Keywords

C.2.1 [**Network Architecture and Design**]: Wireless Communication/Network Communication; B.8 [**Performance and Reliability**]: Performance and Design Aids; C.4 [**Performance of Systems**]: Performance Attributes

Abstract

This thesis validates that packet aggregation is a viable technique to increase call capacity for Voice over Internet Protocol over wireless mesh networks. Wireless mesh networks are attractive ways to provide voice services to rural communities. Due to the ad-hoc routing nature of mesh networks, packet loss and delay can reduce voice quality. Even on non-mesh networks, voice quality is reduced by high overhead, associated with the transmission of multiple small packets. Packet aggregation techniques are proven to increase VoIP performance and thus can be deployed in wireless mesh networks. Kernel level packet aggregation was initially implemented and tested on a small mesh network of PCs running Linux, and standard baseline vs. aggregation tests were conducted with a realistic voice traffic profile in hop-to-hop mode. Modifications of the kernel were then transferred to either end of a nine node 'mesh potato' network and those tests were conducted with only the end nodes modified to perform aggregation duties. Packet aggregation increased call capacity expectedly, while quality of service was maintained in both instances, and hop-to-hop aggregation outperformed the end-to-end configuration 4:1. However, implementing hop-to-hop in a scalable fashion is prohibitive, due to the extensive kernel level debugging that must be done to achieve the call capacity increase. Therefore, end-to-end call capacity increase is an acceptable compromise for eventual scalable deployment of voice over wireless mesh networks.

Declaration of Authorship

I, Docas Dudu Zulu, declare that *Packet aggregation for Voice over Internet Protocol on wireless mesh networks* 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

Acknowledgements

Firstly, I would like to thank my God, the Lord Jesus, my Heavenly Father, for helping me in my research and giving me the excellent spirit of Daniel, the intelligence and the divine speed to overtake, pursue and overcome. My Lord, I thank you. I give you all the glory and I love you so much, my first love, Jesus Christ! Secondly, I thank my daddy and mommy (Apostle Dr. Joshua and Pst. Blessing Simeon) at Command of Faith Miracle Ministries for their teachings of wisdom, prayers and support in my spiritual life. Thirdly, I would like to extend my greatest thanks to my supervisor Dr William D. Tucker for all the support he has given me all through my masters degree. I will not forget Telkom CoE for assisting me financially and giving me an opportunity to further my studies. Most importantly, I would like to thank my one and only 'nkem', Chinedu Godfrey Nwogo. I thank you for being a shoulder to cry on when it was hard, for encouraging me in the Lord, for the support and endless love you have shown me. I thank you and I love you. My colleagues at the Bridging Applications and Networks Group (BANG) lab: Guys, I thank you all. I especially thank Hlabishi Isaac Kobo, for your divine assistance, great ideas and compassion you have shown me. My sisters at home, Sihle and Lindiwe, my brother, Trevor and his wife Eliza: I thank you all for your patience and love you have shown me these two years!

Contents

List of Figures

List of Tables

List of Abbrevations

802.11	Set of wireless standards
ACK	**ACK**nowledgment
AH	**A**ggregation **H**eader
AODV	**A**d-hoc **O**n **D**emand
AODV-UU	**A**d-hoc **O**n **D**emand **V**ector **U**ppsala **U**niversity
AP	**A**ccess **P**oint
BATMAN	**B**etter **A**proach **T**o **M**obile **A**d-hoc **N**etworks
BANG	**B**ridging **A**pplications and **N**etworks **G**roup
BER	**B**it **E**rror **R**ate
BO	**B**ack **O**ff
CoE	**C**entre **o**f **E**xcellence
CPU	**C**entral **P**roccessing **U**nit
DCF	**D**istributed **C**oordination **F**unction
DIFS	**D**istributed **I**nter **F**rame **S**pace
DSDV	**D**istance **V**ector **P**rotocol
FIFO	**F**irst **I**n **F**irst **O**ut
FXS	**F**oreign e**X**change **S**tation
HDR	**H**ea**D**e**R**
HFSC	**H**ierarchical **F**air **S**ervice **C**urve
HTB	**H**ierarchical **T**oken **B**ucket
HWMP	**H**ybrid **W**ireless **M**esh **P**rotocol
IEEE	**I**nstitute of **E**lectrical and **E**lectronics **E**ngineers
IP	**I**nternet **P**rotocol
IPAC	**I**nternet **P**rotocol **A**daptive **C**oncatenation
IPv4	**I**nternet **P**rotocol **v**ersion 4
Iperf	**I**ntelligent **P**erformance **P**rediction
ITU	**I**nternational **T**elecommunication **U**nion
LAN	**L**ocal **A**rea **N**etwork
MAC	**M**edium **A**ccess **C**ontrol
MAP	**M**esh **A**ccess **P**oint

Mbps	**M**ega **b**its **p**er **s**econd
MCI	**M**aximum **C**oncatenation **I**nterval
MCS	**M**aximum **C**oncatenated **S**ize
MOS	**M**ean **O**pinion **S**core
MP	**M**esh **P**oints
MPDU	**M**AC **P**rotocol **D**ata **U**nit
MSDU	**M**AC **S**ervice **D**ata **U**nit
MTU	**M**aximum **T**ransmission **U**nit
NF	**N**et**F**ilter
ns2	**n**etwork **s**imulator version 2
OGM	**O**ri**G**inator **M**essage
PC	**P**ersonal **C**omputer
PLC	**P**acket **L**oss **C**oncealment
PSTN	**P**ublic **S**witched **T**elephone **N**etwork
Qdisc	**Q**ueuing **disc**ipline
QoS	**Q**uality **o**f **S**ervice
R-factor	**R**ating factor
ROHC	**RO**bust **H**eader **C**ompression
RTP	**R**eal-time **T**ransport **P**rotocol
SATNAC	**S**outhern **A**frica **T**elecommunication **N**etworks and **A**pplications **C**onference
SFQ	**S**tochastic **F**airness **Q**ueueing
SIFS	**S**hort **I**nter **F**rame **S**pace
SIP	**S**ession **I**nitiation **P**rotocol
skb	**s**oc**k**et **b**uffer
SNR	**S**ignal to **N**oise **R**atio
SPSSP	**S**imple **P**acket **S**ize **S**election **P**rotocol
TC	**T**raffic **C**ontrol
TCP	**T**ransmission **C**ontrol **P**rotocol
TTL	**T**ime **T**o **L**ive
UDP	**U**ser **D**atagram **P**rotocol
VoIP	**V**oice **o**ver **I**nternet **P**rotocol
WCETT	**W**eighted **C**umulative **E**xpected **T**ransmission **T**ime
WI-FI	**WI**reless **FI**delity
WLAN	**W**ireless **L**ocal **A**rea **N**etworks
WMN	**W**ireless **M**esh **N**etworks

Chapter 1

Introduction

This thesis validates that packet aggregation is a viable technique to increase call capacity for Voice over Internet Protocol (VoIP) over wireless mesh networks (WMNs). VoIP is known for its affordability relative to cellular networks, but its deployment over WMNs has introduced challenges. One challenge is that the transmission of very small VoIP packets degrades the quality of VoIP calls. Packet aggregation techniques were introduced and implemented to decrease the number of VoIP packets on a mesh network, in order to increase VoIP performance. Packet loss, jitter and delay, which are major causes of poor VoIP performance in a WMN's test bed, were measured and then the efficiency of packet aggregation was determined from these results. Section 1.1 introduces wireless mesh networks, routing protocols and the structure of a VoIP system. Section 1.2 addresses the complex interactions of these issues to establish a motivation for this research. Section 1.3 introduces the research question and the overall approach. Section 1.4 outlines the structure of this thesis.

1.1 Background

1.1.1 Wireless mesh networks

WMNs consist of wireless routers, that are known as mesh routers, as well as mesh clients (See Figure 1.1). The Institute of Electrical and Electronic Engineers' (IEEE) 802.11 devices are typically distinguished as Mesh Access Points (MAPs) and mesh routers, called Mesh Points (MPs). MAPs and MPs are designed to forward packets on behalf of other nodes. Their assignment is to extend wireless transmission range. MAP takes the place of a traditional Access Point (AP) on a wireless Local Area Network (LAN), whereas an MP is not an AP but can be a mesh router. This means that mesh clients can connect or associate to a MAP but not to an MP [30]. On the other hand, a mesh client can be mobile and allow end-users to connect to the WMNs, e.g. end-user devices such as laptops, cellphones etc. Mesh clients have added functionality that allow them to function as mesh routers. WMNs can be divided into 3 main groups [1]:

- Infrastructure/Backbone WMNs: The MAPs and MPs act as backbone for the clients when they connect to the MAPs that form the infrastructure mode.
- Clients' WMNs: The devices or the clients form a peer-to-peer network, i.e. they are responsible for performing routing decisions and configurations. In clients' WMNs, MAP is not required as the mesh clients are capable of providing all WMN services required by end-users.
- Hybrid WMNs: Hybrid combines the infrastructure and clients' mesh modes. This means that clients can access the Internet by forming a group of mesh clients and connecting to MAPs.

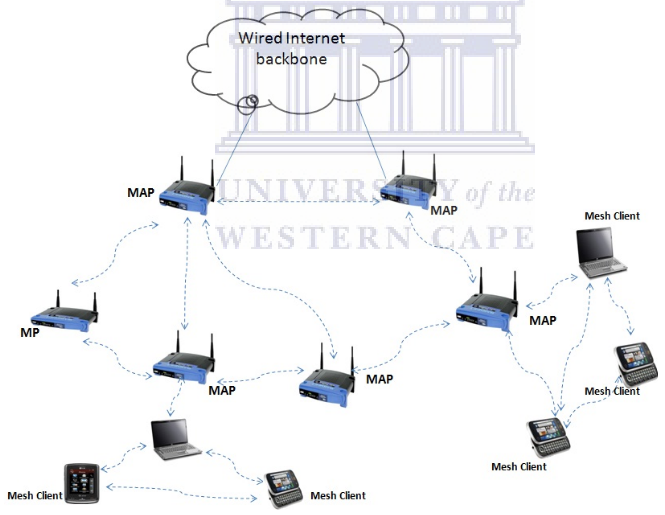

FIGURE 1.1 **Wireless mesh network architecture**
The figure illustrates the architecture of a WMN. It illustrates how mesh routes and mesh clients communicate with one another. The mesh clients represented by laptops and cellphones can communicate with the MAPs.

1.1.2 Wireless mesh routing protocols

WMNs are often called multihop, meaning that communication with neighbouring nodes is possible through more than two intermediate nodes. Therefore a proper routing protocol is required for effective route decision making. A routing protocol is defined as a means or process of determining the paths between source and destination nodes

[30]. Routing protocols in WMNs can be classified as either proactive, reactive or hybrid [10, 30]. In proactive routing protocols, paths to all hosts are made known to a node, irrespective of whether the node can transmit at that specific time or not. In reactive routing protocols, a route to a recipient is only requested on demand, i.e. when a node wants to transmit at that time. In hybrid routing, a group of nodes may be configured as proactive and some may be reactive. Ad-hoc On demand Distance Vector (AODV) protocol is an example of a reactive routing protocol. Better Aproach To Mobile Ad-hoc Networks (BATMAN) is an example of a proactive protocol. Hybrid Wireless Mesh Protocol (HWMP) is an example of a hybrid routing protocol [10]. BATMAN is used in this research because of its ability to select quality links. A brief background of BATMAN is provided below.

BATMAN routing protocol is a proactive protocol and its main focus is to determine the next best single-hop to a neighbouring node. BATMAN was designed to carry voice traffic. Unlike other routing protocols, BATMAN does not find the complete path to the destination node, but uses very small packets, called Originator Messages (OGMs), to find the best quality link to a neighbouring node. OGMs are broadcast every second to inform neighbouring nodes about the sending node's existence [17, 24]. The neigbouring node that receives the OGM, first changes the address of the sender to its own address, then rebroadcasts the OGM to its neighbour until the entire network is flooded with OGMs or until the Time To Live (TTL) expires [24]. When a node receives its OGM, a birectional link check is performed to further check that the link can be used in both directions [17, 24]. BATMAN always keeps a table of information that contains the number of OGMs received in a node. The link-local neighbour that receives the highest OGMs, is then considered as the best route to the next single-hop neighbour [17, 24].

1.1.3 Voice packetization and overhead

VoIP is defined as a means of transporting voice packets over Internet Protocol(IP) with acceptable Quality of Service (QoS), or quality performance and affordable cost [29, 34]. The transmission of voice over an IP network is accomplished using three essential components: codecs, packetizer and playout buffer [4, 19]. The codec accepts analogue voice signals from the input source, and converts them into digital signals, compresses and then encodes them into encoded voice frames.The packetizer takes over and breaks down the encoded frames into a series of small equal-sized packets. Protocol headers assist in the delivery of VoIP traffic. Real-time Transport Protocol/User Datagram Protocol (RTP/UDP) and IP headers are attached to each packet, including signaling protocols, like Session Initiation Protocol (SIP) and H.323, that are responsible for initiating VoIP calls and terminating the connection between the sender and the receiver

[4, 19]. Voice packets are then transmitted to the play-out buffer to remove jitter that might have occurred during transmission. Finally, the packets are being decoded and depacketized back to analogue voice to be played at the receiver. Figure 1.2 illustrates the end-to-end transmission of a VoIP system.

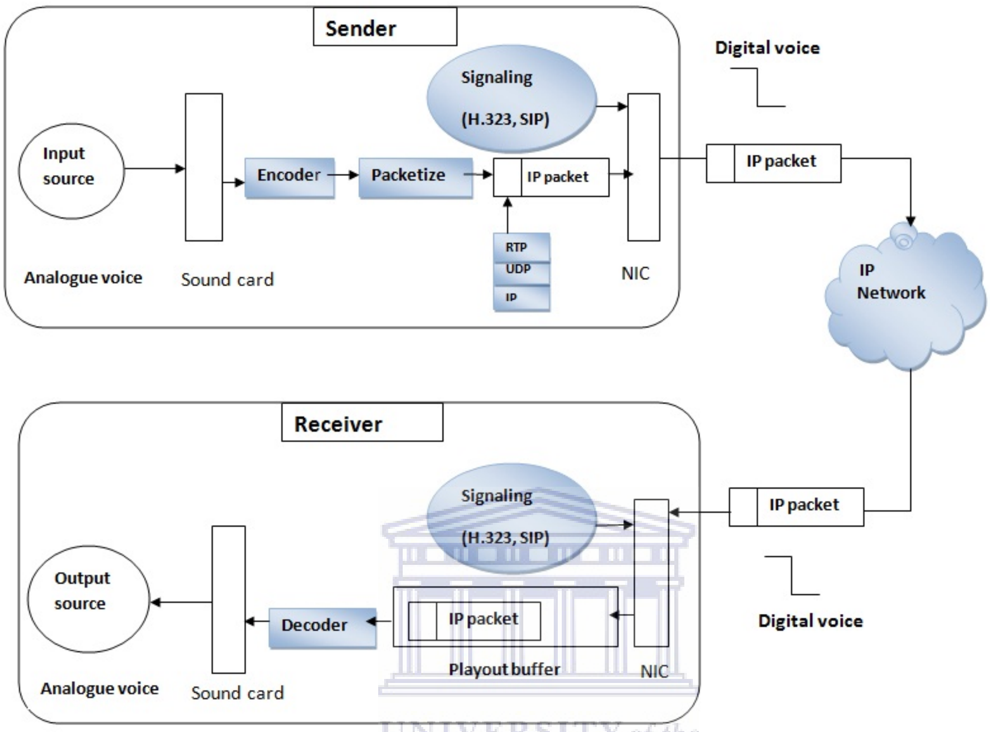

FIGURE 1.2 **VoIP system**
The figure illustrates the processes that are involved when voice is received from a user as analogue and then converted into frames, from frames to layer 3 IP packet structure, transmitted over the Internet and then converted back to analogue for the recipient. The VoIP headers that contribute to overhead are RTP, UDP and IP

1.2 Motivation

VoIP service has increased in popularity due to high Wireless LAN (WLAN) availability. VoIP is known for efficiently providing communication services, such as voice mail and voice conferencing. For instance, Skype has recorded more than 10 billion minutes of call time in its first year of deployment [11]. This tremendous increase in VoIP traffic is caused by the cost-effectiveness achieved by VoIP, the ease of deployment and the implementation of voice compression techniques with bandwidth sharing mechanisms. IP telephony has grown to the point/so much that it is now regarded as an alternative way of communication relative to the public switched telephone network (PSTN). VoIP over WLAN applications can be used at homes and offices, in both developed and developing countries, such as South Africa. Of particular interest to us are wireless mesh

VoIP projects like Village Telco (www.villagetelco.org). A village telco is a community based telephone network that is based on a suite of open source applications that enable entrepreneurs to set up and operate a telephone service in a given area, urban or rural. A village telco can be designed for a rural community with a collection of 802.11bg mesh routers, known as mesh potatoes, that use a Foreign Exchange Subscriber (FXS) port to connect an analogue phone to a VoIP network. VoIP over WMNs provides a cheap and convenient way of communication for low income earners. Cellular networks have become crucial in our daily lives, but for poor communities they are not affordable. Users in rural communities could use VoIP to make calls using mesh potatoes, monitored by village telcos, instead of cellular networks.

However, cheap and convenient VoIP over WMNs has known challenges, namely system capacity and system performance. Maintaining high quality VoIP traffic in WMNs can be difficult. According to [4], IEEE 802.11 based wireless LAN was not originally designed to support delay-sensitive voice traffic and more-over, IP was originally designed for data traffic, not voice. Therefore achieving high-quality in VoIP is a challenge [7]. Packet loss, delay and jitter are major causes of inefficient delivery of high-quality VoIP services [7, 11]. These three are caused by the physics of signal delivery and also high overheads of the Transmission Control Protocol/Internet Protocol (TCP/IP) stack. For instance, in popular voice codec G729a, a voice payload of 20 bytes, requires an additional 40 bytes of RTP/UDP and IP header per packet [11, 31]. On a WMN, with a 2Mbps link speed, the number of calls reduces from 8 calls in a single hop to one call after 5 hops [5, 11]. The Medium Access Control (MAC) layer has to spend precious CPU cycles resolving contention, as shown in Figure 1.3. Therefore, the Short Inter Frame Space (SIFS), Distributed Inter Frame Space (DIFS) and Acknowledgements (ACK) that are distributed after every packet, contributes to the total overhead. Thus call capacity decrease is caused by the transmission of many small voice packets over 802.11wireless mesh networks [11, 31]. Figure 1.3 illustrates the time spent when the MAC layer is resolving contention and the overhead involved for each packet transmitted. For a packet to be sent, it first has to wait for a DIFS. If the channel is idle, the packet can be sent, but if not, the packet enters the contention phase. Whenever a node attempts to transmit packets and the transmission is unsuccessful because the channel is busy, the contention window doubles, and therefore increases the waiting time for the packets which increases delay and thus increases packet loss. This means that each and every small packet has to go through this process during contention. DIFSs and ACKs are added for each packet and add increase to the overhead. This increases media utilization which decreases the throughput. Therefore, the main aim of this research is to improve the performance of VoIP over WMNs. Packet aggregation is used to increase throughput, which increases the number of supported VoIP calls in WMNs.

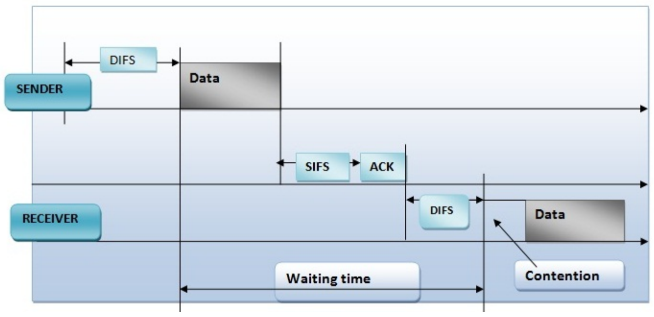

FIGURE 1.3 **MAC layer contention**
The figure illustrates the process that is involved during the contention phase for each packet transmitted. The waiting time for a packet always increases when the channel is busy and nodes are attempting to transmit. Therefore the ACKs of the MAC layer involved for each packet, increase overhead.

1.3 Research question and overall approach

The main research question is: **Can packet aggregation increase the number of supported VoIP calls over WMNs while maintaining high quality performance?** What follows, is the procedure followed to answer the research question.

Techniques from related work that were used to increase VoIP performance, were researched. It became evident that packet aggregation was one of the proven techniques used [5, 11, 31]. Therefore, packet aggregation was implemented in a Linux kernel to increase the number of supported calls. A forced delay approach, where packets have to be delayed for some time in order for aggregation to be possible, was used. Packet aggregation was implemented hop-to-hop and end-to-end, and then the two scenarios were compared to identify the approach with the better performance.

A small scale test bed consisting of four node desktop PCs, was designed at first and then a 9-node network, consisting of 7 mesh potatoes and 2 PCs. The performance testing tool, Iperf was used on both networks and generated VoIP traffic using a realistic VoIP profile to collect quality of service data. Baseline tests with VoIP traffic that *was not* aggregated were then conducted. This was followed by VoIP traffic testing that *was* aggregated with a kernel-level implementation to achieve optimal efficiency.

Results were analysed and the maximum number of supported calls from each test bed was determined by considering traffic characteristics, namely delay, packet loss and jitter. This thesis reports on the results, comparing baseline and aggregation results, and also hop-to-hop and end-to-end results. The next section provides an overview of the research process from related work to our conclusion.

1.4 Thesis outline

Chapter 2 presents related work. Packet aggregation techniques are introduced and the application of each technique to different networks, including wireless mesh networks, wireless LANs and wired networks are compared. Characteristics of VoIP traffic delay, jitter and packet loss are also introduced. These performance metrics are used to determine the quality of voice calls.

Chapter 3 describes the research methods and experimental design. The research question is presented in more depth. Methods regarding the implementation of aggregation and deaggregation modules, in order to execute the network test beds, that were used to evaluate the kernel-level packet aggregation in both hop-to-hop and end-to-end configurations, are discussed in detail.

Chapter 4 discusses the data collected from the hop-to-hop and end-to-end experiments, and the analysis of those results. The performance of each test bed is presented and then results between aggregation tests and the baseline tests are compared and analysed.

Chapter 5 concludes the thesis. A conclusion is drawn based on the results from Chapter 4. The limitations of this research are presented together with recommendations and future work for further research into packet aggregation on wireless mesh networks.

Appendices present published and unpublished work. Appendix A presents a work-in-progress paper that was published by the Southern Africa Telcommunications Networks Applications Conference (SATNAC) in 2010. Appendix B presents an unpublished paper which is essentially a 5-page summary of this thesis to be submitted to a conference and/or journal. The two mentioned papers are co-authored but this work is solely done by the thesis author.

Chapter 2

Related work

This chapter gives an overview of different packet aggregation schemes that can be used to improve VoIP performance, and explains how each of them are implemented. Section 2.1 introduces the implementation of different packet aggregation schemes for both wired and wireless networks, since packet aggregation is the main solution for increasing call capacity. Section 2.2 examines different characteristics of VoIP traffic, as well as the resulting metrics that are used to evaluate the performance of those packet aggregation schemes. Section 2.3 summarises the chapter.

2.1 Packet aggregation

Packet aggregation is defined as a means of combining small multiple packets together to form a larger packet (See Figure 2.1). In contrast, deaggregation is the separation of aggregated packets into their original form. Aggregation can be done at the IP layer, MAC layer and the Application layer [9]. Aggregation at the MAC layer is called frame aggregation and at the IP layer it is called packet aggregation [9]. Aggregation can be implemented in many different ways, depending on the requirements of the network. At the IP layer, aggregation is accomplished by combining IP packets. At the MAC layer, MAC frames are combined and at the application layer, audio frames are combined.

Aggregation at all mentioned layers is possible, although all have pros and cons. Figure 2.1 represents three packets being combined into one large packet. Instead of sending each small packet with its ACK and MAC headers, three combined packets can be transmitted with a single MAC and ACK header by means of aggregation. Packet aggregation does not only reduce overhead, but has an added advantage of saving time. As shown in Figure 2.1, it is evident that the amount of time required to send three packets, is twice the amount needed to send one large packet.

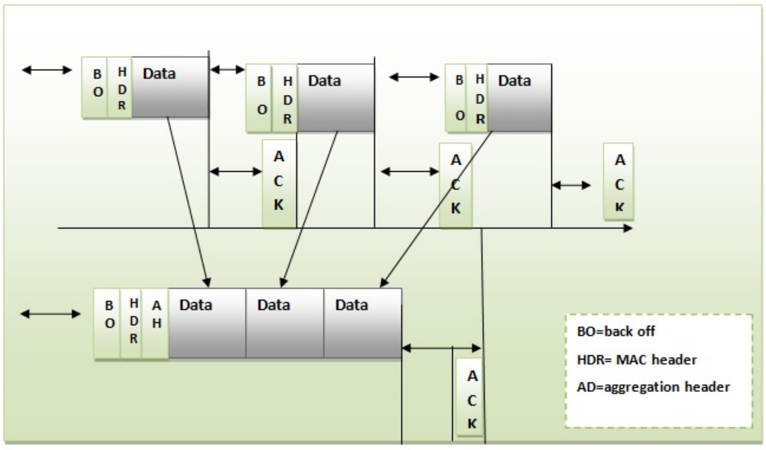

FIGURE 2.1 **Packet aggregation concept**
The top of the figure represents the aggregation of three packets into a single packet. The payload is represented by the word 'Data'. The headers that cause overhead are shown, e.g. the MAC header 'HDR'. The bottom part of the figure shows the result of the aggregation and the reduced overhead. This figure was adapted from an external source [20].

During aggregation packets need to be enqueued to be aggregated. This means of queuing is achieved using a method known as forced delay aggregation. In forced delay aggregation packets are marked with a time stamp and each packet is allocated a maximum amount of delay time. This means that after this delay time has expired, a node will begin to aggregate packets going towards the same destination [9]. Forced delay aggregation is presented in [9, 16]. According to [9], the maximum number of small packets to be aggregated in size is ruled by the Maximum Transmission Unit (MTU) which is 1500 bytes in wired and 23000 bytes in IEEE 802.11 wireless networks. VoIP traffic is sensitive to delay, therefore a packet that did not get aggregated within the maximum delay time, is immediately released from the waiting queue and transmitted unaggregated [9]. This prevents more delay to occur, which will in turn increase packet loss ratio. Therefore it is necessary to choose the correct maximum delay time. Aggregation can be done in two ways: end-to-end aggregation and hop-to-hop aggregation.

2.1.1 Hop-to-hop aggregation

In hop-to-hop aggregation, packets are aggregated from one hop to the next hop [11, 18]. Delay is introduced at every node while aggregation and deaggregation is performed. Aggregation is firstly done by the sending node and upon reception, by the neighbouring node. The packet will first be deaggregated and then aggregated when it is ready for transmission. This process continues until it reaches its destination. Hop-to-hop is known to cause higher delay, but proven to yield a better aggregation ratio (See Figure 2.2).

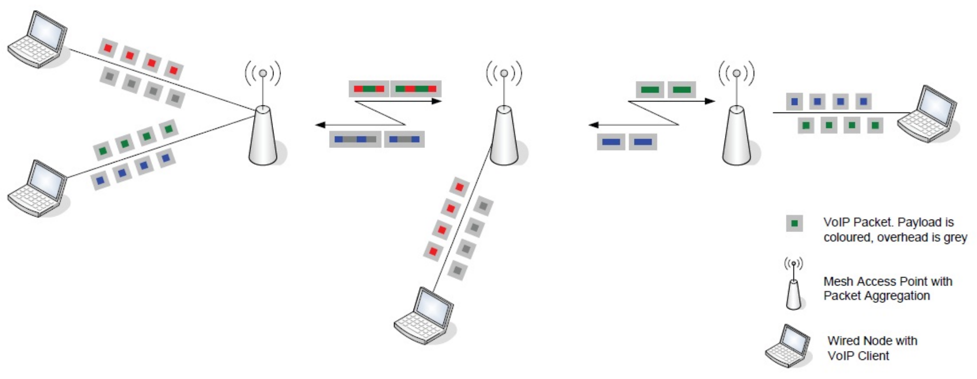
VoIP Packet. Payload is coloured, overhead is grey
Mesh Access Point with Packet Aggregation
Wired Node with VoIP Client

FIGURE 2.2 **Hop-to-hop packet aggregation**
The figures represent the concepts of hop-to-hop aggregation. Wireless nodes are configured with aggregation and deaggregation modules and as packets traverse throught the network, they are being aggregated and deaggregated at each node. This figure was adapted from an external source [18]

2.1.2 End-to-end aggregation

In end-to-end aggregation packets are aggregated only at the sending node and the receving node. In end-to-end, aggregation is done only for packets going towards a common destination [11, 18]. The receiving node deaggregates the packets. Other nodes that are intermediate, are responsible for forwarding the packets until they reach the destination where they will be deaggregated. In end-to-end aggregation, additional delay is only introduced once, at the source, thus end-to-end delay is reduced (See Figure 2.3).

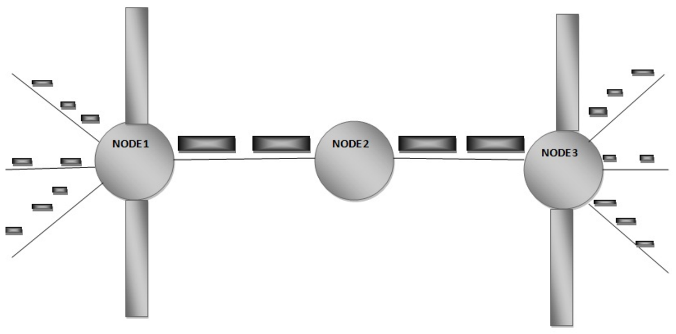

FIGURE 2.3 **End-to-end packet aggregation**
Packets are being aggregated as they are received by NODE 1 and they pass through NODE 2 until they reach their destination at NODE 3 where they are deaggregated.

2.1.3 IP packet aggregation schemes

Aggregation at the IP layer, which is layer 3, is known as packet aggregation. In IP packet aggregation schemes, implementation of aggregation and deaggregation is solely done at the network layer. The researchers below have used different approaches of aggregating IP packets.

Raghavendra et al. implemented packet aggregation at the IP layer by employing an IP based Adaptive Concatenation scheme (IPAC) [26]. IPAC is an end-to-end aggregation adaptive scheme where packets are aggregated or concatenated based on the quality of the route. The quality of the route is obtained by using the Weighted Cumulative Expected Transmission Time (WCETT) routing metric. The WCETT value is used to calculate the Maximum Concatenated Size (MCS) which is a value that the aggregated packet must not exceed, as well as the Maximum Concatenation Interval (MCI) which is the maximum delay interval that packets can be queued before they are considered for concatenation [26]. This work concluded that a good quality link can carry larger aggregated packets, while a poor quality link may drop the packets if it carries packets that are too large [26]. IPAC has been proven to perform well in high traffic loads where there are a lot of congested links. In that case, VoIP performance can improve significantly.

Castro et al. implemented a similar packet aggregation method at the network layer and proved that it increases call capacity [5]. It was implemented and tested in the ns2 simulator. The only difference is that hop-to-hop aggregation was implemented, which allows aggregation to be refined in-between the nodes, but its shortcoming is that it adds additional delay. Yet, it yields a better aggregation ratio [5]. The MAC layer utilization time was also studied and results proved that when aggregation is disabled, the channel is busy more often than when aggregation is enabled. An added benefit is that packet aggregation also reduces the time the MAC layer takes to resolve contention.

Aggregation of packets at the IP layer was also implemented at the network stack of the Linux kernel at [3]. The aggregation technique is the same as the representation in Figure 2.1, but differs in that it is implemented on a Linux kernel. Packet aggregation here was tested with the Ad-hoc On-demand Distance Vector-Uppsala University (AODV-UU) protocol and the implementation was accomplished by making use of queuing disciplines found in the Linux kernel. The idea behind this research was to implement packet aggregation in the Linux distribution of a cheap routing platform, Linksys WRT54GL. A method called Simple Packet Size Selection Protocol (SPSSP) was used to calculate the maximum number of packets to be aggregated. This was determined by the Signal to Noise Rasio (SNR) of every link. According to the researcher's conclusion, time did not allow them to either complete this adaptive part of using this SPSSP protocol, or to implement packet aggregation in the Linksys router, but it was proven that aggregating IP packets at the network layer, increased the number of supported VoIP calls.

VoIP optimization techniques have been studied in [11]. The study focused on the overall performance and real-time characteristics of a wireless mesh network that are affected by hardware capabilities, speed and complexity. In this research, the aggregation of IP packets is performed as represented in Figure 2.1, but forced delay is introduced only at the ingress node (receiver). At the intermediate nodes, the medium access delay is used, which resembles natural delay, meaning no additional delay is introduced. The results show that the combination of forced delay and medium access delay, kept delay to a minimum. The drawback of this approach is that it does not consider conditions in heavily loaded networks. In this research Destination-Sequence Distance Vector (DSDV) protocol is used with various metrics to select the five best routes. With DSDV, the whole path to destination is determined. This means that when one node goes down, packets may be lost. A route to the destination, consists of good and bad routes. Since aggregated packets are larger than normal and require good links, the quality of the routes will affect the aggregated packet. However, in this research, the compression of IP headers, known as Robust Header Compression (ROHC), coupled with packet aggregation, were proven to produce a high transmission rate of VoIP calls over unstable links. Unfortunately, according to [11], ROHC, has only been tested in single hop networks.

The study in [16] explores two packet aggregation techniques, namely forced delay aggregation and congestion triggered aggregation. AODV protocol was used to determine the next routing step for the packets. In forced delay aggregation, additional delay is used and packets are queued, based on the next common hop indicated by the AODV protocol. When enough packets arrive with the same next hop, such that the combined size of these packets is the same as the MTU of the outgoing link, then they are merged into an encapsulated packet and placed in a transmission queue to their respective destinations [16]. The forced delay aggregation is considered to be simple to implement and proven to yield a higher aggregation ratio, but its drawback is that it causes additional accumulative delay.

The second method in this study is congestion triggered aggregation. According to this method, packets are also queued, based on the next routing step, but a module is used to pull packets from the aggregator object after a previous aggregated packet has been transmitted. It has to wait for the interface to finish transmitting and then pull aggregated traffic from the aggregator. This method is proven to decrease delay and works best in congested networks, but packets that do not meet the minimum packet size of aggregation, still have to wait longer, since the aggregator has to determine whether there are any packets that can be aggregated.

2.1.4 Frame aggregation schemes

Frame aggregation is the aggregation of multiple MAC layer frames where implementation is done at the MAC layer of the network stack. Frame aggregation schemes can also be performed hop-to-hop. Hop-to-hop frame aggregation for VoIP was implemented in [21]. In this study the idea was to aggregate frames for congested traffic only. According to their simulation results, hop-to-hop aggregation schemes can improve VoIP performance and this approach can work for any traffic with small sizes. Figure 2.4 illustrates how multiple frames are combined.

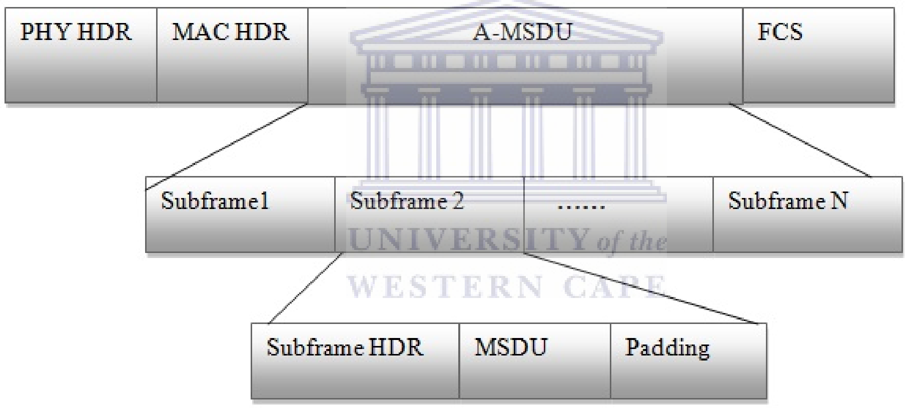

FIGURE 2.4 **Frame aggregation**
The figure illustrates the concept of frame aggregation where subframes are combined together to form one bigger frame called an MSDU.

Skordoulis et al. conducted research on the next-generation 802.11n wireless LAN and performed frame aggregation in the MAC layer in order to achieve high throughput and efficiency [28]. In this study, multiple MAC layer frames were aggregated to form one large frame. These MAC frames are differentiated into two: there is MAC Service Data Unit (MSDU) aggregation and MAC Protocol Data Unit (MPDU) aggregation. In MSDU multiple subframes are grouped or aggregated together to form a single frame, whereas an MPDU groups multiple subframes and combine them into an 802.11n header size. In MSDU, the maximum number of subframes to group, is guided by the maximum determined MSDU threshold. The maximum subframes to aggregate in an MPDU depends on the maximum number of packets found in the transmission queue. The transmission queue can hold up to 64 bytes of subframes. The researchers concluded that a better queue other than First In First Out (FIFO), will lead to better channel efficiency. The results proved that frame aggregation can reduce MAC layer overheads and thus increase the overall throughput.

Lin et al. has also done frame aggregation and optimal size adaptation for IEEE 802.11 WLANs at the MAC layer. In this study a frame size is calculated based on the probability of the Bit Error Rate (BER) [22]. In a high BER, the optimal frame size is estimated to be small and in a low BER, the optimum frame size is estimated to be large. Small multiple frames are aggregated up to the optimum frame size determined. However, this model is proven to achieve higher throughput for WLANs and single hops. Since hop-to-hop and end-to-end aggregation schemes can only be implemented if more than one hop exists, this means that these schemes cannot be applied in single hop networks. Therefore, this approach might not be applicable to multihop wireless networks.

2.1.5 Audio aggregation schemes

Audio aggregation is when audio frames produced by codecs are aggregated together to form a large aggregated audio. Audio aggregation is implemented at the application layer of the network stack [9]. Figure 2.5 is a schematic diagram of two audio frames being aggregated into one and sent as one packet of payload. The research discussed below, is work done on audio aggregation.

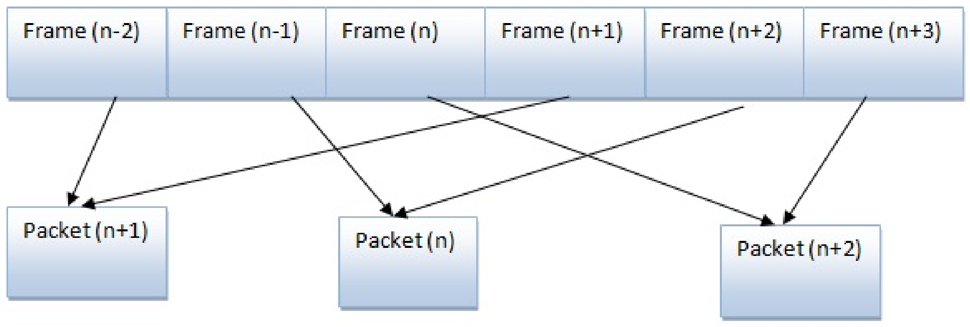

FIGURE 2.5 **Audio aggregation**
The figure illustrates the concept of audio aggregation: two frames are being aggregated, frame(n-2) and frame(n+1) are combined together into one frame and transmitted with packet(n+1), frame (n-1) and frame(n+2) are also combined together and transmitted with packet(n).

Garg et al. conducted a study on the limitations of IEEE 802.11 (a/b) in supporting VoIP calls over a wireless LAN and proved that VoIP performance is affected by a number of parameters. Some of these parameters include the type of codec used and the audio payload size, specifically when collision prevention techniques, such as Distributed Coordination Function (DCF), are present [12]. In this research calculations were performed to compare the differences between smaller audio frames and larger audio frames regarding different codecs. The analysis of the number of VoIP connections was based on the assumptions that one end-point of each VoIP call is a wireless client, while the other end-point is on the wired network. In this study the number of maximum connections of VoIP for each codec, was calculated using mathematical methods.

The calculations were done for three standard codecs, namely IT US G.711 a-Law, G.723.1 and G.729. In the G.729 codec for a payload of 10 ms audio payload, only 7 connections could be made by an access point. In G.711 with 20 ms payload, the number of VoIP connections was 12, and with 28 ms payload, the number of connections was 40 in a single hop. It it evident that the larger the audio frames are combined and sent as a stack, the more the number of connections increase. This study also proved that the choice of codec is very important and can limit the performance of VoIP. For instance, with G.711 codec, when the audio frame size is 40ms, the number of connections that can be supported is 21 connections, but on a G.729 codec, the number of connections is 28 with the same size of audio payload [12]. Therefore G.729 codec should always be the preferred codec for VoIP traffic. The conclusion has been drawn that the larger the payload per frame, the more the number of VoIP connections increase.

2.2 Voice traffic characteristics

QoS in VoIP has been identified as one of the major reasons why implementing VoIP applications is such a challenge in today's technology. In fact, Reynolds et al. states that to guarantee QoS in VoIP, is not impossible, but rather an engineering challenge [27] . In voice traffic, delay cannot be allowed, otherwise voice packets will be compromised, whereas in data traffic the communication is asynchronous. In reality an email message that got delivered one minute ago, will not make such a difference if delivered three minutes later. Packet loss, delay and jitter are major causes of inefficient delivery of high-quality VoIP services and these are typically the characteristics of VoIP. A further discussion of these characteristics of VoIP traffic [7, 9, 34], follows in the section below.

2.2.1 Delay

When Voice packets are transmitted from the sending to the receiving node, some packets take more time than expected to be transmitted. This is known as delay in VoIP [7]. Delay can be experienced in three ways: accumulation delay, processing delay and network delay [33]. Accumulation delay is caused by the processing of voice samples. According to Dely et al. , accumulation delay depends on the speech codec used, because a codec buffers and then compresses audio samples to one frame [9]. Accumulation delay is the actual time needed to group encoded frames into one packet. Network delay is the time required to perform buffering of voice packets and then simultaneously transmitting them across a communication network to their destination [33]. Delay at both the sending and receiving node is approximately the same, therefore the International Telecommunications Union (ITU) recommends that the maximum acceptable delay for VoIP user applications, should be between 0 to 150ms (See Section 2.2.4) [33].

2.2.2 Jitter

Jitter is defined as a variation of packet delay, that is caused by queuing lengths, traffic contention and the use of different routes throughout the network. Delay is different for every packet and if delay keeps varying, it becomes cumbersome to maintain delay at an acceptable rate. Therefore delay needs to be kept constant and this is achieved through the use of jitter buffers [7]. Jitter buffers temporarily store all incoming packets so that they can be delivered to the receiving node in a steady way [7, 9].

2.2.3 Packet loss

When speech is transmitted over the network, it is expected that it arrives at the receiving node just as it was sent. Unfortunately, due to possible overflowing of queues, some packets arrive late and this further causes packets to be discarded during transmission. This is called packet loss [14]. Packet loss is caused by a number of factors, including changes in the routing tables, routes and also the available resources [14]. In 802.11 lost or discarded packets cannot be retransmitted, because voice is synchronous and therefore critical to time. Instead, a technique called Packet Loss Concealment(PLC) handles lost packets. PLC tries to patch or replace the missing gaps by means of approximating the missing voice frames in the packet [9]. Some packets are also declared as lost upon arrival to destination node, when the network is experiencing too high delay or high jitter due to negative factors, such as congestion. According to [9, 14], the maximum tolerable or acceptable packet loss was originally 5%, but the voice sound irritated users

[14]. The maximum packet loss ratio cannot be determined as it was done for delay, because packet loss ratio depends on the codec and the packet size used [9]. Therefore packet loss ratio and delay is estimated in relation to the codec used. A mechanism called E-model is used for the estimation of packet loss. Recommended by the ITU for different codecs, *http://standardsdocuments.tiaonline.org/tia-tsb-116-a.htm*, the E-model and its usage of estimating packet loss ratio and maximum tolerable delay, is defined at section 2.2.4 below.

2.2.4 Voice performance metrics

High quality VoIP performance can be achieved, provided that some metric evaluates the performance of any new VoIP application. A performance metric that is very familiar and widely used to measure the quality of a call, is the Mean Opinion Score (MOS) which is defined as the arithmetic average of opinion, based on the user's perspective of voice quality [6]. When MOS is "excellent", it is given the highest quality score of 5. "Good" is a 4, "fair" is given a 3, "poor" given a 2 and "bad" is given a 1 [6]. A subjective measure, known as the E-model, produces results that determine the MOS. The E-model is proposed by the ITU. It is a tool that provides a method that measures user satisfaction of voice quality between two connections [6] . The output of the E-model is the Rating factor, known as the R-factor and it is calculated as follows [6]: **R=Ro - ls - ld - le - A** where:

- **Ro** is the signal-to-noise ratio,
- **ls** is the sum of speech impairment occurring simultaneously during transmission,
- **ld** is the sum of all the delay impairments occurred,
- **A** is the Advantage factor that serves to improve the total R-value by means of counterbalancing the impairments.

The R-factor output, calculated according to the above formula, is weighed with a scale of 100. 100 represents the best voice quality and anything below 60, is declared as the worst voice quality. The R-factor output then determines the MOS and the relationship is as follows (see Table 2.1).

The ITU has presented the E-model output scale comparisons on voice quality as perceived by the user (see Figure 2.6) . The R-factor values determine the maximum delay, including jitter, that produce acceptable voice quality. According to ITU recommendations, the maximum delay for a VoIP network using G.729a codec, is 150ms. The graph

R-factor	Quality of voice rating	MOS
90 <R <100	Best	4.34 - 4.5
80 <R <90	High	4.03 - 4.34
70 <R <80	Medium	3.60 - 4.03
60 <R <70	Low	3.10 - 3.60
50 <R <60	Poor	2.58 - 3.10

TABLE 2.1: The R-factor table
This is the R-factor table representing how MOS is related to the R-factor. The highest value of R-factor determines a high MOS and thus yields the best quality voice. This table was adapted from an external source [6].

at Figure 2.7 shows that when G.729a packet loss is at 2 % and less, the R-factor value in the Y-axis is above 75. When delay is 150ms and when G.729a packet loss is 3 and 4 %, then the R-factor value is below 70. This implies that many users will be dissatisfied, according to the E-model scale represented in Figure 2.6 . It means that for each end-to-end connection, the packet loss and delay should be kept under these constrains. The authors of [3, 9] and [11] have done packet aggregation and tested performance of VoIP under these constraints. Their results produced acceptable voice quality, while the number of handled calls also increased.

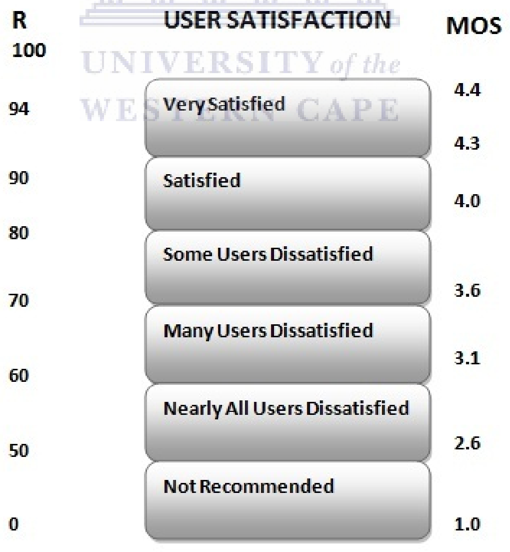

FIGURE 2.6 **E-model output scales**
The figure illustrates the E-model output scales as recommended by the ITU, and how they are related to each other. When the R-factor value is high, the MOS is also rated high, representing excellent voice quality *[http://standardsdocuments.tiaonline.org/tia-tsb-116-a.htm]*. This ensures user satisfaction for many users, whereas the low R-factor values are not recommended because of the low quality voice they offer.

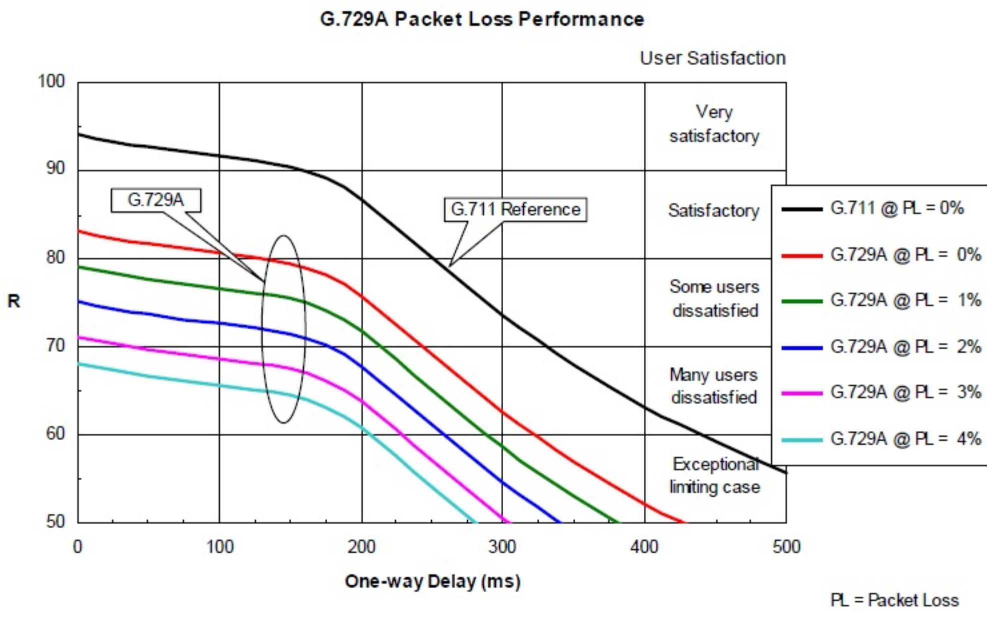

FIGURE 2.7 **R-factor vs. delay and packet loss**
The figure illustrates a graph representation of how low R-factor values affect the quality of a call. When delay is above 150ms, the quality of the call starts degrading and then packet loss increases, thus decreasing the overall performance of VoIP in that particular network. This figure was copied from [9].

2.3 Summary

A variety of researchers proved that packet aggregation is able to improve and increase call capacity, whether the network is a single wireless LAN or a multi-hop network. This can save resources and channel busy time. Aggregation was done at the MAC Layer, by means of aggregating sub-frames, at the network layer, by aggregating IP packets, and also at the application layer, by aggregating audio frames. Aggregation was proven to decrease VoIP overhead tremendously. Delay, jitter and packet loss were discussed, as well as minimum standards for these VoIP characteristics, in order to provide acceptable voice quality, in accordance with ITU standards. Packet aggregation was implemented for both hop-to-hop and end-to-end. Both methods increased call capacity. Most packet aggregation techniques employ forced delay to allow packets to be aggregated, but other studies use media access delay, in conjunction with forced delay, causing overall delay to reduce. Packet aggregation was implemented with different protocols, including AODV-UU and DSDV protocols. Most researchers opted for AODV for routing protocol, since it has been proven to perform better than DSDV protocol. Thus the protocol used, also has an effect on routing packets, especially when such packets are aggregated.

Chapter 3

Methods

This chapter presents the methods to achieve and measure a solution to increase VoIP performance on wireless mesh networks. The chapter presents the research question and justifies the research methods chosen to answer that question. Thus, the chapter details the experimental design employed, including the implementation of aggregation and deagreagation mechanisms into the Linux kernel, and the test beds used to collect and measure performance data.

Section 3.1 presents the primary goal of this project and how it is to be accomplished. Section 3.2 presents the research questions and the strategies used in response. Section 3.4 introduces the experimental designs and the tools used. Section 3.6 summarises the chapter.

3.1 Adding packet aggregation to BATMAN-adv

Packet aggregation in VoIP has been proven to increase call capacity by decreasing the VoIP protocol overhead in many different networks. Because channel quality is important in VoIP performance, the routing protocol used, also has an effect on VoIP performance. Castro et al. has evaluated packet aggregation performance on different channel quality links and found that more packets can be aggregated in a good quality link, while less may be aggregated in a bad link [5]. This also means that a routing protocol that can respond to node failure, detect bad quality links and be able to determine good quality links immediately, will make packet aggregation more efficient.

In this project the aim is to perform packet aggregation with a new improved routing protocol, known as BATMAN. Unlike other routing protocols, BATMAN does not find the complete path to destination node, but only finds the best quality link to a neighbouring node in the right direction [17, 24]. This prevents alternative route lookup or route discovery when there is node failure. Therefore in this project, the primary goal is to first implement packet aggregation and then test it on a wireless mesh network, running the BATMAN protocol. The secondary goal is to implement and test packet aggregation in a mesh potato network running BATMAN protocol. This will serve to improve packet aggregation ratio, since BATMAN has a way of determining good links differently and eventually this will improve VoIP performance.

3.2 Research question

The main research question is: **Can packet aggregation increase the number of supported VoIP calls over wireless mesh networks while maintaining high quality performance?** The research question can be answered as follows: An efficient algorithm will be designed to increase VoIP performance by not increasing accumulative delay and maintaining high quality performance. A software-like design will be developed that will be automated and will be implemented in Linux. Implementing in Linux, prevents adding too much extra delay, as a system that is already part of the Linux kernel code will be used. This system is called queuing discipline (qdisc). With qdisc you are allowed to arrange packets any way you desire. This allows the privilege of not adding excess delay, since qdiscs are designed for packet handling.

To increase VoIP performance, overheads will be reduced by performing packet aggregation. Thus efficient delivery of aggregated packets will be accomplished by using the BATMAN protocol which is proven to be able to select good quality links. This implementation will be placed within hops, i.e. in every hop, packets that are to be transmitted are aggregated and packets received by a node are deaggregated as presented in Section 2.1.1. This means that resources allocated for a two-hop network, will be the same resources as for a four-hop network, since packet aggregation also reduces the traffic rate during transmission in the network.

3.3 Research methods

An empirical study is a way of gaining knowledge by way of doing direct observation or gaining experience [13]. An empirical study can be either quantitative or qualitative. Quantitative methods are known to be based on numbers and statistics. In this section, a discussion on how quantitative methods were used to collect results, is presented. The procedure followed to design the system will also be presented.

3.3.1 Quantitative methods

Quantitative research is carried by experimental designs and non-experimental designs [8]. Network performance tools are described as a strategy to collect and analyse data in quantitative research. Therefore, quantitative methods were chosen to answer the research question, because these tools can be used to conduct performance testing of the system before and after modification. After execution, network performance tools produce statistical information, therefore this statistical information will be used to evaluate the performance of the system [8].

Quantitative methods, by means of experiments using simulation, were performed by [9, 11] as seen in Section 2.1. Network performance tools are able to produce statistical graphs depicting the performance of the system. The network was designed first and then a number of traffic flows, using the network performance tool, Iperf, were injected. More information on this tool is given in Section 3.4.

3.3.2 Research design

Research design refers to the methodology of the research project. Successful research must undergo stages that involve identifying the problem, until the stages of results presentation and analysis are reached. In this research, four stages were followed (See Figure 3.1):

- Problem identification: Our study was focused on VoIP over WMNs. Past journals and conference papers were studied in a bid to identify areas of improvement.
- Literature review for VoIP in WMNs: Literature reviews of different packet aggregation techniques and their application to different networks with different routing protocols were gathered. VoIP was analyzed in terms of its performance when applied specifically to WMNs. New solutions were identified in order to raise its standard of performance.
- Implementation and building a testing environment: This stage starts with the implementation of the solution, which is packet aggregation in the Linux kernel, and then deploying it to work with BATMAN, setting up and designing two network topologies, which include configuring Iperf, using a realistic VoIP profile, and then loading packet aggregation modules to the nodes destined for aggregation. This stage allows the performance testing to be conducted efficiently.
- Testing results and analysis: Iperf performance tool produces statistical results in data, as well as graphs for each test. This means that at this stage the results of

interest are presented. These include accumulative throughput and jitter, including delay and packet loss. These characteristics of VoIP traffic, are measured and analysed, ensuring that they meet the standards mentioned in Section 2.2.4 of the related work.

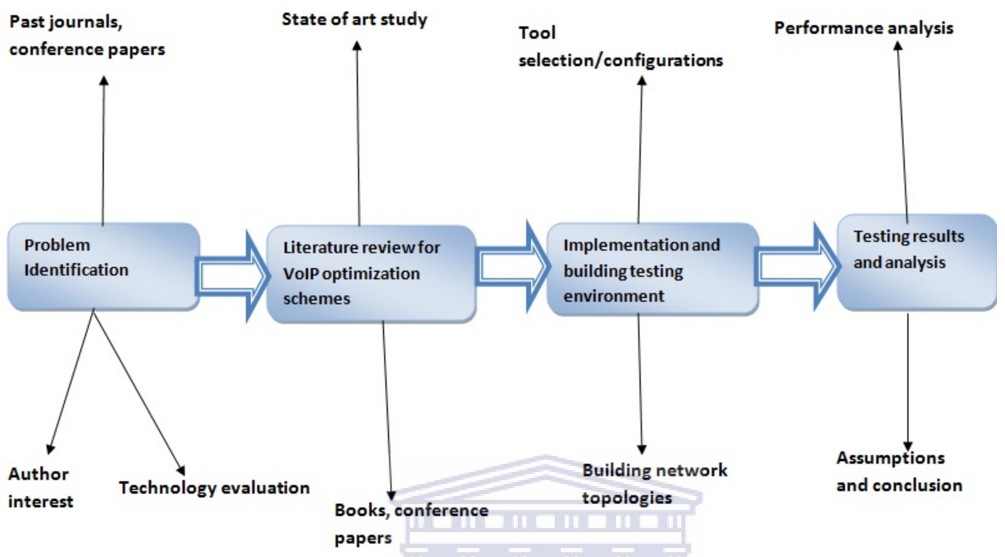
Past journals, conference papers
State of art study
Tool selection/configurations
Performance analysis
Problem Identification
Literature review for VoIP optimization schemes
Implementation and building testing environment
Testing results and analysis
Author interest
Technology evaluation
Books, conference papers
Building network topologies
Assumptions and conclusion

FIGURE 3.1 **Research methodology**
Illustrates the methodology followed when conducting research. It includes identifying what still needs to be done, finding relevant related work, designing and testing packet aggregation solutions in different WMN topologies and then, results analysis and presentation.

3.4 Experimental design

This section discusses the practical solution of the research project, which is the implementation of aggregation and deaggregation. In this project, aggregation as a queuing discipline and deaggregation using the netfilter subsystem found in the Linux kernel, were implemented. A discussion about the evaluation of aggregation and deaggregation modules, using the Iperf network performance tool to get results, will follow. Finally, the parameters measured from the results are described.

3.4.1 Aggregation implementation

Our aggregation module is implemented as a simple queue, known as the queuing discipline(qdisc). Every network device has queues which are used to accept packets for transmission. In Linux kernel language, these queues are called qdiscs and they form a major component of the Linux traffic control code in the Linux kernel. There are ingress(inbound traffic) and egress(outbound traffic) queuing disciplines [2, 3]. Qdiscs can have what we called classes and filters. Classes are queues that are connected to the qdisc itself, to further handle traffic. Each class consists of one queue and a class can have many subclasses [2, 25]. There can be classless qdiscs and classful qdiscs: classless are the ones that do not have a class, i.e. the qdisc handles all the necessary queuing of packets, whereas classful qdiscs divide the traffic for classes connected to it [2, 25]. Classless qdiscs are considered to be simple, straightforward and easy to implement. Examples of classless qdiscs found in the Linux kernel, include First-In-First-Out (FIFO), Stochastic Fairness Queuing (SFQ) etc. (http://tldp.org/HOWTO/Traffic-Control-HOWTO/index.html). Classful qdiscs are the complicated qdiscs because of the inclusion of classes and subclasses. Examples in the Linux kernel, includes Hierarchical Token Bucket (HTB), Hierarchical Fair Service Curve (HFSC) etc . In this research, a simple classless qdisc was created. This qdisc accepts packets from the interface, in this case wlan0, by means of the enqueue method. The Linux qdiscs have a standard way in which they are created. They have functions that they use to control how they handle traffic and these functions have to be similar for all the qdiscs in the kernel. This means that to create a new qdisc, it must have the standard functions in order to be accepted by the kernel. The standard functions are as follows [2, 25]:

- *enqueue* accepts packets and places them in the queue of the qdisc.
- *dequeue* removes a packet that is eligible for transmission from the queue.
- *requeue* inserts a packet back to the queue exactly in the location it was before it had been dequeued.
- *drop* function drops one packet from the queue.
- *init* initializes the qdisc and prepares it for the handling of packets.
- *change* changes the configuration of the qdisc.
- *reset* reinitializes the qdisc into its initial state.
- *destroy* removes all classes, filters and free resources assigned to it.

The important function in this research is the dequeue function, because that is where aggregation is performed. As packets are being dequeued, they are being aggregated. The aggregation method is as follows: First of all, packets need to meet a minimum value or size before they are considered for aggregation. The value is currently set to 200 bytes. This value is important, because packet aggregation seeks to reduce voice overheads by minimizing the number of small packets and sending bigger packets. It is thus clear that by allowing small packets, the goal is not achieved. The size that the aggregated packet must not exceed, is set to 1500 bytes. This is done to prevent packet loss issues.

The next step was to set a timeout value. That is the amount of time packets are delayed, so that during that time, packets can be aggregated. The time is set to 5ms. If enough packets that are enqueued, meet the minimum determined value when they reach the dequeue function, they are immediately aggregated and sent for transmission. Packets that did not get aggregated within the maximum delay time, are immediately released and sent as they are. This is because if they are delayed longer, the chances of packet loss may increase and the recommended maximum delay value might not be met, as seen in Section 2.2.4.

When packets are aggregated, a new large socket buffer, that will hold all the packets combined, is created. A socket buffer (skb) is a data type used in the Linux kernel. Its primary assignment is to hold network packets [3]. During aggregation, new IPv4 and MAC headers are created. The old MAC header is destroyed, but the old IPv4 header is kept. The old IPv4 header cannot be discarded, because it contains the IPv4 address of each packet, while a MAC header can easily be replaced. The newly created IPv4 header is responsible for holding the identification number, which is a value that each aggregated packet must have, so that the packet will be recognized in the deaggregation module. This value is randomly selected and is currently set to 253. Figure 3.2 represents the original packet structure before aggregation, as well as the combined packets after aggregation.

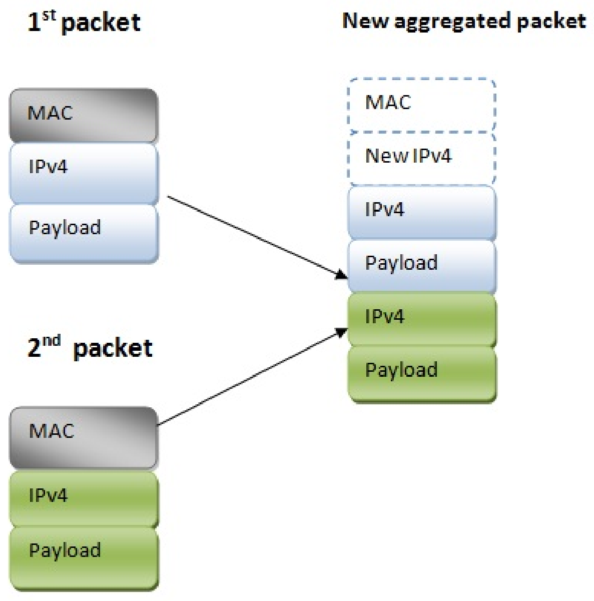

FIGURE 3.2 **Aggregation packet structure**
The figure illustrates how aggregation is accomplished. Packet 1 and packet 2 are combined and placed into a new bigger skb. The old MAC headers, from packet 1 and 2 are destroyed and the new MAC is created. An additional IPv4 header is created, to hold the aggregated packet, while the payload and the original IPv4 header is kept.

3.4.2 Deaggregation implementation

Deaggregation is the process of separating packets that were bundled or combined together to return them to their original format. In this project, deaggregation was implemented in the Linux kernel, by using the available kernel code. Figure 3.3 presents the concept of deaggregation in the kernel.

Hooks that are found in the netfilter subsystem of the Linux kernel, were used to implement the deaggregation of packets. Hooks are defined as points or locations in the network stack, where packets traverse [32]. Netfilter is a subsystem that is defined as a framework, and uses hooks for packet handling (http://netfilter.samba.org). In simple terms, netfilter is a kind of subsystem that decides how packets are to be handled, by a specific protocol. The netfliter subsystem allows the kernel to register or to connect to it. This enables the subsystem to know and listen to different hooks for each routing protocol. The hooks are differentiated, with numbers that are ranked in priority, from the first to the fifth priority . When the kernel is registered to the netfilter subsystem, each packet will pass through the subsystem. The subsystem will then check if any of the packets are registered to any of the hooks. The hooks found in the Linux kernel code are as follows [32] (see Figure 3.3):

- The NF_PRE_ROUTING is the first netfilter hook. It is assigned to either declare the packets as stolen, i.e. if they are taken over by another module, or declare them as accepted, if no module or function took them.
- NF_IP_LOCAL_IN is the second netfilter hook, which handles packets that are destined for the local host.
- NF_IP_FORWARD is the third netfilter hook, that forwards packets destined for another interface in the network.
- NF_IP_LOCAL_OUT is the fourth netfilter hook, which is for packets from a local process, on the way out to their destination.
- NF_POST_ROUTING is the fifth netfilter hook, that handles packets just before they hit the wire.

In this project, the deaggregation module is positioned or placed in the first netfilter hook, the NF_PRE_ROUTING. Since NF_PRE_ROUTING is the first priority hook from the interface that handles packets, this allows us to be able to hijack all aggregated incoming packets, and cause them to enter the deaggregation module immediately. The deaggregation module is the function that is registered to this first priority hook. As soon as the packets enter the deaggregation module, they are immediately unpacked or unaggregated and inserted back to the networking stack. When packets are aggregated in the dequeue function of the aggregation module, they are stamped with a number, which is the identification value as mentioned in Section 3.4.1. Each aggregated packet must have this value when passing this hook. This means that only packets with this identification value will enter the deaggregation module. As soon as a packet is taken over by the deaggregation module, the NF_PRE_ROUTING hook has to return a message to the netfilter subsystem, indicating that a packet has been taken over by another module. This is done through the use of codes, returned by the hook as follows [32]:

- NF_ACCEPT - keep the packet, continue handling the packet as normal.
- NF_DROP - drop or discard the packet.
- NF_STOLEN - the packet has been taken over by a module.
- NF_QUEUE - queue packet for user space.
- NF_REPEAT - call this hook function again.

In this case, once the packet with the identification value has been taken over by the deaggregation module, the code that is returned is NF_STOLEN. Other packets on the network without the identification value are all accepted, with the code NF_ACCEPT. Figure 3.4 represents the location of the deaggregation module, i.e. how it has been hooked in the netfilter subsystem of the networking stack, and illustrates the handling of packets as soon as they arrive at the device.

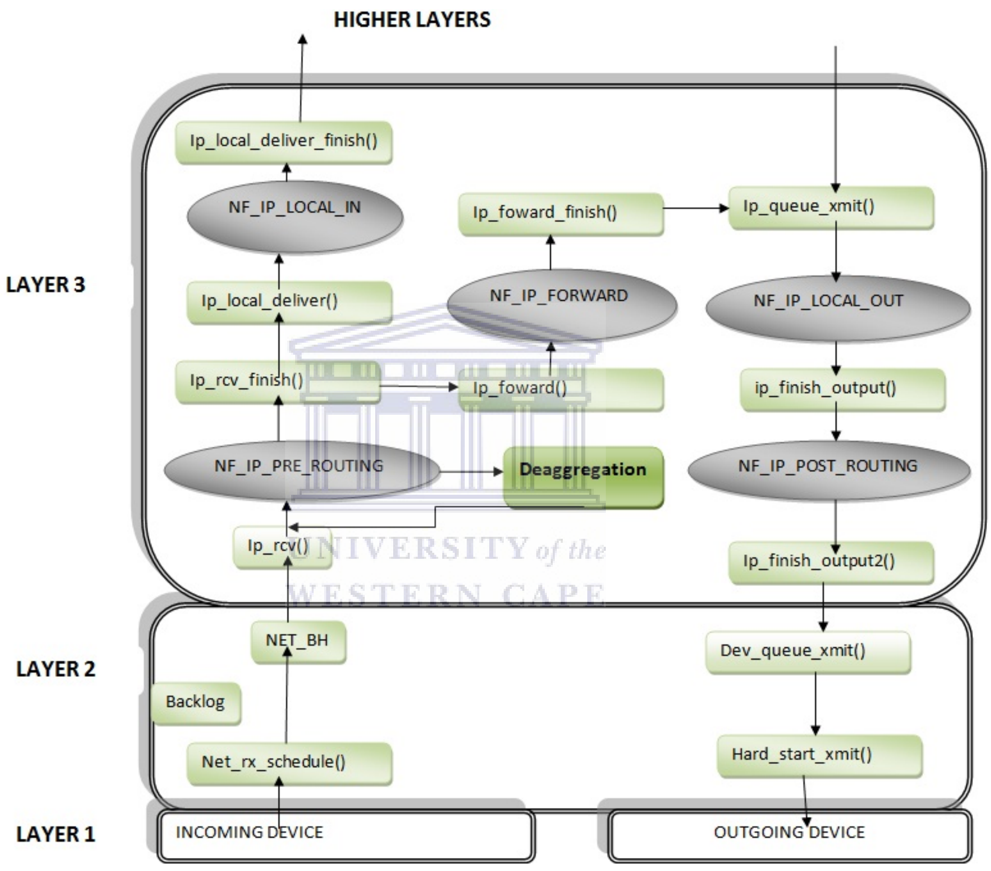

FIGURE 3.3 **Deaggregation module location in the network stack**
The figure illustrates the journey of a packet through the network stack. A packet is received by an incoming device at layer 1, it passes through to layer 2 and is routed to layer 3. As soon as the packets arrive at the NF_PRE_ROUTING hook, the deaggregation module takes over aggregated packets, deaggregate them and insert them back. The transmission then continues to higher layers.

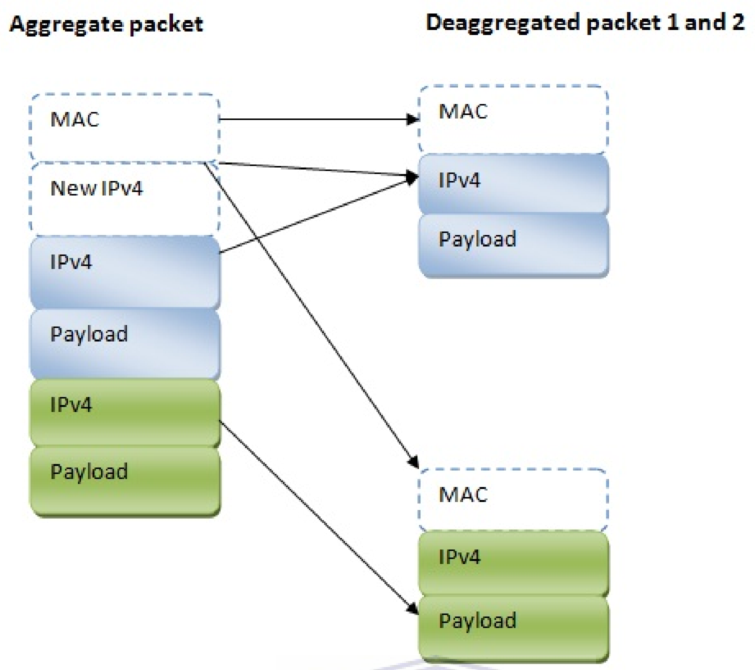

FIGURE 3.4 **Deaggregation packet structure**
The figure illustrates the concept of deaggregation. As packets are received in the network stack, those that have the desired identification value will be separated into their original format. In this case, the payload of packet 1 together with its original IPv4 header and MAC header, will be copied and restored. The same procedure applies to packet 2.

3.4.3 Kernel configuration

The aggregation and the deaggregation modules were compiled as standalone loadable modules. These modules must be loaded into the kernel and activated so that they can be used. To load the modules, *insmod* was used. *Insmod* is a program used to insert a module into the kernel (http://linux.die.net/man/8/insmod). All existing Linux qdiscs have to be activated, and are activated by being attached to the desired device interface. The deaggregation module does not need to be activated after being loaded, but since the aggregation module is a qdisc, it has to be activated. The aggregation module is activated using a Linux application tool, called Traffic Control (TC). TC is a tool used to attach a qdisc to network device interfaces. A new simple add-on-like module, belonging to the TC tool, was created. This add-on is a helper method that is used to activate the aggregation module. The name of the aggregation module is ”aggregate”. This means that the aggregation module must have the name ”aggregate” and the add-on must also have ”aggregate” in its code. TC belongs to the iproute2, which is a collection of utilities TCP/IP networking, in traffic control in Linux (http://www.linux-foundation.org/en/Net:Iproute2). The add-on module is placed inside the TC folder and compiled together, in order for it to be recognized. TC simply helps to attach the aggregate module to the wireless interface *wlan0*. The following statement activates the aggregation module: *tc qdisc add dev wlan0 root aggregate max 1500 min 200*. The statement can be explained as follows: The name of the qdisc is "aggregate", assigned to be "root". When the module is configured as root, it means that as soon as packets hit the device, the qdisc aggregate immediately handles all traffic. The qdisc is added on device interface *wlan0*. The maximum number of packets to be aggregated is 1500 bytes and 200 bytes is the minimum.

3.4.4 Traffic generation and data collection

Iperf (Intelligent Performance Prediction) network performance tool was used during the experiments. Iperf is a tool that measures network performance by measuring the throughput. It creates User Datagram Protocol (UDP) and Transmission Control Protocol (TCP) streams of data [15]. Iperf produces reports of accumulative throughput, which are used to analyse the performance of a system. The performance parameters described previously, jitter, packet loss and delay, are also produced from reports generated by Iperf. Iperf reports were used to evaluate the system behaviour for the purpose of quantitative research. The main task of this experiments is to evaluate the performance of the system. UPD trafffic is generated from node to node to measure the quality of VoIP traffic in wireless mesh network test beds. The aim is to measure how VoIP quality degrade when the number of calls increases. In this experiment, two network scenarios were created: baseline tests (without aggregation), and aggregation tests (with aggregation). To answer the research question, the performance of packet aggregation must be evaluated in terms of the number of supported calls. This can also be referred to as the number of supported flows. The are many codecs that are used in voice communications. Each has different requirements in terms of tolerable packet loss and delay. Since this research focuses on the performance of VoIP, G.729a codec which is mostly used in VoIP, is explored. According to [11, 21], VoIP traffic is modeled using the ITU G.729a voice codec. Therefore this voice codec is acceptable for voice traffic. This is evident from its popularity and the fact that it is mostly used in popular VoIP phones, such as Zxel Prestige [11]. Castro et al. and Ganguly et al. have used this codec for their study of VoIP traffic in WMNs [5, 11]. A flow or call is considered to be supported, if the call has voice quality that has maximum delay, including jitter less than 150 ms, and maximum packet loss ratio less than 4 %. This is a an acceptable quality for the G.729a codec (see

Section 2.2.4). Therefore, these standard values are used in this research to determine the number of supported calls from the reports produced by Iperf.

Baseline tests, in which packet aggregation was not activated, were conducted first. Another round of tests were also conducted. In these tests aggregation module is activated, and deaggregation module loaded. In order to test the performance of the system, a realistic VoIP profile was used, which is proven to be approximately the same as a normal VoIP conversation [23]. Studies have shown that the average VoIP conversation is on average 180 seconds [23]. This means that a person at station A will talk for 60 seconds, pause for one second, talk for another 60 seconds, pause for two seconds and talk again for the last 60 seconds. Station B then replies back for another 180 seconds the same way. Therefore 180 seconds was selected for the length of each test. This means that Iperf was set to generate UDP packets for a period of 180 seconds, from sender to receiver and another 180 seconds, for the receiver to reply. Iperf can be set as a server (sender) or client (receiver). For instance, in the hop-to-hop test bed, in figure 3.6, Section 3.5.1, A will be set as a server and B will be set as a client, when node A is communicating with node B.

The default VoIP payload for a G.729a VoIP codec is estimated to be 20 bytes, and when the RTP/UDP/IP headers are included, that adds up to 60 bytes of VoIP payload [11]. Therefore Iperf was configured to generate UDP packets of 60 bytes each, for 180 seconds at a time. Studies have proven that as the number of hops increases, the number of supported calls decreases [11]. This means that packet aggregation must be tested in a test bed that consists of more than one hop. Therefore, the hop-to-hop and end-to-end test beds, were configured as a two-hop network. This was accomplished using node B as the node that only forwards traffic to node C and D, in the hop-to-hop test bed. The link from node A to node C was terminated, to ensure that the only way to node C was through node B. This means it took two hops for traffic to be transmitted from node A to node C. In this way, packet aggregation was tested for more than one hop, since WMNs are multi-hop networks. Iptables were used to cut the connection between the nodes by means of filtering out traffic. For instance, node A rejects or filters out all traffic coming from node C. Iptables is an application that allows the administrator to test, maintain and inspect the rules in the IPv4 tables (http://linux.die.net/man/8/iptables). This is also done for sending traffic from node A to D and vice versa. From A to C,C to A, A to D and D to A, total four times, which means the traffic was increased up to 80 times. Traffic generated four times, is also the same as four flows of traffic, therefore 80 times will equal 80 flows. By increasing the number of flows, a more congested network was created, which helps to determine the maximum number of supported calls the network can support. The same procedure was followed for the end-to-end test bed in figure 3.7, Section 3.5.1. Connection between PC1 and PC2 was terminated, so that the only way to PC2 was through any of the mesh potatoes.

When testing the performance of packet aggregation with the above mentioned configurations, it requires that the experiments have to be done step by step. A series of steps that illustrate the whole procedure involved in the testing stage, is presented. The procedure is shown in Figure 3.5 below:

- The first step involves designing the hop-to-hop and end-to-end WMNs test beds inside a laboratory building, in a conducive environment for running tests.
- The second step includes configuring and installing Iperf in all four nodes and in the two PCs of the nine-node test bed, setting up the length of the test, the size of the packets and the bandwidth.
- The third step begins with the baseline test by generating VoIP traffic 80 times, loading the aggregation and deaggregation modules respectively, and starting to run aggregation tests.
- The fourth step is where the analysis of baseline tests vs. aggregation tests begins.
- The fifth step involves separating quality calls from poor calls obtained from the aggregation and baseline tests.
- Finally, the comparison of baseline tests and aggregation tests begins, where hop-to-hop is compared to end-to-end. The results are presented in graphs, and a conclusion is drawn.

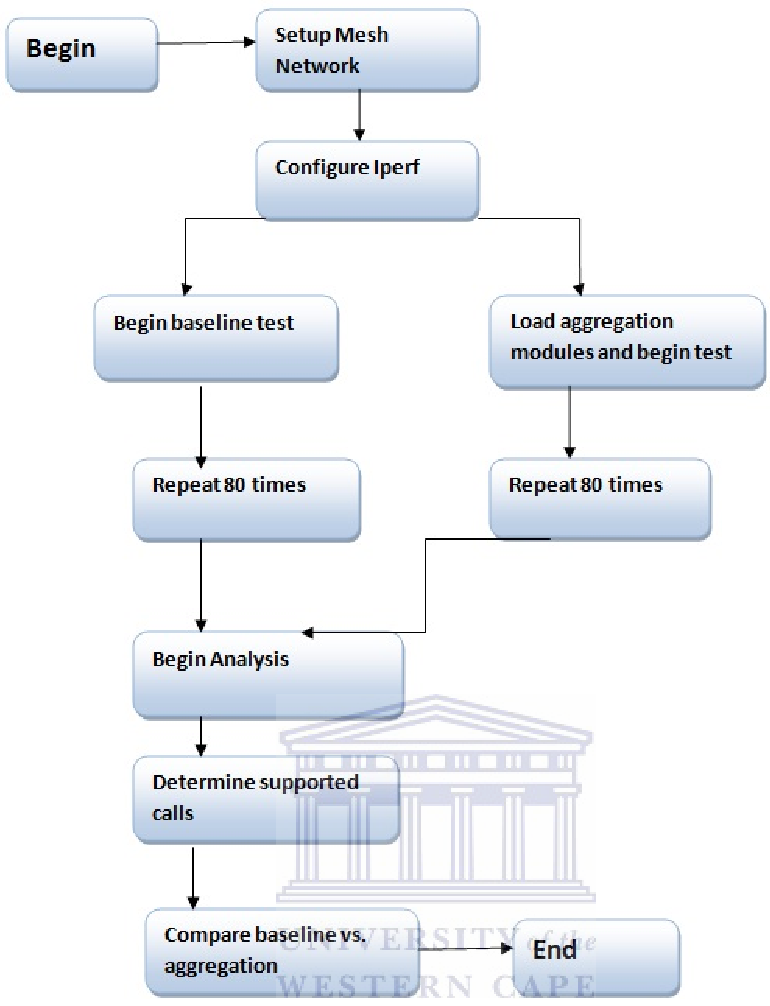

FIGURE 3.5 **Testing procedure**
The figure represents the step by step stages followed during the testing process. The network test beds were firstly designed, the statistics were configured to smoothen the test environment, the packet aggregation method was executed, and results were collected and analyzed.

3.5 Experimental scenarios

This section presents the hop-to-hop and end-to-end test beds that were used to test packet aggregation.

3.5.1 Hop-to-hop test bed

A four-node test bed was designed to test hop-to-hop aggregation. The four-node test bed is represented in figure 3.6. This test bed consists of four PCs running Ubuntu 10.4, with Linux kernel version 2.6.32.35. BATMAN-adv 2010.0.1 was used as the routing protocol.

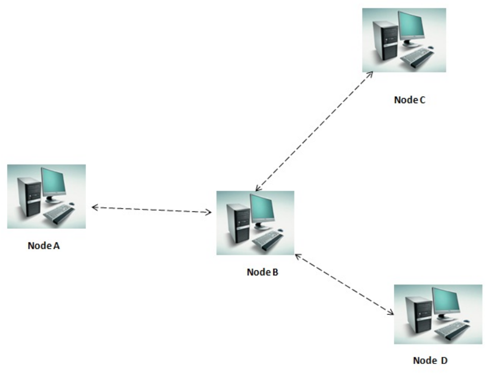

FIGURE 3.6 **Hop-to-hop test bed**
The figure represents the four-node test bed of four PCs. Node A can only communicate to node C, via node B, but node A cannot communicate directly to node C. Same applies to node D, node D can only communicate to node A via node B, because node B is configured to only forward traffic.

3.5.2 End-to-end test bed

A nine-node test bed was also designed to test end-to-end aggregation. It is represented in figure 3.7. The nine-node test bed consists of two PCs and seven mesh potato devices. BATMAN-adv 2011.0.1 protocol was used on this test bed. The differences in BATMAN-adv versions between the four-node and the nine-node, is due to the fact that the mesh potato devices were all running BATMAN-adv 2011.0.1. Therefore on the two PCs, BATMAN-adv 2011.0.1 had to be installed, so that there could be communication between the mesh potatoes and the PCs.

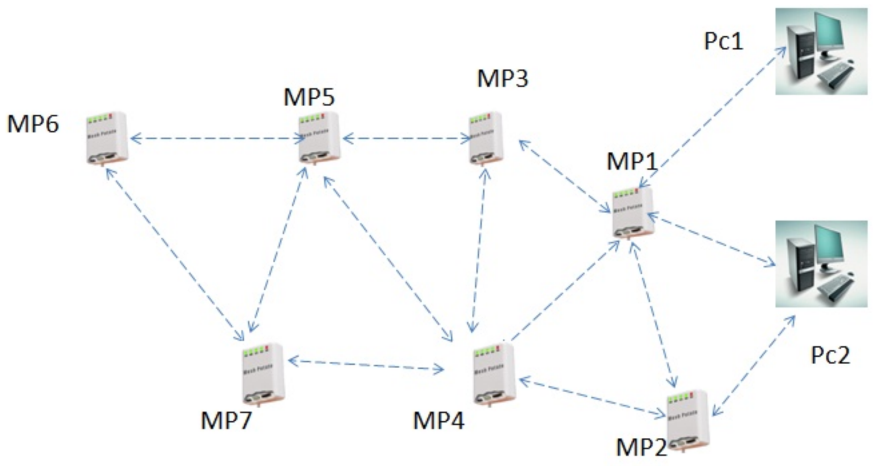

FIGURE 3.7 **End-to-end test bed**
The figure represents the end-to-end test bed of seven mesh potato devices, and two PCs. PC1 can transmit packets to PC2, via any of the mesh potatoes, or any path which is indicated by BATMAN as a quality link, but PC1 cannot communicate directly to PC2. The same applies to PC2.

3.6 Summary

It is possible to achive high quality VoIP performance in VoIP over WMNs, but reseachers state that it will be an engineering challenge [7]. This chapter discussed the primary goal of this project, which is to implement packet aggregation and test it in a WMNs, running BATMAN protocol. The research question in Section 3.2 was presented and discussed in detail. It was answered using quantitative research methods. Iperf is a network performance tool, selected to evaluate packet aggregation implementation in this research. Iperf produced results that were then analyzed using VoIP performance standards set by ITU. This chapter also presented the hop-to-hop and end-to-end test beds that were used to test packet aggregation, with a more in-depth explanation of how each test bed was configured and used.

Chapter 4

Results and discussion

This chapter presents a primarily graphical representation of the results obtained from Iperf. It presents the performance of the WMNs, in terms of packet loss and jitter, including delay and throughput. Hop-to-hop and end-to-end aggregation results are presented and analysed. The number of supported calls are determined and presented. Unaggregated tests results are compared with aggregation tests results, and the behaviour of the overall system is discussed. Section 4.1 presents the hop-to-hop mesh test bed results, and Section 4.2 presents the end-to-end mesh potato test bed results, and the analysis of those results. Section 4.3 summarises the chapter.

4.1 Hop-to-hop test bed results

During the testing process, Iperf reports the packet losses, jitter and throughput in detail. Therefore, graphs are used to present the performance of aggregation and non-aggregation tests. They present the performance of VoIP traffic obtained from hop-to-hop test bed, with and without aggregation.

Packet loss is defined as discarded packets that were not received by the receiving node [14]. Packet loss is presented first from 10 flows to 80 flows, as shown in the graph in figure 4.1. The figure shows the packet losses experienced during the experiments. The graphical representation of packet loss in this figure, clearly shows a significant difference when aggregation is used and when it is not used. In the graph, it shows that without aggregation, the packet loss is constant at average 3 %, for the first 20 flows, but from 30 flows and above, the packet loss starts increasing. This indicates that the traffic is becoming too congested for the network to handle. When looking at the aggregated traffic, it is observed that packet loss is acceptable for up to 60 flows, which indicates a

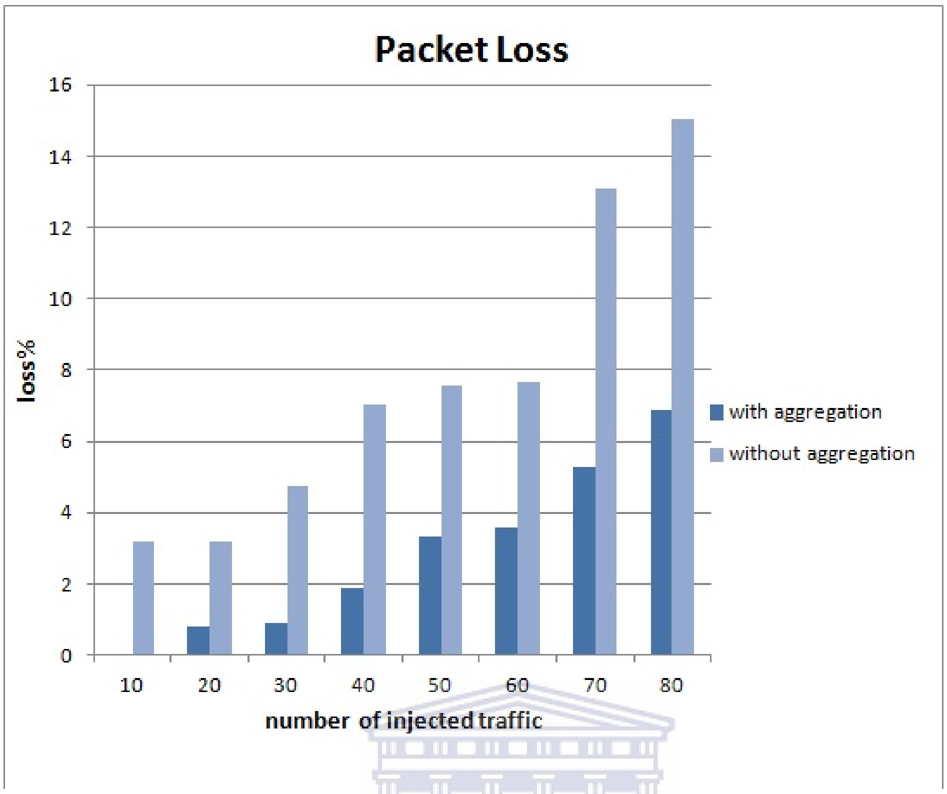

FIGURE 4.1 **Hop-to-hop packet loss**
The figure illustrates a bar graph representing average packet loss, with and without aggregation. The packet loss without aggregation is gradually increasing from 30 to 80 flows, and with aggregation the packet loss appears to be less than without aggregation.

significant improvement on the performance of VoIP traffic as compared to when packet aggregation is not used.

The graph in figure 4.2 presents jitter results received from the hop-to-hop test results. Jitter is the variation of packet delay that is caused by queuing lengths, traffic and the use of different routes throughout the network [7]. In figure 4.2 below, it is observed that from 10 to 60 flows, jitter is on average approximately 0.2ms, for unaggregated traffic, but for aggregated traffic, jitter is approximately 0.17ms. It is also observed that jitter seems not to differ much between aggregated traffic and unaggregated traffic. The reason for this is that firstly, with aggregation, packets needs to be delayed for some time for aggregation to be accomplished, and secondly, different routes in the network cause jitter to be high, as Collins mentioned [7]. In the hop-to-hop test bed, represented in figure 3.6 in Section 3.5.1, there are no different routes in the network. Node A can only send traffic to node C, through node B and no other route exists that can be used to send traffic to node C. As a result, there is a minor difference in

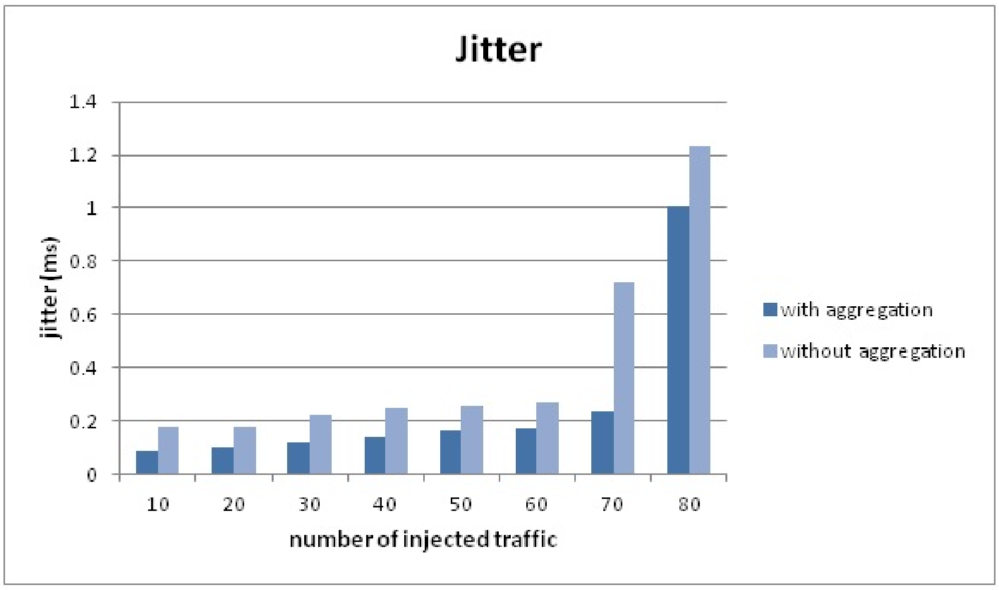

FIGURE 4.2 **Hop-to-hop jitter**
The figure illustrates the jitter that is obtained from the hop-to-hop test bed, between aggregated and unaggregated traffic. Jitter appears to be consistent for the first 60 flows, with and without aggregation, but jitter without aggregation appears to be higher than jitter with aggregation. Flow 70 and 80 shows that jitter is increasing, due to increasing number of injected traffic flows.

jitter between aggregated and non-aggregated traffic. Even though packets were delayed during aggregation, aggregation tests outperformed the non-aggregation tests, because jitter in aggregated traffic is less than jitter in unaggregated traffic, as seen in the graph. According to studies, during contention, the MAC layer spends time resolving the traffic, therefore, with smaller packets of unaggregated traffic, the time spent is noticeable and it adds up to the total jitter [11].

Throughput is the data transfer rate or the amount of data that can be transfered in a given time. Throughput obtained from the results is presented by the graph in figure 4.3. When media utilization decreases, throughput increases. This means that when the transfer rate is high, many packets are delivered to respective recipients successfully. The throughput results presented in the graph in figure 4.3, shows that as the number of injected traffic flows increases, from 10 to 80 flows, the throughput also increases significantly. Higher throughput is achieved for aggregated traffic than unaggregated traffic. This means that a higher aggregation ratio is achieved, which implies that many packets are aggregated, and immediately transmitted successfully. According to Iperf reports,as seen in the throughput graph, when throughput is 0.12 MBytes/sec with aggregation, the number of packets transmitted in 180 seconds, is on average 21.5 MBytes. Without aggregation, when throughput is approximately 0.10MBytes/sec, the number of bytes transmitted in 180 seconds is 18.6 MBytes. These results show a

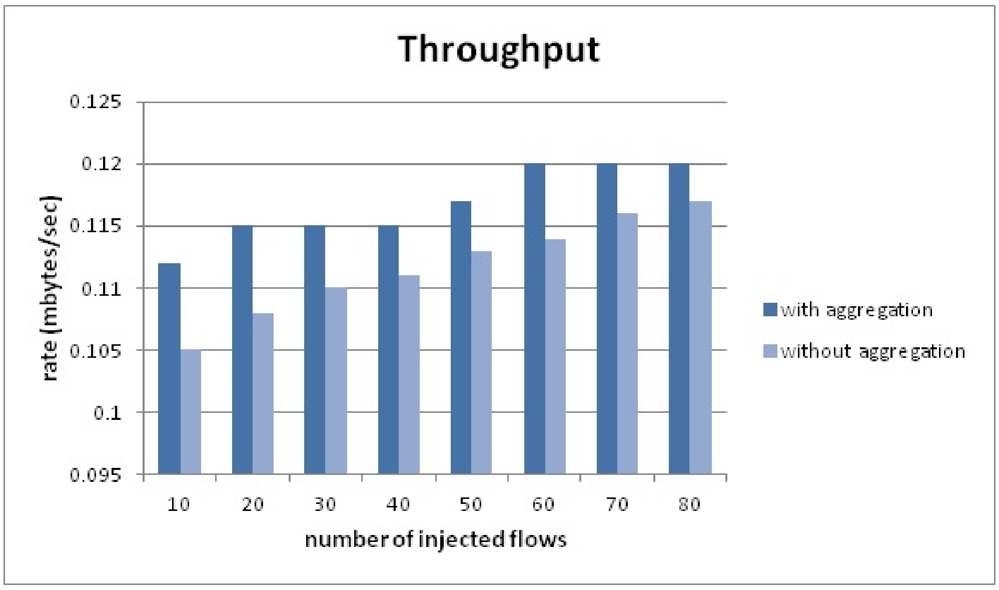

FIGURE 4.3 **Hop-to-hop throughput**
The figure represents the accumulative throughput from the hop-to-hop test bed, with and without aggregation. With aggregation, throughput appears to be consistent for the first 40 flows, but from 60 to 80 flows throughput increases. Without aggregation throughput also increases, but is less than that of aggregated traffic.

significant increase in the throughput achieved and the delivery of voice packets, when aggregation is used, and when it is not.

Next, the number of supported calls are presented, with performance parameters, jitter, delay, and packet loss taken into consideration. The ITU standards were used to determine supported calls, as mentioned in Section 3.4.4 of chapter 3. ITU recommends total packet loss less than 4 %, for voice quality to be acceptable to the user, and jitter, including delay to be no more than 150ms. Section 2.2.4 has in-depth information of these standards. This means that a VoIP flow is called a supported flow or supported call, if the packet loss is less than 4 %, and if jitter, including delay, is less than 150ms.

Table 4.1 presents the number of supported calls without aggregation. In unaggregated traffic, as shown on the results, it is observed that the network can support 20 flows, but when the number of injected flows increases from 30 to 80, the network is unable to support these flows. This is because the packet loss ratio increased, to above 4 %, which indicates that many packets were not delivered, therefore they were immediately discarded. High packet loss ratio is due to the fact that the network became congested, which in a practical scenario is when many people begin to make calls simultaneously. This increases media utilization, and without aggregation, this results in heavy packet losses. When this is the case, many users will not be satisfied with the voice quality,

Table 4.1: Supported calls from the hop-to-hop test bed without aggregation.

WITHOUT AGGREGATION			
No of injected flows	**Packet loss(%)**	**Jitter (ms)**	**No of calls supported**
10	3.2	0.1786	Supported
20	3.2013	0.1818	Supported
30	4.72	0.2264	Not supported
40	7.01	0.247455	Not supported
50	7.57	0.2545	Not supported
60	7.65449	0.2677	Not supported
70	13.082	0.7212	Not supported
80	15.03	1.23046	Not supported

and some users may not receive connection at all. From 10 to 20 traffic flows, it is observed that the packet loss is between 3.2 % to 3.2013 %, which represents R-factor values of 60 to 70, and a MOS score of 3.1 to 3.6. The MOS and R-factor values are shown in figure 2.6 of Section 2.2.4. These MOS score values mean that many users will be dissatisfied, and some users satisfied. This means that those users who are satisfied, are experiencing high quality calls, whereas for the rest, the quality is low but not poor. In the performance of VoIP traffic from 30 to 80 flows, packet loss ranges from 4.72 % to 15.03 %, which is extremely high. This means that the R-factor is from 0 to 60, indicating a MOS of 2.6 to 1.0, meaning that all users will be dissatisfied at this point, because the quality is unacceptable. Jitter is below 150ms, which is acceptable, but packet loss has a limit.

Table 4.2: Supported calls from the hop-to-hop test bed with aggregation.

WITH AGGREGATION			
No of injected flows	**Packet loss(%)**	**Jitter (ms)**	**No of calls supported**
10	0.02211	0.112	Supported
20	0.8135	0.115	Supported
30	0.8807	0.115	Supported
40	1.8564	0.115	Supported
50	3.3406	0.117	Supported
60	3.5798	0.12	Supported
70	5.2604	0.12	Not supported
80	6.900027	0.12	Not supported

Table 4.2 presents the number of supported calls with aggregation, obtained from the hop-to-hop test bed. The aggregation tests' results, show that the number of supported calls increases tremendously, compared to the results without aggregation. According to the results in this table, VoIP flows from 10 to 80 have jitter that is less than 150ms, and from 10 to 60 flows, packet loss is less than 4 %, which ITU tolerates. Therefore,

the network was able to support 60 flows of traffic with aggregation. At 10 to 40 flows, a conclusion can be drawn that these are high quality VoIP calls, because at these flows, packet loss ranges from 0.02211 % to 1.8567 %, which is less than the packet loss of 3.3406 % to 3.5798 %. This is because, according to the ITU perfomance graph, packet loss that is less than 2 %, represents high quality VoIP calls, but those above 2 % and less than 4 %, those are medium quality calls, whereas anything above 4% is the worst.

The performance graph presented by ITU in figure 2.6 of Section 2.2.4, indicates that packet loss from 0.02211 % to 1.8567 %, has the R-factor value that is from 80 to 90. This gives a high MOS score of 3.6 to 4.3, which indicates that many users will be satisfied. The goal is to have a high MOS score, which represents the highest quality of a VoIP call. The highest MOS score is 4.4, as explained in Section 2.2.4. Packet losses of 3.3406 % to 3.5798 % have R-factor values from 60 to 70, which indicate a MOS of 3.1 to 3.6, meaning that many users will be dissatisfied, and a few satisfied. During the last injected flows, which range from 70 to 80, the packet loss was above 4 %, which means those are poor quality calls. They are therefore not supported. The poor quality of these flows, may be the result of decreased accumulative throughput. Due to increased delay, many packets were sent unaggregated. Thus a higher aggregation ratio was not achieved at these flows.

When aggregation tests' results are compared with unaggregation tests' results, it is evident that with aggregation, 40 more calls were achieved than without aggregation. This means without aggregation on the hop-to-hop test bed, only 25 % of the number of calls are supported, but with aggregation 75 % of calls are supported. Therefore packet aggregation increases the number of supported calls by 50 %, as compared to the number of supported calls without aggregation. This proves that packet aggregation can reduce VoIP overheads.

4.2 End-to-end test bed results

This section represents end-to-end test bed results obtained from Iperf reports. Packet aggregation was implemented end-to-end on the nine-node mesh potato test bed. The aggregation and deaggregation modules were loaded on PC1 and PC2 (see Figure 3.7 in Section 3.5.1), meaning that PC1 and PC2 were the only nodes aggregating and deaggregating VoIP packets.

Figure 4.4 presents the average packet loss obtained from 10 to 80 flows, for testing VoIP traffic generated from PC1 to PC2 and vice-versa. According to the graph in figure 4.4, it is clear that with aggregation, packet loss is higher than without aggregation, but

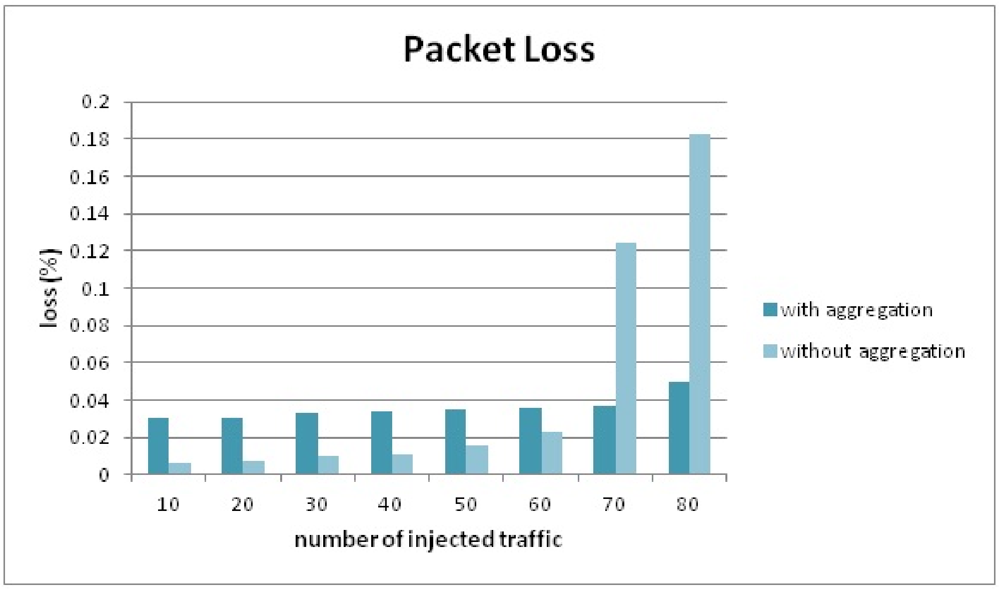

FIGURE 4.4 **End-to-end packet loss**
The figure illustrates the graph representing the average packet loss, with and without aggregation, when aggregation is implemented at end nodes.

the packet loss ratio with aggregation is consistent. Packet loss from aggregation tests appears to be higher only for the first 60 flows, and from 70 to 80 flows, the packet loss is less than without aggregation. This is because on the end-to-end test bed the traffic was not congested, because it was only PC1 that was sending traffic to PC2, and there are many alternative routes to PC2. The packet aggregation module is designed in such a manner that for packets to be aggregated, they must reach a minimum value before they are considered for aggregation. The value is currently set to 200 bytes (see Section 3.4.3). Therefore, packets may have to wait in queues for more packets if there is less traffic in the network, in order to reach the minimum value. If packets wait longer, the timer expires, which means some packets may arrive late. Some may not arrive and will therefore be considered as lost. Without aggregation, packet loss increases as the number of injected flows increases. Without aggregation, packet loss ratio increases from 0.01 % to 0.18 %, but with packet aggregation, packet loss is consistent at approximately 0.03 %. From 70 to 80 flows, packet loss is higher without aggregation than with aggregation. This means that without aggregation the quality of the calls at 70 flows, will be worse than the quality of the calls at 10 flows. Whereas with aggregation, the quality of the calls at 10 flows, is the same high quality as at 70 flows, which implies high quality VoIP calls were achieved with aggregation, in all 80 flows.

Figure 4.5 represents jitter, including delay for the 80 flows. The graphs show that jitter increases from 10 flows to 80 flows for unaggregated traffic. With aggregation, the average jitter is very low, compared to unaggregated traffic. In the related work

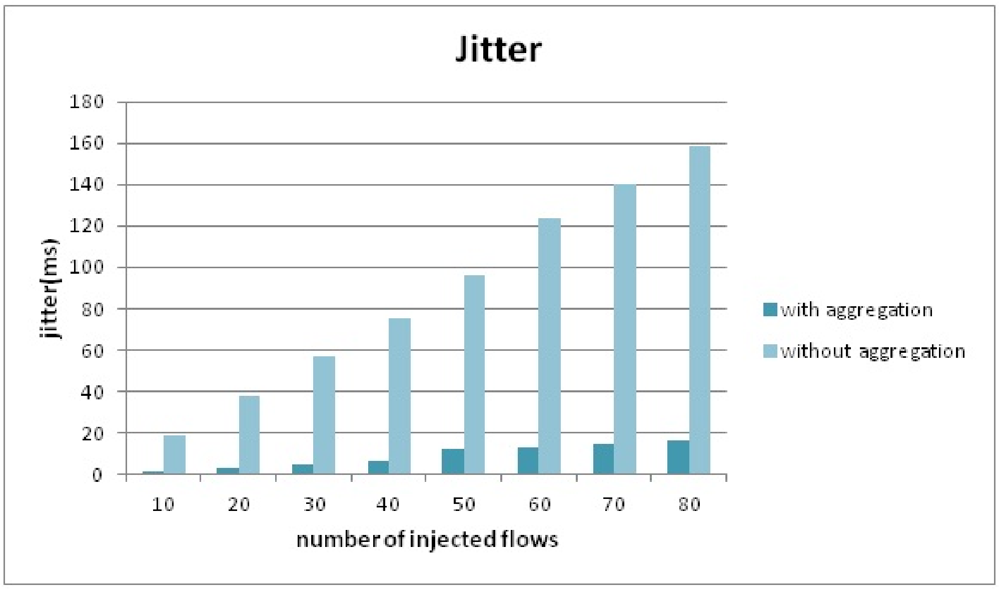

FIGURE 4.5 **End-to-end jitter**
The figure represents the jitter values of the end-to-end test bed with and without aggregation.

in Section 2.2, it is mentioned that jitter is caused by queuing lengths, contention and the use of different routes. When packet aggregation is not used, small packets are transmitted over the network. This increases the queuing lengths and the network contention. Therefore, the jitter adds up to high levels, but packet aggregation reduces the number of packets on the network, which reduces network contention.

Packets can be delayed. As long as they arrive, they are not considered as lost. They are only considered lost if they are not received at all by the receiving node. Jitter, which is the variation of delay for each packet, does not affect packet loss, but affects throughput. That is why without aggregation, jitter increased, but packet loss as shown in figure 4.4 was less. It is also observed that with aggregation, jitter was less than without aggregation, even though packets had to be delayed for some time for aggregation to be possible. This shows the effectiveness of packet aggregation in WMNs. It means that even though the end-to-end network was not congested as the hop-to-hop network was, the aggregation module was able to aggregate packets.

In figure 4.6, throughput is presented as collected from the Iperf reports. According to the results presented, extremely high throughput was achieved when traffic was aggregated , compared to when it was not. With aggregation, throughput is at average 0.12MBytes/sec. Without aggregation, it is approximately 0.3MBytes/sec. This means that at 0.3MBytes/sec only 5.54MBytes of packets were transmitted in 180 seconds. With aggregation, 0.12Mbytes/sec throughput was achieved, and 21.4MBytes of packets were transfered in 180 seconds. The difference between 21.4MBytes and 5.54MBytes

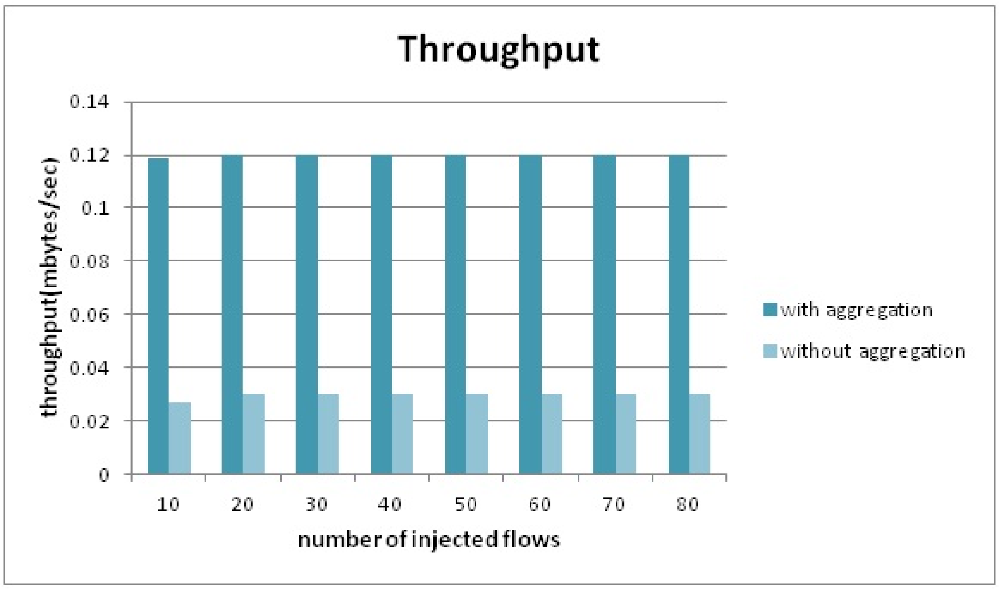

FIGURE 4.6 **End-to-end throughput**
The figure illustrates the graphical representation of the accumulative throughput from the end-to-end test bed.

is very significant, which means that with aggregation, 15.86MBytes more packets were transfered in 180 seconds, than without aggregation. This shows that packet aggregation reduces media utilization, because aggregation allows the transfer of bigger packets, which ensures the network is less busy, resulting in the saving of resources.

The number of supported calls, were then determined, based on the output of the end-to-end test bed experiments. The results are presented in Table 4.3 (without aggregation) and Table 4.4 (with aggregation).

TABLE 4.3: Supported calls from the end-to-end test bed without aggregation

WITHOUT AGGREGATION			
No of injected flows	**Packet loss(%)**	**Jitter (ms)**	**No of calls supported**
10	0.00646	18.5773	Supported
20	0.00725	37.8193	Supported
30	0.00954	56.765	Supported
40	0.01069	75.6743	Supported
50	0.01568	96.13	Supported
60	0.02266	123.208	Supported
70	0.12455	140.522	Supported
80	0.183	158.292	Not supported

Table 4.3 represents the number of supported flows, for the end-to-end mesh potato network test results (without aggregation). The average jitter and packet loss ratios were again used to determine the number of supported calls according to the ITU standards.

Table 4.4: Supported calls from the end-to-end test bed with aggregation

WITH AGGREGATION			
No of injected flows	**Packet loss(%)**	**Jitter (ms)**	**No of calls supported**
10	0.0301	1.544	Supported
20	0.0303	2.963	Supported
30	0.0327	4.826	Supported
40	0.0338	6.697	Supported
50	0.0349	12.165	Supported
60	0.0354	13.093	Supported
70	0.0362	14.664	Supported
80	0.0498	16.411	Supported

It is observed that the number of supported flows without aggregation, is 70 flows, because these flows have less than 150ms jitter and packet loss is less than 4 %. Jitter is less than 150ms from 10 to 70 flows, without aggregation, and the corresponding R-factor value ranges from 70 to 90, with a MOS score of 3.6 to 4.3. This means that these calls are medium to high quality. At 80 flows, the jitter becomes high and unacceptable at 158.292ms, which gives R-factor values of 70 and less. This indicates a MOS score of 3.1 to 1.0, meaning that the quality is very poor. Thus this flow is not supported.

With aggregation results in Table 4.4, the total number of supported flows is 80 flows, with R-factor values from 80 to 90 and a MOS score of 3.6 to 4.3. This means high quality calls were produced for all 80 flows, but without aggregation only 70 flows were quality calls. Packet loss with and without aggregation is less than 4 %, which means packet loss is tolerable. In the aggregation tests, the number of supported flows increase by only 10, compared to the number of supported flows without aggregation. This means that packet aggregation on the end-to-end test bed only achieved a 12.5 % increase in the number of supported flows compared to those without aggregation. This is because on the end-to-end test bed, alternative routes from PC1 to PC2 exist, because the network is large, and therefore the congestion is not as much as it is in the hop-to-hop test bed. For instance, referring to figure 3.7 in Section 3.5.1, when VoIP traffic is transmitted from PC1 to PC2, PC1 can either choose to go via MP1 or via MP2, or via any of the mesh potatoes, depending on the best route selected by BATMAN.

But on the hop-to-hop test bed in Section 3.5.1 Figure 3.6, the only way from node A to node C is through node B. Also, the only way to node D is through node B. No alternative routes exist, therefore the network becomes more congested than the end-to-end test bed. When the network is congested, the network cannot support many calls when aggregation is not used. The hop-to-hop test bed achieved more supported flows than the end-to-end test bed. In congested scenarios, the aggregation module does not wait long, because as packets enter, they are immediately aggregated. Unfortunately, when there are fewer packets on the network, the packets that do not reach the minimum allowed limit, are released when the delay timer expires. Moreover, on the hop-to-hop test bed, aggregation and deaggregation modules are loaded on nodes A,B,C and D, which means all four nodes had to perform aggregation. On the end-to-end test bed though, PC1 and PC2 are the only nodes performing aggregation and deaggregation of packets. This means that hop-to-hop aggregation, aggregates more packets than end-to-end.

TABLE 4.5: Hop-to-hop vs. end-to-end

	Hop-to-hop 4 node PCs	**End-to-end 7 mesh potatoes+2 PCs**
Without aggregation (baseline)	20 calls	70 calls
With aggregation (kernel-level)	60 calls	80 calls

Table 4.5 summarises the results from the hop-to-hop and end-to-end test beds, by comparing hop-to-hop to end-to-end aggregation. 80 flows of traffic was injected, and the number of supported VoIP calls were determined by considering packet loss, jitter and delay. On the hop-to-hop test bed, the number of calls supported without aggregation were 20 calls, and with aggregation were 60 calls. This means that the supported calls increase by 40 compared to the number of supported calls without aggregation. The end-to-end test bed without aggregation can support 70 calls. With aggregation, it can support 80 calls, which means the supported calls were increased by only 10 when compared to the number of supported calls without aggregation.

4.3 Summary

This chapter presented an analysis of the results of the experiments. The results show that packet aggregation can reduce traffic in WMNs, by combining small multiple packets together. Packet aggregation shows a significant increase in the number of supported calls, compared to when packet aggregation is not used. Jitter with aggregation is less than jitter without aggregation, for both end-to-end and hop-to-hop test beds. This means that higher throughput was achieved with aggregation, than without aggregation. This increased the number of supported calls. Packet aggregation is effective whether the network is large or small, but a higher aggregation ratio is achieved when the network is congested. Hop-to-hop aggregation enables more supported calls than end-to-end aggregation. Referring to the research question, a conclusion can be drawn that kernel level packet aggregation, with mesh potato devices running BATMAN protocol, can increase the number of supported calls.

Chapter 5

Conclusion

This chapter contextualises the results in terms of justifiable ramifications, expresses the limitations of the research design and results, makes recommendations for others conducting similar work, and then makes suggestions for future work. Section 5.1 draws a conclusion based on the results presented in Chapter 4. Section 5.2 presents the limitations regarding the manner in which the experiments were conducted. Section 5.3 presents recommendations to the future researchers working in this field. Finally, Section 5.4 suggests future work, with interesting research topics combining packet aggregation and WMNs.

5.1 General conclusion

The main aim of this project was to improve VoIP performance in WMNs. Packet aggregation is one mechanism that is widely used to increase VoIP performance by increasing the number of supported VoIP calls in WMNs. In this research project, packet aggregation was implemented directly to the Linux kernel, using a queuing discipline, and the deaggregation module was implemented as a hook in the netfilter subsystem, found in the kernel. Quantitative methods were used to collect and analyze data with Iperf network performance tool, using a realistic VoIP traffic profile. Packet aggregation was tested with two different WMN test beds: a four-node mesh network with hop-to-hop aggregation, and a nine-node mesh potato network with end-to-end aggregation.

Hop-to-hop aggregation was found to be more effective, and produced better results than end-to-end, because the number of supported calls increased by 40, over unaggregated traffic. On the end-to-end test bed, the number of calls supported only increased by 10 , over unaggregated traffic. Moreover, on the hop-to-hop test bed with aggregation, jitter and packet loss are all within acceptable tolerances, and therefore produced high quality calls. A higher aggregation ratio is achieved because the throughput produced is higher than without aggregation. On the end-to-end test bed with aggregation, packet loss is acceptable for the first 60 injected flows, and without aggregation , packet loss is also acceptable up to 60 flows. The difference is that packet loss from the aggregation tests is higher than unaggregated tests, only for the first 60 flows, but from 70 to 80 flows, packet loss without aggregation increases and becomes higher than with aggregation. During the analysis of the results obtained, packet aggregation performed very well in congested traffic, because the hop-to-hop test bed was more congested than the end-to-end test bed. Yet, it achieved a higher aggregation ratio than the end-to-end test bed.

Hop-to-hop aggregation on the hop-to-hop test bed, outperformed the end-to-end implementation 4:1. The end-to-end only increased the number of supported calls with 12.5 %, whereas hop-to-hop increased supported calls by 50 %. The smaller scale hop-to-hop, was more congested than the end-to-end, and therefore the jitter was always minimal, because as packets were received by the aggregator, they were immediately aggregated, but in end-to-end, the aggregator may have to wait for more packets if the stack of packets is not large enough to be aggregated. However, based on the results, end-to-end aggregation appears to be more suitable for larger scale networks, because fewer CPU cycles are consumed as opposed to hop-to-hop, whereby packet aggregation is performed at each node. Thus end-to-end aggregation has an added advantage when deployed in larger networks than in smaller scale networks.

5.2 Limitations of the research design

The main limitation of this research is the number of nodes that were used to test packet aggregation. Four and nine nodes are very few, because WMNs are deployed in both urban and rural areas where the network must provide connection for all mesh clients. Only 80 flows of VoIP traffic were injected on the end-to-end and hop-to-hop test beds, although the end-to-end test bed is larger than the hop-to-hop. Testing the end-to-end test bed with more VoIP traffic flows, say 140 flows, would have created a more congested network, which would have produced more realistic results. Hop-to-hop aggregation was only tested on the four-node test bed, and end-to-end aggregation was only tested on the nine-node test bed. If both hop-to-hop and end-to-end aggregation had been implemented on the four-node test bed, and on the nine-node test bed respectively, then hop-to-hop and end-to-end aggregation results from both test beds could be compared. This would have demonstrated better which method efficiently aggregates more packets.

For both the end-to-end mesh potato test bed, and the hop-to-hop test bed, the experiments were conducted inside a building. Mesh potato networks are meant to be deployed in rural places, which means that the quality of the links may not be the same as in a building. Therefore, packet aggregation may aggregate packets too large for the link to carry, if the link is poor, which will result in heavy packet losses. Running the experiments out of the building would have produced the exact maximum sizes in bytes of the aggregated packets that the network can carry. This would have been better preparation to deploy packet aggregation modules in rural areas on a larger scale network with mesh potatoes.

5.3 Recommendations for similar work

Kernel aggregation is extremely difficult, and there are few documents online about this level of debugging in the kernel, except the online HOWTOs. This means that if one desires to take packet aggregation to another level, one must at least have basic knowledge of how the network stack handles packets in the kernel. Linux kernel code is not documented in detail, and documentation that is available, is mainly available for the users, not for kernel developers. Therefore, *printk* is a very useful function recommended in order to debug the kernel, since little information about kernel development is available. *Printk* is different from popular *printf*, because *printk* prints only statements and errors from modules loaded in the kernel, while *printf* is used for any program that is being executed in user space.

5.4 Suggestions for future work

Following on the work described by this thesis, one could also explore packet aggregation using more nodes, first in a laboratory setting, then in a real urban and rural mesh network. This will allow to evaluate the packet aggregation performance in longer links and longer transmission ranges. To shore up some of the limitations, one could implement packet aggregation inside the mesh potato devices. This will allow the testing of hop-to-hop aggregation on a mesh potato network, since mesh potatoes run a different kernel, OpenWRT. The 802.11n wireless standards implemented frame aggregation in the MAC layer, where as in this research, packet aggregation was implemented in the IP layer of 802.11 a/b/g standards. Testing the IP layer aggregation in the kernel, against the MAC layer aggregation, would demonstrate a more efficient method in terms of the accumulative throughput produced, and the overall performance of each method. This will identify improvements in the kernel level aggregation algorithm.

BATMAN has been used in this research, and BATMAN does not use routing metrics to determine a quality link but uses OGMs, a good follow-up research would be to do packet aggregation based on the link quality indicated by the BATMAN's OGMs. This would mean that the maximum and the minimum aggregation sizes will not be fixed but pre-calculated for each route. Knowing the quality of the link, will help to determine the maximum size of the aggregated packet that each link can carry. Developing this kind of mechanism can help to improve the aggregation ratio, and therefore improve the quality of VoIP calls. Header compression is also another mechanism proven to reduce VoIP overheads. Implementing header compression in combination with kernel level packet aggregation done in this research, could increase VoIP performance greatly.

Bibliography

[1] Akyildiz, I., Wang, X., & Wang, W. (2005). Wireless mesh networks: a survey. *Computer networks*, *47*(4), 445–487.

[2] Almesberger, W. (1999). Linux network traffic control implementation overview. *2001*, 296–301.

[3] Brolin, J. & Hedegren, M. (2008). Packet aggregation in Linux. Master's thesis, Karlstad University, Sweden.

[4] Cai, L., Xiao, Y., Shen, X., & Mark, J. (2006). VoIP over WLAN: Voice capacity, admission control,QoS and MAC. *International Journal of communication systems*, *19*(4), 491–508.

[5] Castro, M., Dely, P., Karlsson, J., & Kassler, A. (2007). Capacity increase for Voice over IP traffic through packet aggregation in wireless multihop mesh networks. *2*, 350–355.

[6] Cole, R. & Rosenbluth, J. (2001). Voice over IP performance monitoring. *ACM SIGCOMM Computer Communication Review*, *31*(2), 9–24.

[7] Collins, D. (2002). *Carrier Grade Voice Over IP*. McGraw-Hill.

[8] Creswell, J. (2009). *Research design: Qualitative, quantitative, and mixed methods approaches*. Sage Publications, Inc.

[9] Dely, P. (2007). Adaptive aggregation of voice over IP in wireless mesh networks. Master's thesis, Karlstad University, Sweden.

[10] Dludla, A., Ntlatlapa, N., Nyandeni, T., & Adigun, M. (2009). Towards designing energy-efficient routing protocol for wireless mesh networks. *Southern Africa Telecommunication Networks and Applications Conference (SATNAC 2009)*, 1–2.

[11] Ganguly, S., Navda, V., Kim, K., Kashyap, A., Niculescu, D., Izmailov, R., Hong, S., & Das, S. (2006). Performance optimizations for deploying VoIP services in mesh networks. *Selected Areas in Communications(JSAC)*, *24*(11), 2147–2158.

[12] Garg, S. & Kappes, M. (2003). Can i add a voip call? In *Proceedings of IEEE International Communications Conference, ICC'03*, volume 2, (pp. 779–783).

[13] Goodwin, C. (2009). *Research in psychology: Methods and design.* Wiley.

[14] Groom, F. & Groom, K. (2005). *The basics of voice over Internet Protocol.* Chicago: International Engineering Consortium.

[15] Hsu, C. & Kremer, U. (1998). IPERF: A framework for automatic construction of performance prediction models. In *Workshop on Profile and Feedback-Directed Compilation (PFDC), Paris, France.*

[16] Jain, A., Gruteser, M., Neufeld, M., & Grunwald, D. (2003). Benefits of packet aggregation in ad-hoc wireless network. *Department of Compututer Science, University of Colorado, Boulder, CO, Tech. Rep. CU-CS-960-03.*

[17] Johnson, D., Ntlatlapa, N., & Aichele, C. (2008). A simple pragmatic approach to mesh routing using BATMAN. *2nd IFIP International Symposium on Wireless Communications and Information Technology in Developing Countries, CSIR, Pretoria, South Africa.*

[18] Kassler, A., Castro, M., & Dely, P. (2007). VoIP packet aggregation based on link quality metric for multihop wireless mesh networks. In *Proceedings of the Future Telecommunication Conference (FTC).*

[19] Kazemitabar, H., Ahmed, S., & Nisar, K. (2010). A comprehensive review on VoIP over wireless LAN networks. *International Journal of Computer Science Letters, ISSR journal, 2.*

[20] Kim, K. & Hong, S. VoMESH: Voice over wireless mesh networks. In *Wireless Communications and Networking Conference, WCNC 2006.*, volume 1, (pp. 193–198).

[21] Lee, K., Yun, S., Kang, I., & Kim, H. (2008). Hop-by-hop frame aggregation for VoIP on multi-hop wireless networks. In *Proceedings of IEEE International Conference on Communications, ICC'08*, (pp. 2454–2459).

[22] Lin, Y. & Wong, V. (2006). Wsn01-1: frame aggregation and optimal frame size adaptation for IEEE 802.11 n WLANs. In *Global Telecommunications Conferenc, GLOBECOM'06*, (pp. 1–6).

[23] Maxemchuk, N. & Lo, S. (1997). Measurement and interpretation of voice traffic on the internet. In *IEEE International Conference on Communications, ICC 97*, volume 1, (pp. 500–507).

[24] Murray, D., Dixon, M., & Koziniec, T. (2010). An experimental comparison of routing protocols. *ANTAC: Australasian Telecommunication Networks and Applications Conference*, 159–164.

[25] Narasimhan, K. (2000). An implementation of differentiated services in a Linux environment. Master's thesis, Computer Engineering, North Carolina State University.

[26] Raghavendra, R., Jardosh, A., Belding, E., & Zheng, H. (2006). IPAC: IP-based adaptive packet concatenation for multihop wireless networks. In *Fortieth Asilomar Conference on Signals, Systems and Computers, ACSSC'06*, (pp. 2147–2153).

[27] Reynolds, R. & Rix, A. (2001). Quality VoIP - an engineering challenge. *BT Technology Journal*, *19*, 23–32.

[28] Skordoulis, D., Ni, Q., Chen, H., Stephens, A., Liu, C., & Jamalipour, A. (2008). IEEE 802.11 n MAC frame aggregation mechanisms for next-generation high-throughput WLANs. *Wireless Communications, IEEE*, *15*(1), 40–47.

[29] Varshney, U., Snow, A., McGiven, M., & Howard, C. (2002). Voice over IP. *Communications of the ACM*, *45*(1), 89–96.

[30] Waharte, S., Boutaba, R., & Ishibashi, B. (2006). Routing protocols in wireless mesh networks: challenges and design considerations. *Multimedia Tools Appl.*, *29*, 285–303.

[31] Wang, W., Liew, S., & Li, V. (2005). Solutions to performance problems in VoIp over a 802.11 wireless LAN. *IEEE Transactions on Vehicular Technology*, *54*(1), 366–384.

[32] Welte, H. (2000). The netfilter framework in Linux 2.4. In *Proceedings of Linux Kongress.*

[33] Zhang, L., Zheng, L., & Ngee, K. S. (2002). Effect of delay and delay jitter on voice/video over IP. *Computer Communications, IEEE Std. 1076*, *25*(9), 863–873.

[34] Zheng, L., Zhang, L., & Xu, D. (2001). Charecteristics of network delay and delay jitter and its effect of voice over IP (VoIP). *In Proceedings of IEEE International Conference on Communications, ICC'01*, *1*, 123–126.

Appendix - Work-in-progress for SATNAC 2010

Call Capacity for Voice over Internet Protocol on Wireless Mesh Networks

Docas D. Zulu and William D. Tucker
Department of Computer Science
University of the Western Cape, Private bag X17, Bellville 7535, South Africa
Tel: +27 21 959 3010, Fax: +27 959 3006
Email: {**2612638**, btucker}@uwc.ac.za

Abstract-This paper describes work in progress on call capacity optimization for voice over Internet Protocol on wireless mesh networks. In a developing country such as South Africa, evidence has shown that rural inhabitants find it difficult to afford the voice services offered by cellular networks. Voice over Internet Protocol is known for its affordability relative to cellular voice services, therefore deploying such services for rural communities will not only benefit rural inhabitants but also offer economic advantages to service providers. We are interested in the provision of voice services with rural wireless mesh networks. Unfortunately voice on mesh networks can experience packet loss and delays that cause reduction in voice quality. Transmission of small voice packets over wireless mesh networks imposes high overhead that leads to a tremendous decrease in call capacity. Therefore, we aim to study the performance of voice over 802.11 wireless mesh networks and evaluate packet aggregation mechanisms that merge small voice packets into a single large packet, in order to preserve voice quality with more calls. We will implement and evaluate packet aggregations mechanisms on a 'mesh potato' network with iterative cycles of laboratory experiments using a network simulator to collect data for performance evaluation.

Index Terms— WiFi 802.11, Quality of Service (QoS), Voice over Internet Protocol (VoIP), wireless mesh networks, packet aggregation.

I. INTRODUCTION

This paper describes work in progress concerning call capacity optimization for voice over Internet Protocol (VoIP) on wireless mesh networks (WMNs) by using optimization techniques such as packet aggregation. VoIP services are increasing in popularity due to ubiquitous Internet availability. For instance, Skype recorded more than 10 billion minutes of call time in its first year of deployment. This tremendous volume is due to cost-effectiveness achieved by VoIP and that its deployment is easy [1]. VoIP over wireless networks can also be used at homes and offices, in both developed and developing countries such as South Africa. Of particular interest to us are wireless mesh VoIP projects like Village Telco (www.villagetelco.org). A village telco is a community based telephone network that is based on a suite of open source applications that enable entrepreneurs to set up and operate a telephone service in a given area, urban or rural. Mesh networks are also inexpensive and easy to deploy.

A village telco can be designed for a rural community with a collection of 802.11bg mesh routers, known as 'mesh potatoes', that use an FXS port to connect an analog phone to a VoIP network, e.g. with Asterisk. Thus, end-users in rural communities can make 'free' VoIP calls using mesh potatoes connected via a village telco, and can make prepaid PSTN breakout calls provided a gateway is in place. However, this cheap and convenient VoIP over wireless mesh has its downfalls. For instance, maintaining QoS for VoIP traffic in a mesh network can be difficult. Packet loss can be deleterious due to interference when using unlicensed bands, and also high overheads of the TCP/IP stack. Research has shown that on a wireless mesh network with 2Mbps link speed, the number of calls reduces from 8 calls in a single hop to one call after 5 hops [2]. This major call capacity reduction is caused by the transmission of so many small voice packets over 802.11wireless mesh networks. Our challenge is to learn how to deal with such a problem.

The rest of the paper is organized as follows. The next section describes related work. Section III proposes methods to learn how to increase call capacity. Finally, Section VI concludes the paper and identifies future work.

II. RELATED WORK

Research has shown that one of the major reasons why the number of calls decreases as the number of hops increases is high overhead in the lower layers of the OSI stack, and that MAC layer headers are the dominant factor that causes high overhead [3][4]. Other research has shown that there are several mechanisms to reduce high overhead, e.g. header compression using a scheme called Robust Header Compression (ROHC) [5]. ROHC can reduce a 40 byte RTP/UDP/IP header to a 2 byte connection ID that can be used for only one hop. IP-based adaptive packet concatenation (IPAC) is a packet aggregation scheme that aggregates packets based on the quality of the link [6] (see Figure 1). This work showed that a good quality link can carry larger packets while a poor quality route may drop the packets if it carries packets that are too large.

Packet aggregation is classified as end-to-end or hop-by-hop [2]. End-to-end packet aggregation is done at every source. That is, packets sent toward a common destination are aggregated together. In hop-by-hop aggregation, packets are aggregated and disaggregated at every hop by adding a forced computation delay at every hop.

Research has exposed limitations of the distributed coordination function (DCF) of IEEE 802.11ab in supporting VoIP calls over a wireless LAN in [7]. 802.11 DCF is a MAC technique that assists in preventing collisions by employing CSMA/CD. The study focused on the upper bound on the number of simultaneous VoIP calls that can be supported in a single hop running DCF. Calculations using mathematical methods were done for three standard codecs namely ITU'S G711 a-Law, G723.1 and G729. In this study with a G711 codec, a 20ms payload entailed a maximum of 12 connections and a 28ms payload had a maximum of 40 connections. Therefore increasing the size of the payload was found to be a solution to increase call capacity. Conclusions were drawn that the larger the payload per frame in a wireless mesh network, the more the number of supported voice calls could increase. This study showed that smaller voice payload packets can decrease the number of supported medium quality calls and increasing the payload per frame is a desirable solution.

III. Methods

We wish to explore such techniques, as described in the previous section, on mesh potatoes for a typical village telco deployment environment. Iterative cycles of laboratory experiments will be conducted on a simulated mesh network using simulation tools such as ns-2/ns-3. We also intend to conduct similar experiments on an actual mesh network with mesh potato devices.

A mesh potato runs OpenWrt and there are QoS scripts that are used or installed inside OpenWrt to maintain QoS. We would like to develop a mechanism that will increase call capacity while the QoS scripts still maintain QoS. Modification will be done on the QoS scripts inside OpenWrt such that the packet aggregation technique improves call capacity while voice packets are not lost. Factors such as packet loss, latency and jitter will be measured to ensure that QoS is not compromised when this packet aggregation technique is implemented.

Packet aggregation techniques implemented on wireless mesh networks have been shown to increase call capacity tremendously [2]. Therefore we propose examining packet aggregation algorithms (see Figure 1) on mesh potatoes.

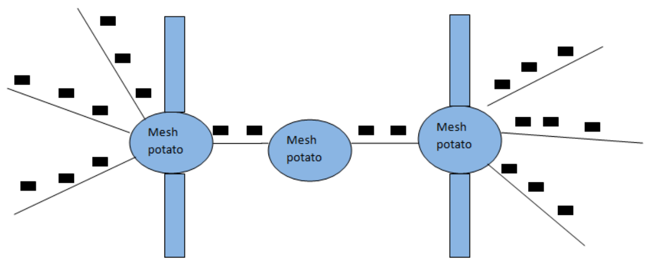

Figure 1 illustrates small voice packets from different calls being aggregated to form one large packet and then being disaggregated.

Research has shown that high protocol overhead is mainly caused at MAC layer 2 and also at layer 1. Thus aggregation at the IP layer of the TCP/IP stack can help relieve overhead [4]. The use of packet aggregation mechanisms will result in a decrease of protocol overhead thus increasing the number of supported calls. Our goal is to learn which packet aggregation mechanisms will work best for a mesh potato network.

IV. Conclusion and Future Work

VoIP has been described in related work as an affordable protocol when deployed over mesh networks with attendant QoS challenges. We want to improve VoIP capacity on wireless mesh networks composed of mesh potatoes. This paper has provided a description of the drawbacks of VoIP traffic over wireless mesh networks. Research has shown that MAC layer overhead is the dominant factor that reduces call capacity. We will experiment with hop-by-hop packet aggregation techniques on mesh potatoes to increase the number of VoIP calls supported.

Research has shown that a good quality route can carry a large aggregated packet while a poor quality route can suffer higher packet loss if large packets are transmitted over it [6]. Therefore for future work we would like to determine the ideal aggregated packet size in order to maintain VoIP quality. Header compression has been shown to be also one of the effective techniques to increase the number of calls supported. Therefore we would like to compare header compression techniques with packet aggregation on a mesh potato network to discover the call capacity management techniques that are most effective on those devices.

References

[1] A. Kassler, M. Castro, and P. Dely, "VoIP packet aggregation based on link quality metric for multihop wireless mesh networks," *Proceedings of the Future Telecommunication Conference,* Beijing, China, 2007.

[2] V. Navda, S. Ganguly, K. Kim, A. Kashyap, D. Niculescu, R. Izmailov, S. Hong, and S. Das, "Performance Optimizations for Deploying VoIP Services in Mesh Networks," *IEEE Journal on Selected Areas in Communication (JSAC)*, 2006, pp. 2147-2158.

[3] W. Wang, S. Liew, and V. Li, "Solutions to Performance Problems in VoIP Over a 802.11 Wireless LAN," *IEEE Transactions on Vehicular Technology*, vol. 54, 2005, pp. 366-384.

[4] M.C. Castro, P. Dely, J. Karlsson, and A. Kassler, "Capacity Increase for Voice over IP Traffic through Packet Aggregation in Wireless Multihop Mesh Networks," *Future Generation Communication and Networking (FGCN*), Korea, vol. 2, 2007, pp.350-355.

[5] S. Jung, S. Hong, and P. Park, "Effect of robust header compression (rohc) and packet aggregation on multi-hop wireless mesh networks," *The Sixth IEEE International Conference on,Computer and Information Technology, CIT'06.* 2006, pp. 91–91.

[6] R. Raghavendra, A.P. Jardosh, E.M. Belding, and H. Zheng, "IPAC: IP-based adaptive packet concatenation for multihop wireless networks," *in Proc. of IEEE Asilomar Conference on Systems, Signals and Computing,* Pacific Grove, CA, 2006, pp. 2147-2153.

[7] S. Garg and M. Kappes, "Can I add a VoIP call?," *in Proc. IEEE Int. Conf. Commun.,* Anchorage, AK, 2003, pp. 779–783.

Docas D. Zulu an MSc student of Computer Science at the University of the Western Cape (UWC). Her research interests include VoIP, cryptography and wireless mesh networking.

William D. Tucker is a Senior Lecturer of Computer Science at UWC. His interests include Internet Protocol network and their applications in developing regions.

Appendix - Unpublished 5 page paper

Packet aggregation for voice over Internet Protocol on wireless mesh networks

Docas D. Zulu and William D. Tucker
Department of Computer Science
University of the Western Cape, Private bag X17, Bellville 7535, South Africa
Tel: +27 21 959 3010, Fax: +27 959 3006
Email: {**2612638**,btucker}@uwc.ac.za.za

Abstract-This paper validates that packet aggregation is a viable technique to increase call capacity for voice over Internet Protocol over wireless mesh networks because the technique can deliver the same quality of service with fewer packets that need to be routed in an ad hoc fashion. Wireless networks are an attractive way to provide voice services to rural communities, as evidenced by ubiquitous cellular coverage in South Africa, and therefore afford economic advantage to service providers. However, since most rural inhabitants cannot afford cellular voice services, alternative and cheaper wireless networks can be very attractive to both inhabitants and service providers. One such alternative is a wireless mesh network (WMN), or of more interest to service providers, a large collection of WMNs. Due to the ad hoc hop-to-hop routing nature of mesh networks, packet loss and delay can reduce voice quality. Even on non-mesh networks, voice quality is reduced by the high overhead associated with a multitude of relatively small voice packets. Therefore, we sought to show that conventional packet aggregation techniques should also succeed on wireless mesh networks that present interesting challenges due to their ad hoc nature. One is that hop-to-hop kernel modifications are difficult and time consuming to debug. We implemented and tested kernel level packet aggregation of voice packets on four mesh nodes running Linux and conducted standard baseline vs. aggregation tests with a realistic voice traffic profile in hop-to-hop mode. We then transferred the kernel level modifications to either end of a nine node 'mesh potato' network and conducted those tests with only the end nodes modified to perform aggregation duties. We verified the expected increases in call capacity with packet aggregation while maintaining quality of service in both instances, and noticed that hop-to-hop aggregation outperformed the end-to-end configuration 3:1. However, we feel that implementing hop-to-hop in a scalable fashion is prohibitive due to the extensive kernel level debugging that must be done to achieve the call capacity increase. We therefore suggest that end-to-end call capacity increase is an acceptable compromise for eventual scalable deployment of voice over wireless mesh networks.

Categories and Subject Descriptors
C.2.1 [Computer-Communication Networks]: Wireless Communication, **B.8.2 [Performance and Reliability]:** Performance and Design Aids, **C.4 [Performance of Systems]:** Performance attributes.

I. Introduction

This paper describes a study concerning call capacity optimization for voice over Internet Protocol (VoIP) on wireless mesh networks (WMNs) by using optimization techniques such as packet aggregation. VoIP services are increasing in popularity due to ubiquitous Internet availability [1][2]. WMNs has been proven to provide users with the freedom of roaming, it's known benefits include ease of deployment and expansion, better and wider coverage, cost effective in maintenance and quick recovery from node failure [1]. VoIP over WMNs can also be used at homes and offices, in both developed and developing countries such as South Africa. Of particular interest to us are wireless mesh VoIP projects like Village Telco (www.villagetelco.org). A village telco is a community based telephone network that is based on a suite of open source applications that enable entrepreneurs to set up and operate a telephone service in a given area, urban or rural. A village telco can be designed for a rural community with a collection of 802.11bg mesh routers, known as 'mesh potatoes' that use an FXS port to connect an analog phone to a VoIP network. Research shows that the main challenges of VoIP over WMNs are system capacity and system performance i.e. high quality VoIP service. Maintaining high quality VoIP traffic in a (WMNs) can be difficult. Packet loss, delay and jitter are major causes of inefficient delivery of high-quality VoIP services. These three are caused by using unlicensed bands and also high overheads of the TCP/IP stack [1]. For instance, in popular voice codec G729a a voice payload of 20 bytes is used by requires an additional 40 bytes RTP/UDP/IP header per packet [6]. On a (WMNs) with 2Mbps link speed, the number of calls reduces from 8 calls in a single hop to one call after 5 hops [1][3]. This major call capacity decrease is caused by the transmission of many small voice packets over 802.11wireless mesh networks [1][3]. The main aim of this research is to study the performance of VoIP over WMNs and we propose a packet aggregation technique to increase the number of supported VoIP calls in WMNs.

The rest of the paper is organized as follows. Section II reviews related work. Section III describes the methods used in implementing packet aggregation. Section IV describes the experimental results obtained from simulation that outlines the significant of packet aggregation in WMNs. Finally, Section V concludes the paper and identifies future work.

II. WORK RELATED TO PACKET AGGREGATION

Studies have proven that high overhead in the lower layers of the network stack causes poor VoIP performance and that packet aggregation is one of the solutions to overcome such negativity [1][4][5].
Aggregation in the IP layer is called packet aggregation, in the MAC layer it is called frame aggregation or frame concatenation [6]. Aggregation can be done either end-to-end or hop-to-hop. The end-to-end approach aggregates packets at the source and only packets going towards a common destination and in hop-to-hop, aggregation and deaggregation is performed at every node [1]. End-to-end aggregation introduces delay only at the source thus reduces the overall delay while hop-to-hop aggregation introduces delay at every node which leads to higher delay but can achieve higher aggregation ratio than end-to-end [6].

Packet aggregation can be implemented in many ways depending on the requirements and the size of a network.

Lin et. al [6][6] has done frame aggregation and optimal size adaptation for IEEE 802.11 WLANs. In this research a frame size is calculated based on the probability of the Bit Error Rate (BER), where the frame size is estimated to be small on a link with a high BER and a frame size that is big where the BER is low. However this model is proven to achieve high throughput than fixed frame size but only applicable for WLANs and single hops [7][6]. Therefore may not apply to WMNs due to issues such as self interference.

IP based Adaptive Concatenation scheme (IPAC) shown in [7] is an end-to-end aggregation scheme where packets are aggregated based on the quality of the link. This work concluded that a good quality link can carry larger packets while a poor quality link may drop the packets if they are too large. IPAC is proven to perform well in high traffic loads even though end-to-end delay is high [8].
Robust Header Compression (ROHC) is header compression scheme that can reduce a 40 byte RTP/UDP/IP header to a 2 byte connection ID that can be used for only one hop [9][8].

III. PACKET AGGREGATION SCHEME

A. Aggregation

Packet aggregation is defined as a means of combining small multiple packets together to form a larger packet. Packet aggregation techniques implemented on wireless mesh networks, wireless networks or wired networks have been proven to increase call capacity tremendously [10][9][10][11].

Packet aggregation has been implemented as a queuing discipline (qdisc(s)) in this project. Every network device has queues that which is used to accept packets for transmission. In Linux kernel language these queues are called qdiscs of which they form a major component of the Linux traffic control code. We have implemented our aggregation scheme as a qdisc because a qdisc can simply be attached to a network interface in this case we attach it to wlan0 a wireless interface. We have ingress (receiving) and egress (outgoing) qdiscs [11][11]. Qdiscs are designed to have an enqueue function that accepts packets and a dequeue function that dequeues packets out of the device as soon as they are ready for transmission [12][11]. The qdiscs found in the Linux kernel all have a standard way that they are written in; they have functions that control packet handling. The kernel is designed in such a way that it only accepts qdiscs that are written in the same manner as other qdiscs found in the kernel. The qdiscs functions are enqueue(), dequeue(), requeue(), drop(), init(), change(), reset(), destroy() and dump(). The dequeue() function is the important function in our implementation because that is where aggregation was performed. When packets hit the device they are immediately enqueued by function enqueue() and then function dequeue() will then pull them out for transmission and just before the dequeue() function transmits them, they will first be aggregated and then sent out.

During aggregation, a new large socket that will hold the packets that are combined is created. A socket buffer (skb) is a data type that its primary assignment is to hold network packets and it is used in the Linux kernel. A *skb* consists of a MAC header, IPv4 header and a payload. Aggregation was done through combining multiple *skbs* by first taking the payload together with the IPv4 headers and inserting into a new empty large *skb*. When packets are aggregated a new IPv4 header and a MAC header is created. The new IPv4 header and the MAC header is used to identify the aggregated packet, since we are implementing at layer 3 of the network stack an IPv4 header is required to be able to transmit our aggregated packets.

The old MAC header is destroyed but the IPv4 address of each packet is kept since it contains the IP addresses of each packet. The new IPv4 header holds the identification number that will be used as a value to check for in the deaggregation module. This value is randomly selected currently set to 253. Note that we combine *skbs* into one aggregated *skb*. The maximum number of *skbs* to be aggregated is set currently to 1500 bytes. Figure 3 below shows aggregation of two *skbs* but any number of *skbs* can be aggregated in this fashion ruled by the maximum limit value set.

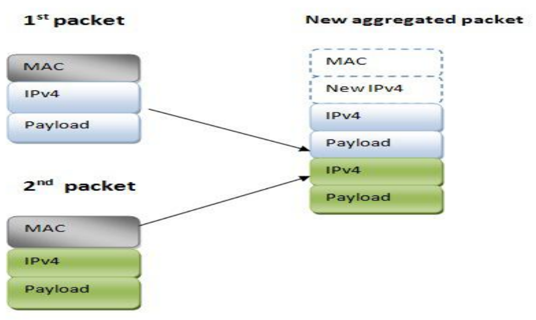

Figure 1 *illustrates two packets being aggregated*

B. Deaggregation

Deaggregation is done in the similar manner as aggregation, a new empty *skb* is created and then the first *skb* (payload, corresponding IPv4 header and the MAC header) in the aggregated *skb* is copied into new empty *skb*. The IPv4 header that was created for the aggregated group of packets is discarded as it was only used for transmission (see figure 4 below).

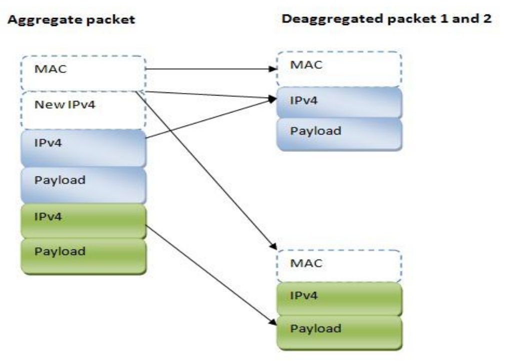

Figure 2 *illustrates the concept of deaggregation, the payload of packet 1 together with its original IPv4 header and the MAC header will be copied and restored.*

Deagregation module is implemented using hooks that are found in the netfilter subsystem of the Linux kernel. Netfilter is a subsystem that is defined as a framework for packet handling (http://netfilter.samba.org). Hooks are locations or points in the network stack where packets traverse. We have decided to implement deaggregation as a hook because hooks have levels of priority therefore enabling us to register our deaggregation method as the first priority hook to be attended. When the aggregated *skb* is accepted in the receiving node the aggregated *skb* need to be immediately deaggregated, therefore hooks enable us to hijack aggregated traffic as soon as it hit the receiving node. The first netfilter hook is known as the NF_PRE_ROUTING its assignment is to declare packets as stolen if they are taken over by another module or accepted if no module hijacks them. There are a series of return codes that this hook returns depending on what happens to the packets. In our implementation the return code returned by this hook is NF_STOLEN which means that the packet has been taken over by another module which is our deaggregation module. So in our implementation we use this hook to sort of steal packets so that we can immediately deaggregate them and insert them back to the stack. Therefore we took advantage of the first netfilter hook and attached our deaggregation function there (see figure 4 below).

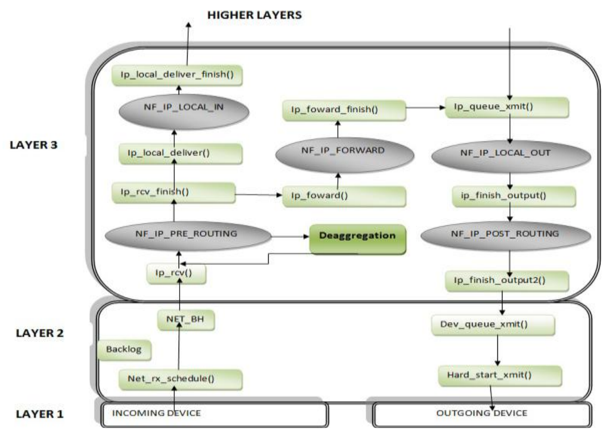

Figure 3 *illustrates the journey of a packet through the network stack. A packet is received by an incoming device at layer 1 it passes through to layer 2, routed to layer 3 and then the deaggregation module takes over aggregated packets.*

C. Kernel Configuration

The aggregation module was compiled as a standalone loadable module, to load this module we used a linux application called Traffic Control (tc). TC is a tool used to attach a qdisc to network device interfaces. We created a new a simple add-on module that is a helper when calling the aggregation module. This means that the name that we used to call our aggregation module called "aggregate" should be the same as in this add-on. TC belongs to the iproute2 which is "a collection of utilities TCP/IP networking in traffic control in Linux" (http://www.linux-foundation.org/en/Net:Iproute2). Our add-on aggregate is placed inside TC and compiled together for it to be recognized, in short TC just help us to attach our aggregate module to wlan0 interface. The following is what we did to activate our aggregate module *tc qdisc add dev wlan0 root aggregate max 1500 min 200*. The name of our qdisc is aggregate, we want it to be the root i.e. to handle all incoming *skbs* and we add it on device *wlan0*, 1500 is the maximum number of packets to be aggregated in bytes and the minimum number is 200 bytes.

D. Experimental Design

In this section we present the process that we have followed to conduct our experiments.

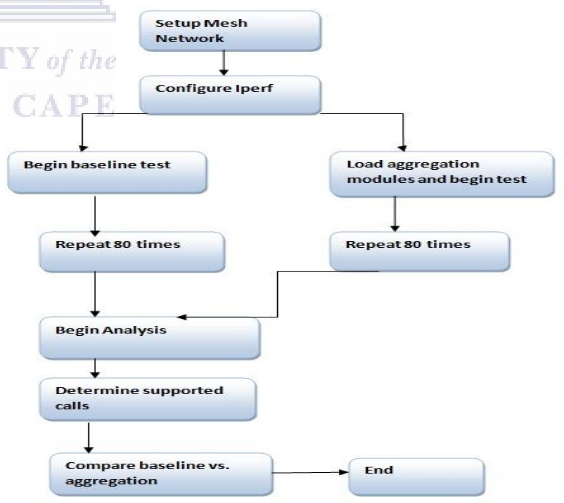

Figure 4 *represents the experimental process*

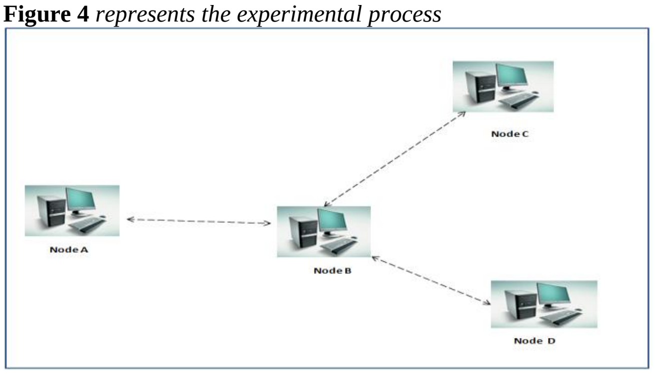

Figure 5 *presents the 4 nodes testbed*

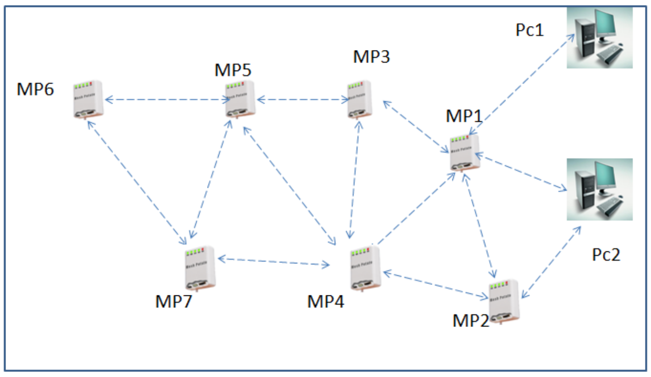

Figure 6 *illustrates the 9 nodes mesh potato network*

We have used 4 nodes and 9 nodes setup for conducting our tests represented. The 4 nodes were all running Ubuntu 10.4 with Linux kernel version 2.6.32.35. We have used BATMAN-ADV 2010.0.1 protocol for the 4 node setup and for the 9 nodes the mesh potatoes were all running BATMAN-ADV 2011.0.1 we therefore installed BATMAN ADV 2011 on Pc1 and Pc2 on the 9 node setup so that there could be communication between the mesh potatoes and the Pcs(See **Figure6**). BATMAN is a is a proactive routing protocol that does not determine the whole path to destination but only the best next hop to the right direction. This prevents processes such as route discovery and varying qualities of the link. Of which in this project we need quality links for effective aggregation. We have used IPERF a traffic generating tool that can generate User Datagram Protocol (UDP) traffic and evaluate the performance of a network.

We have conducted a baseline test which is when our aggregation module is not activated and when aggregation is activated. In order to test the performance of our system we test it according to the VoIP profile that which is proven to be approximately the same as a normal VoIP conversation. Studies have shown that the average VoIP conversation is 180 seconds [12]. This means that a person at station A will talk for 180 seconds and then station B will reply back for another 180 seconds. We therefore have used 180 seconds for the length of our first test this means that we have set Iperf to generate UDP packets for 180 seconds from sender to receiver and then another 180 seconds for the receiver to reply back. Iperf can be set as a server (server (sender) or client (receiver) this means that (referring to the 4 node setup) when node A is communicating with node B node A will be a server and node B will be a client.

The default VoIP payload for a G.729 VoIP codec is estimated to be 20 bytes and when the RTP/IP headers are included makes the total of VoIP payload to be 60 bytes [1]. We therefore set Iperf to generate UDP packets of 60 bytes each for 180 seconds. It is proven that as the number of hops increase the number of supported calls decreases we therefore test our solution on a two hops network. This means that for the 4 node (**Figure 5**) we use node B as the node that only forwards traffic to node C and D. We have used iptables to filter out traffic, this means that for traffic A-C we cut the connection between A and C using iptables such that the only way to C is through B, in this way we have tested our solution for more than 1 hop. We apply the same rule for sending traffic from node A-D and vice versa. The test was repeated 80 times this means that from A-C,C-A, A-D and D-A is 4 times and we generate the traffic 80 times of which we refer 4 times as 4 flows of traffic and 80 times as 80 flows. This means that by increasing the number of flows we are increasing the number of VoIP calls of which will help us to know how many calls the network can support with and without aggregation. We apply the same for the 9 node by cutting the connection between Pc 1 and Pc2 such that the only way to Pc2 is through any of the mesh potato.

IV. RESULTS

Packet loss, delay and jitter have been proven to be major cause of inefficient delivery of quality VoIP performance. We therefore take note of the performance of these characteristics of VoIP traffic generated by Iperf. We present our results as follows:

WITHOUT AGGREGATION				WITH AGGREGATION			
Injected VoIP flows	Packet loss (%)	Jitter(ms)	Supported calls	Injected VoIP flows	Packet loss (%)	Jitter(ms)	Supported calls
10	3.2	0.1786	Supported	**10**	0.02211	0.112	Supported
20	3.2013	0.1818	Supported	**20**	0.8135	0.115	Supported
30	4.72	0.2264	Not supported	**30**	0.8807	0.115	Supported
40	7.01	0.247455	Not supported	**40**	1.8564	0.115	Supported
50	7.57	0.2545	Not supported	**50**	3.3406	0.117	Supported
60	7.65449	0.2677	Not supported	**60**	3.5798	0.12	Supported
70	13.082	0.7212	Not supported	**70**	5.2604	0.12	Not supported
80	15.03	1.23046	Not supported	**80**	6.900027	0.12	Not supported

Table 1 *represents the number of calls supported with and without aggregation on the 4 nodes mesh network, supported calls increases up to 60 flows with aggregation.*

WITHOUT AGGREGATION				WITH AGGREGATION			
Injected VoIP flows	Packet loss (%)	Jitter(ms)	Supported calls	Injected VoIP flows	Packet loss (%)	Jitter(ms)	Supported calls
10	0.00646	18.5773	Supported	**10**	0.0301	1.544	Supported
20	0.00725	37.8193	Supported	**20**	0.0303	2.963	Supported
30	0.00954	56.765	Supported	**30**	0.0327	4.826	Supported
40	0.01069	75.6743	Supported	**40**	0.0338	6.697	Supported
50	0.01568	96.13	Supported	**50**	0.0349	12.165	Supported
60	0.02266	123.208	Supported	**60**	0.0354	13.093	Supported
70	0.12455	140.522	Supported	**70**	0.0362	14.664	Supported
80	0.183	158.292	Not supported	**80**	0.0498	16.411	Supported

Table 2 *represents the number of calls supported with and without aggregation on the 9 nodes mesh potato network, supported calls increases up to 80 flows with aggregation.*

HOP-TO-HOP AGGREGATION			END-TO-END AGGREGATION	
	Without aggregation	With aggregation	Without aggregation	With aggregation
No of injected flows	80 flows	80 flows	80 flows	80 flows
No of supported flows	20 flows	60 flows	70 flows	80 flows

Table 3 *compares hop-to-hop aggregation vs. end-to-end aggregation. Hop-to-hop achieved 40 supported calls while end-to-end supported 10 calls*

.

Studies have proven that for VoIP conversation to be acceptable or for the user to be satisfied of the perceived voice quality the average packet loss for a conversation should be less than 4 % for the G.729 VoIP codec. As well as jitter including end-to-delay needs to be less than 150 ms for acceptable VoIP quality, this means that flows are considered as supported flows if the average packet loss is less than 4 % and the jitter is less than 150ms. We therefore used this measure to determine the number of supported calls based on the packet loss and jitter results received from Iperf.

The results show a significant difference when aggregation is used and when it is not used, when we look at 4 nodes results (See **Table 1**) we observe that without aggregation the number of supported flows is 20 flows this means that after 20 flows the quality of the calls degrades which is not acceptable to the user. But when we look at aggregated traffic we observe that 60 flows are supported which means that aggregated traffic can add 40 more flows as compared to unaggregated traffic. This means that in the 4 nodes setup packet aggregation can achieve 75% of the number of supported calls where as unaggregated traffic can only achieve 25%.

When we look at the 9 nodes mesh potato results (See **Table 2**) we observe that 70 flows can be supported with unaggregated traffic, whereas with aggregation 80 flows can be supported which only shows an increase of 10 flows between the two. This is because on the 9 nodes alternative routes from Pc1 to Pc2 exist since the network is large and therefore no heavy congestion exists. But on the 4 nodes, from node A the only way to node C is through node B no alternative routes therefore the network becomes congested as keep injecting VoIP traffic. When the network is congested the network cannot support many calls when aggregation is not in use that is why the 4 nodes network achieves many calls than the 9 nodes. We also observe that on the 4 nodes (**Figure 5**) packet aggregation was implemented hop-to-hop i.e. on Node A,B,C and D, and on the 9 nodes packet aggregation was implemented end-to-end i.e. on Pc1 and Pc2. Studies from the related proved that hop-to-hop yield a better aggregation ration than end-to-end [3].

V.CONCLUSION AND FUTURE WORK

We implemented packet aggregation on wireless mesh network and we have found that packet aggregation can reduce overheads and increase the quality of VoIP performance. We tested hop-to-hop vs end-to-end aggregation, baseline tests against aggregation tests and packet aggregation showed a significant increase in the number of supported calls as compared to the baseline. We tested our solution on a 4 nodes and 9 nodes mesh potato network and we have found out that hop-by-hop packet aggregation can aggregate more packets than end-to-end and that packet aggregation is more efficient on congested traffic. We recommend printk for kernel debugging because it’s one of the easiest ways to know what’s going on in the background. We could not port the kernel aggregation modules in the mesh potato kernel due to time constrain, however it is possible. This limited us to only test our implementation in the mesh potato network end-to-end. In future we desire to implement hop-by-hop packet aggregation directly in the mesh potatoes.

REFERENCES

[1] V. Navda, S. Ganguly, K. Kim, A. Kashyap, D. Niculescu, R. Izmailov, S. Hong, and S. Das, «Performance Optimizations for Deploying VoIP Services in Mesh Networks», *IEEE Journal on Selected Areas in Communication (JSAC)*, 2006, p. 2147-2158.

[2] A. Kassler, M. Castro, and P. Dely, VoIP Packet Aggregation based on Link Quality Metric for Multihop Wireless Mesh Networks, *Proceedings of the Future Telecommunication Conference, Beijing, China*, 2007.

[3] M.C. Castro, P. Dely, J. Karlsson, and A. Kassler, Capacity Increase for Voice over IP traffic through Packet Aggregation in Wireless Multihop Mesh Networks, Future Generation Communication and Networking(FCGN), 2007, p. 350–355.

[4] W. Wang, S. Liew, and V. Li, Solutions to Performance Problems in VoIP Over a 802.11 Wireless LAN, *IEEE Transactions on Vehicular Technology*, vol. 54, 2005, p. 366-384.

[5] D.P. Hole and F.A. Tobagi, Capacity of an IEEE 802.11 b Wireless LAN supporting VoIP,” *Proc. of IEEE ICC*, 2004, p. 196–201.

[6] V.W. Y Lin, Frame Aggregation and Optimal Frame Size Adaptation for IEEE 802.11n WLANs., *in Proceedings of IEEE GLobal Telecommunications Conference*, San Francisco, 2006,p. 1-6.

[7] R. Raghavendra, A.P. Jardosh, E.M. Belding, and H. Zheng, “IPAC: IP-based Adaptive Packet Concatenation for Multihop Wireless Networks, in Proc. of Asilomar Conference on Systems, Signals and Computing, Pacific Groove, CA, 2006, p. 2147-2153.

[8] S. Jung, S. Hong, and P. Park, “Effect of Robust Header Compression (ROHC) and Packet Aggregation on Multi-hop Wireless Mesh Networks,” *Computer and Information Technology, CIT'06.* 2006, p. 91–91.

[9] A. J. Kassler , and P. Dely, On Packet Aggregation for VoIP in Wireless Meshed Networks, *in Proc. of 7th Scand. Workshop on Wireless Ad-hoc Networks,* Stockholm, Sweden, 2007.

[10] K. Kyungtae and H. Sangjin, VoMESH: Voice over Wireless Mesh Networks, *in Proc. of IEEE Wireless Communications and Networking Conference(WCNC)*, vol. 1, p. 193-198, 2006.

[11] K.P. Narasimhan , An Implementation of Differentiated Services in a Linux Environment,” Master's thesis, North Carolina State University, 2000.

[12] C.N. Chuah and R.H. Katz, “Statistical analysis of packet voice traffic in Internet multimedia applications,” *unpublished, July*, 2000.