AALTO UNIVERSITY
School of Science and Technology
Faculty of Information and Natural Sciences
Department of Computer Science and Engineering

Design, Implementation and Evaluation of Traffic Shaping Proxies for Power Saving in Mobile Audio Streaming

Mohammad Ashraful Hoque

Master's Thesis
Espoo, Jan 31, 2010

Supervisor: Professor Antti Ylä-Jääski, Aalto University
Instructor: Matti Siekkinen, Ph.D, Aalto University
Instructor: Dr. Jukka Nurminen, Nokia Research Center

AALTO UNIVERSITY

School of Science and Technology
Faculty of Information and Natural Sciences
Degree Programme of Computer Science and Engineering

ABSTRACT OF
MASTER'S THESIS

Author: Mohammad Ashraful Hoque

Title of thesis:
Design, Implementation and Evaluation of Traffic Shaping Proxies for Power Saving in Mobile Audio Streaming.

Date:	Jan 31, 2010	**Pages:** 9 + 71
Professorship:	Data Communications Software	**Code:** T-110
Supervisor:	Professor Antti Ylä-Jääski	
Instructor:	Matti Siekkinen, Ph.D	
Instructor:	Dr. Jukka Nurminen	

The mobile applications, which depend on network activities, have significant effect on comparative battery lifetime. This thesis performs a study on power consumption for mobile Internet radio application with WLAN and 3G networks. Then we propose two proxy based solutions to increase power savings: application layer and transparent proxy, where transparent proxy is applicable only for WLAN network condition. Though the proxy approach was not effective in 3G networks, but our experiment results reveal that our implemented proxies can save power maximum 65% and 57% respectively in WLAN condition, depending on the audio streaming rate.

Keywords: PSM, 3G, WLAN, Burst, Proxy, Transparent, TCP, M-TCP, Freeze-TCP, Internet Radio, Power Consumption.
Language: English

Acknowledgements

I would like to express my gratitude to my instructor, Matti Siekkinen, Ph.D, who has helped me and looked after my work during the project by providing his valuable ideas or thoughts. He also has answered my every queries being patient. Only due to him I have learned some data analysis tools, which I have used in this project to present most results.

I also want to pay my gratitude to my instructor Dr. Jukka Nurminen, from Nokia Research Center, and also want to thank Heikki Waris and Jussi Ruutu for their important feedbacks during the project.

I would also like to thank my supervisor, Professor Antti Ylä-Jääski, for considering me in this project and encouraging me. Special thanks to Professor Sasu Tarkoma for encouraging me by his valuable comments.

In addition, I am very grateful to my collegue, Jani Heikkinen, who has been very helpful, co-operative and patient in numerous miscellaneous incidents.

I have been very fortunate by being in touch with such nice persons from both Aalto University and Nokia Research Center.

Espoo, January 31st 2010

Mohammad Ashraful Hoque

Abbreviations and Acronyms

3G	3rd Generation
3GPP	3rd Generation Partner Project
802.11	IEEE WLAN Standard
AAC	Advanced Audio Coding
AP	Access Point
AB-PCM	Adaptive-Buffer Power Save Mechanism
BER	Bit Error Rate
CAM	Continuous Active Mode
CBR	Constant Bit Rate
CNAME	Canonical Name
CPU	Central Processing Unit
CWND	TCP Congestion Window
DTIM	Delivery
FEC	Forward Error Correction
GET	HTTP request method
HTTP	Hyper Text Transfer Protocol
IEEE	Institute of Electrical and Electronics Engineers
I-TCP	Indirect Transmission Control Protocol
ICY	I Can Yell (SHOUTcast protocol)
ICY-Metadata	ICY Header
ICY-Metaint	ICY Header
MMS	Microsoft Media Service
MS-RTSP	RTSP windows Media extension
NAT	Network Address Translator
POST	HTTP request method
PSM	Power Saving Mode
PSNA	Power Saving Network Architecture
PS-TP	Power Saving Transport Protocol
QoS	Quality of Service
RRC	Radio Resource Control

RNC	Radio Network Controller
RT_PS	Real Time Power Saving Protocol
RTSP	Real Time Streaming Protocol
RTSPU	RTSP using UDP
RTSPT	RTSP using TCP
RTP	Real Time Protocol
RTCP	Real Time Control Protocol
RTT	Round Trip Time
RWND	TCP Receiving Window
SACK	Selective Acknowledgement
SDP	Session Description Protocol
STPM	Self Tuning Power Management
TDMA	Time Division Multiple Access
TCP	Transmission Control Protocol
TIM	Transfer Indication Map
UDP	User Datagram Protocol
VBR	Variable Bit Rate
WLAN	Wireless Local Area network
WNIC	Wireless Network Interface Card
WCDMA	Wideband Code Division Multiple Access
ZWA	Zero Window Advertisement

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The popularity of mobile phones has gained its peak. While serving the basic purpose of talking or voice communication, mobile phones are also having other a lot of functionalities and applications. These devices are augmented with cameras, high computation and storage capabilities, audio and video applications. Besides, to provide better data communication services, mobile phones are also equipped with built-in wireless LAN (WLAN), 3G, IP telephony, audio, video streaming capabilities, etc.

This chapter presents the motivation behind this thesis work. Then also outlines the scope and structure of this thesis introducing the basic ideas required to understand the work.

1.1 Motivation

The integration of wireless network interfaces in mobile phones helped to achieve seamless connectivity but the constraint of limited battery capacity is challenged by this integration. Wireless interface cards consume as much power as an idle mobile phone. In work [47], *Sumit Roy et al.* reports that WNIC consumes around one third of the total mobile phone's power consumption and under heavy traffic this consumption jumps to two third. Hence, the reduction of the power consumption of this unit would yield a potential saving in overall power consumption of the mobile phone. This is also true for 3G networks but the scenario is more challenging (Chapter 6).

These mobile devices work on limited power source battery. Therefore, the applications developed for these devices must make efficient use of the hardware resources to reduce power consumption. But sometimes the standalone

particular application is not enough to reduce the power consumption. It might require the help of other dependent hardware and protocols. For instance streaming applications are high power consuming software. Because the continuous delivery of media traffic to the mobile device requires to power up a radio frequency controller. Here, WLAN PSM (Power Saving Mode) is not sufficient enough to reduce the overall power consumption. In case of 3G, besides link layer characteristics, the higher value of the inactivity timers may lead to inefficient power saving. Furthermore, most of the underlying protocols are also not application and energy aware. Then the processing or decoding the media data and then present to the user also require power. Therefore, there is a need of efficient techniques to be deployed from both hardware and software point of view.

In our thesis work we focus on audio streaming application (Mobile Internet Radio) for energy efficient solution for both WLAN and 3G networks. Internet Radio application uses TCP based HTTP like ICY streaming protocol, unlike other HTTP streaming protocols used by media services, which uses only one single GET request to deliver the audio stream and then play. Here the Internet radio stations are free and broadcast live or simulated live media streams.

1.2 Thesis Scope

The purpose of this thesis work is to cover the aspects related to the power consumption in mobile audio streaming. Here, audio streaming is dependent on streaming application itself, then different networking technologies, protocols, techniques, Internet architecture, so on and therefore, they have effect on total power consumption in mobile devices.

The very general approach to achieve power efficiency would be to increase the battery capacity. But, the recent researches reveal that, there is very little hope of further battery capacity improvement. Therefore, a lot of works have been continuing to increase the power efficiency from different perspectives. For instance, new operating systems are designed with more focus on power or energy efficiency through reducing power while system is idle, adopting operating system for hard disk management [41]. Strategies have been developed including the MAC layer transmission [58]. From the application perspective, there are some middleware to perform complex and power consuming computations on behalf of the mobile clients [52], another approach is to compress the data before exchange, for instance lossless compression of e-mail [55].

This thesis work concentrates on application level power consumption and proposes two proxy-based solutions to save the power for Mobile Internet Radio Streaming and evaluates their performance.

Our work begins by outlining the power consumption behavior in presence and absence of 802.11 PSM while running different radio stations at the mobile client. Then we describe two proxy-based schemes and their working mechanism. The proposals are based on shaping the audio traffic at the proxy server, i.e. buffer for a fixed amount of time and then send the buffered traffic in a single burst to the client. We also studied the effect of underlying transport protocol (TCP) and Internet architecture on the application layer bursts and thus mobile power consumption.

Afterwards we present the power consumption status in presence of the proxy server and with several test case scenarios. In addition to the WLAN, we have also verified the feasibility and performance of the application layer proxy server in 3G networks.

1.3 Thesis Structure

This thesis is organized as follows. Chapter two begins by presenting the steps involved in audio/video streaming. Afterwards we present key application layer streaming protocols and then various streaming techniques followed by the streaming servers to cope with the network and client situation. Next, chapter three introduces IEEE 802.11 power management technique and a quite through review of past works related to WLAN power management. In chapter four we present our power measurement framework, where we explain detail about our target architecture and application. In addition, we also describe two different proxy-based solutions to reduce the power consumption for the target application. Chapter five demonstrates our experiment results for different perspectives in WLAN condition and further analysis. In chapter six, we present our proxy performance in 3G networks. Finally we wrap up our study by presenting the outline of our future work in the conclusion.

Chapter 2

Streaming: Protocols and Techniques

Media streaming is the process of playing audio or video while being downloaded from Internet. It enables users to play media without having the media file locally stored. In the process, web servers contain the references of the media stream files hosted on different media servers or networks. When a user clicks on the link, a media server or network streams the content directly to the user. Here, media contents are stored on a different host or network, therefore, streaming does not have any impact on the web servers.

In this chapter, we present our detail study about media streaming. We begin with the steps involved in audio/vedio streaming process. Then we describe important streaming protocols and after that different kind of streaming techniques followed by media streaming servers to provide to support better stream quality and maximize user connection capacity.

2.1 Media Streaming

A complete media streaming system works by creating, delivering and at the end playing the media at the client player. The key elements involved to accomplish the whole process are: *capturing*, *encoding*, *editing*, *serving* and *playing* (Figure 2.1).

Capturing: The very first approach to create streaming media is to capture audio or video from an analog source like camcorder or tape. Then digitize and store the media on the disk. This is done with special analog-capture card and software. The capture card may also support the live delivery of

the digitized media while storing on the disk. However, with newer digital camcorder, the media does not require to be digitized and can be stored directly on the disk.

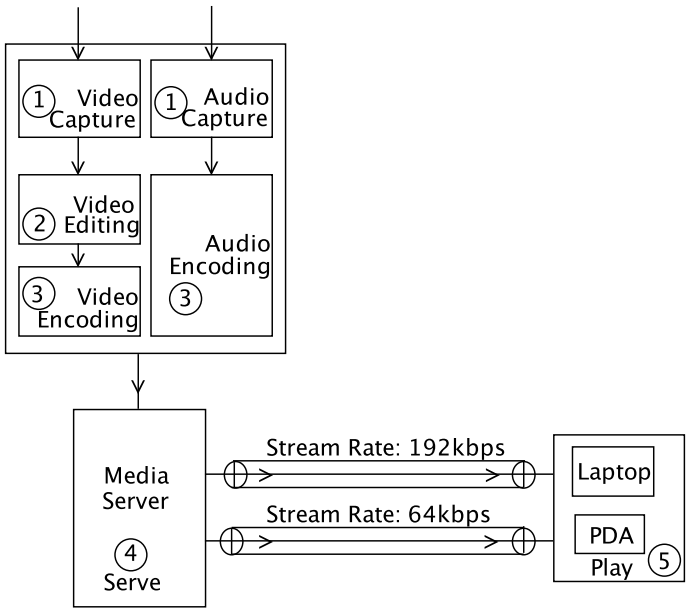


Figure 2.1: Streaming Procedure.

Editing: When a media stream is stored, it can be edited with various tools to include media metadata or to integrate other audio/video to create multimedia presentation for entertainment or training.

Encoding: Once the audio or video is edited it can be encoded to the appropriate streaming file format with the specified resolution, frame rate and data rate. It may require creating separate files for the same media content with different data rate, to support clients with different Internet connections or to deal with bandwidth fluctuation.

Serving: As the media is encoded, it is ready to be delivered to the clients. The media server delivers the media content using appropriate transport protocols. Here, the server is optimally configured or licensed to deliver the media to a fixed number of stream connections.

Playing: After requesting to the server, client player starts receiving the media stream, buffers and plays. The players can pause, play, stop, fast forward depending on the kind of stream or streaming methods (Section 2.2).

2.2 Streaming Methods

Media streaming can be classified based on the media creation time and the mode of request into *on demand* and *live streaming*.

In case of *on demand* streaming, a media content is available for users all the time and delivered or streamed upon request. Here, the media contents are pre-recorded, preassembled and users can have control over playing the stream like stop, pause, fast-forward or even can jump to future frames.

Live streaming makes it possible to deliver the media data to the users within the seconds of the incident. It also enables multiple clients to access the same media contents at the same time just like TV channel broadcasting. But users cannot fast forward or rewind the stream because the incident is happening in real time and the server is also delivering the media files as a unique data flow. Similar kind of media stream, called *simulated live*, is also delivered like *live streaming*. The only difference with *live streaming* is that *simulated live* events are pre-recorded.

Live or *Simulated live* streaming requires a small buffer at the client to store a small portion of the stream before playing. Clients discard the buffered stream after it has been processed or presented to the user.

2.3 Streaming Protocols

The delivery of media stream is eased with the use of specific application layer protocols, where the protocols are responsible to control the media session, to encapsulate media information, and to control the delivery of the media data. The prevailing application layer streaming protocols are Real Time Streaming Protocol (RTSP), Microsoft Media Service (MMS), and Hyper-Text Transfer Protocol (HTTP). Other streaming protocols like Real Time Protocol (RTP), RTP Control Protocol (RTCP) and Session Description Protocol (SDP) are used in conjunction with other key streaming protocols like RTSP.

2.3.1 Real Time Streaming Protocol

RTSP [48] is an application layer protocol to establish and control multiple media delivery sessions. It provides an extended framework to choose media delivery channels such as UDP, multicast UDP and TCP, and RTP [36] based delivery mechanism. Here, the set of media streams or sessions to be

Method	Description
DESCRIBE	Retrieves the description of a media object mentioned in the rtsp URI.
SETUP	Specifies the transport mechanism to be used to deliver the media.
PLAY	Tells the server to send the media traffic as mentioned in the SETUP.
PAUSE	Informs the server to halts the media transmission temporarily.
TEARDOWN	Stops the media transmission for the given rtsp URI.
REDIRECT	Tells the client to connect to the another server.

Table 2.1: RTSP Methods.

controlled is presented using a media description protocol like SDP [39].

An RTSP session starts by requesting a presentation or media to be started at the server. Server labels each session with an identifier. This session identifier represents the shared state between the server and client and is used in all subsequent controls. If the state is lost, RTSP stops the transmission of media through not receiving the RTCP [19] messages while using RTP.

An RTSP session is not bound to any specific underlying transport layer protocol. During an RTSP session a client may use UDP to send and receive control requests or may establish and tear down a number of TCP connections. Since the order of these control commands matters, it is most likely that RTSP uses TCP to transmit them. The Table 2.1 summarizes some required and recommended RTSP commands.

During a media session using RTSP, media packets are delivered via RTP over UDP and control packets are via RTCP. Figure 2.2 illustrates the number of connections associated with a RTSP streaming session and how RTSP works with other protocols.

However, RTSP and HTTP both are text based protocol and similar in syntax. It was done intentionally and therefore, any extension to the HTTP can also be added with RTSP. But, RTSP presents some basic differences with HTTP:

1. RTSP is independent of underlying transport layer protocols.
2. RTSP server has to maintain states by default. Figure 2.3 illustrates RTSP sate-machine diagram.

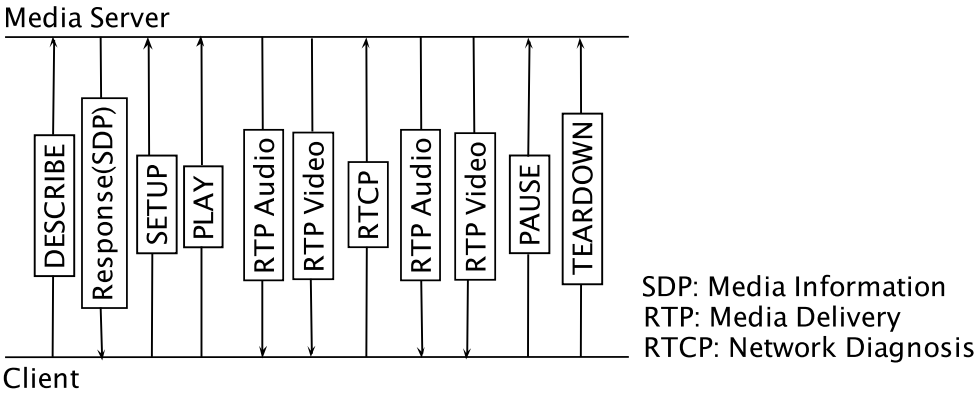


Figure 2.2: RTSP Session.

3. Both client and server can issue RTSP requests.
4. Media Data can be carried by an out-of-bound protocol like RTP.
5. RTSP request URI always contains the absolute URI.

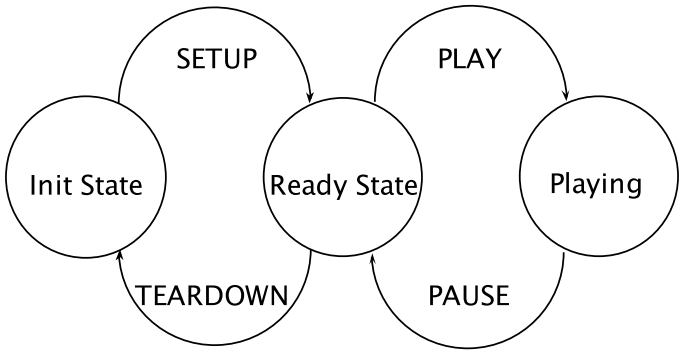


Figure 2.3: RTSP States.

2.3.2 HTTP Streaming Protocols

Many existing streaming services require specialized servers and protocols to provide media contents to the end users. For instance, RTSP requires the assistance of two other protocols like RTP, RTCP and SDP. In addition, Windows Media, Real Media require their own extension of RTSP protocol.

Besides, if the underlying protocol is UDP, the streaming can be blocked by the router firewall or NAT.

However, HTTP based streaming works on the top of TCP and requires a simple web server. Streaming works with some HTTP requests and resonses. For this reason, HTTP streaming owes the largest penetration and it constitutes a large fraction of Internet bandwidth used for streaming. It uses pseudo streaming, where a file can be played while it is being downloaded. The stream is accessed via a direct HTTP link or embedded into a HTML web page.

HTTP streaming includes different media formats like, Windows Media, Flash Video, MP3, etc. It may also include various streaming services like QuickTime [3], Windows Media [7], Real Media [14], SHOUTcast [15] media streaming.

QuickTime HTTP Streaming Protocol

QuickTime HTTP Streaming [2] uses RTSP and exploits the use of HTTP GET and POST methods. Client sends one GET request to the server to open a server-to-client connection and a POST request to open a client-to-server connection. The GET connection is used to send the media data to the client and POST connection is used to receive the RTSP requests from the client.

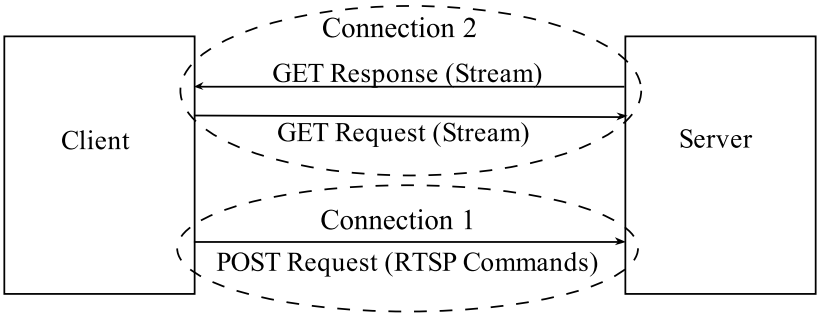


Figure 2.4: QuickTime HTTP Streaming.

Client continues to send the RTSP commands as the message body of the POST request but the server does not reply to these requests (Figure 2.4). RTSP methods sent over POST connection are encoded using *base64* method. This prevents intermediate HTTP proxy servers from deciding that an embedded RTSP request is a malformed HTTP request. The client may stop

POST connection at any time to free the resources at the server. But, the best time is to release the POST connection after sending the RTSP PLAY request.

Microsoft Media HTTP Streaming

Microsoft Media HTTP streaming protocol [5] can be thought as the HTTP tunnelled version of the MMS protocol. The MMS control requests are sent via HTTP GET request and for each requrest one separate TCP connection is established. This protocol works in two operating modes:

Simple mode requires at least two TCP connections initially. In addition, one separate TCP connection each time client pauses, resumes, seeks, etc. If the traffic load blancer is used at the server side, *simple mode* can suffer, beacuse control request connections can be established with different server and may incur streaming errors.

HTTP Pipelining,like QuickTime HTTP Streaming (Section 2.3.2) uses single HTTP connection to send all the MMS stream requests over a single TCP connection. Even though, pipelining greatly reduces the problem of the load balancer but may suffer if the intermediate HTTP proxy servers do not support pipelining.

Apple HTTP Streaming Protocol

Apple [27] uses a modified version of HTTP to deliver stored or live audio/video stream from iPhone, iPod, etc. They name it HTTP Live Streaming and trying to standardize it as an IETF protocol. According to the proposed draft [46], HTTP Live Streaming consists of three components: server component, distribution component and the client software.

Server Components include the media *encoder* and the *segmenter*. The *encoder* receives real-time signal from the audio-video device, encodes and encapsulates for delivery. Currently supported video encoding format is MPEG-2 and AAC or MP3 for audio. However the encoder does not do any transcoding.

The *segmenter* reads the encoded Stream and splits into a series of media files of equal duration. In addition, *segmenter* generates a index file containing the references of the segmented media files and updates the index file. *segmenter* updates this index file whenever it creates a new file. The purpose of the index file is to track the media file segments. The extension of the media

segments is *.ts* and index file format is an extension of *.m3u*.

Distribution Components are generally the web servers which deliver the index and media file segments over HTTP to the clients.

A ***Client Component*** is the client software and works by fetching the index file. Then starts downloading the media files in sequence specified in the index file. The client presents the stream to user only when a sufficient amount of stream is buffered.

SHOUTcast Streaming Protocol

Most popular Internet Radio Streaming engine SHOUTcast [15] uses their own extension of HTTP, called ICY protocol [18] to stream MP3 or AAC to the listeners. The client player sends simple HTTP request to the SHOUTcast server including a ICY Header *ICY-Metadata*. In reply server sends a ICY response header which contains metadata, like address, stream rate, etc, of a particular radio site and immediately followed by endless MP3 or AAC stream. ICY responses are same as HTTP, except HTTP is replaced with ICY and having different headers.

Since a SHOUTcast Server has to continue streaming songs to the clients as long as they are connected, it is required to provide a source of stream to draw on indefinitely. Like HTTP Streaming (Section 2.3.2) in web based applications, each connected client is provided a play list with *.pls* or *.m3u* extension. The play list file contains the URI of the radio streaming server.

In order to display the title of a song the client must include a special *ICY-Metadata* header in the original request. In response server embeds chunks of self-delimiting metadata and sends back a header *ICY-Metaint* which specifies the number of bytes of MP3 data that will be sent between each chunk of metadata.

2.3.3 Microsoft Media Service

MMS is Microsoft's proprietary protocol to transport unicast media streams from Windows Media Service to Windows Media Player or other Windows Media Service. The client can transport MMS Protocol control requests to the server to perform actions such as starting and stoppping a media session. The media data can follow the same TCP connection or can be transmitted as flow of UDP packets.

While sending media data to the client, client can make request to the server

to change the data rate of the stream being transmitted. In addition, if the data flow is UDP, client can request server to resend a UDP packet, in case of a packet loss.

MMS protocol functions in similar way of the RTSP Windows Media Extensions (MS-RTSP). It does not support server-side playlist seeking and advanecd fast start. However, it is proper to this MMS protocol when a streaming server does not support MS-RTSP or Windows Media HTTP streaming.

2.3.4 Other Streaming Protocols

Real-Time Transport Protocol (RTP)

RTP provides end-to-end delivery for real time audio/video data for streaming or other services. Applications run RTP on the top of UDP to make proper utilization of its multiplexing and checksum capabilities. RTP itself does not have own mechanism to ensure timely or in-order packet delivery or QoS. The sequence number contained in the RTP packet only allows the receiver to reconstruct the senders sequence number. Here, the sequence number can also be used to determine the location of the packets.

An RTP session consists of number of application participants, where each of the applicants can send or receive media data. A network address and two port numbers identify the participants. One port number is for media data and the other one is for RTCP. The number of media sessions depends on the media types, which enables the participants to choose the media types they really want to reciece. For instance. one participant may just want to receive only the audio part of a media streaming video.

RTCP checks the quality of the connection and the stream and sends feedback to both sender and receiver. This helps both sender and receiver to negotiate the best transmission rate. RTCP also provides identification and control mechanism for RTP transmission. Figure 2.5 illustrates an RTP session.

Even though, RTP was primarily designed to work on multicast to support multiple participants of a multimedia conference or interested in the same stream media, but, it also supports unicasting and various application needs.

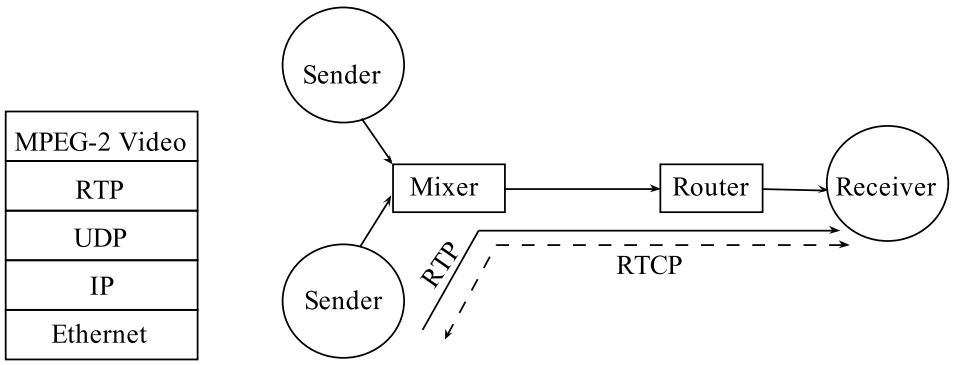


Figure 2.5: RTP protocol stack and an RTP Session.

RTP Control Protocol (RTCP)

Real Time Control protocol is used with RTP. It is based on periodic transmission of control packets to the all participants of an RTP session. Therefore, the main purpose of this protocol is to maintain the QoS by providing the feedback information such as packet loss, jitter, etc to all the participant but RTCP itself does not carry any media data packetss. As the control packts are important and can not be lost, RTCP works on TCP.

The feedback regarding flow-control and congestion control help to negotiate the best media transfer rate. However, feedback from the receiver also helps to diagnosis the problem in a distribution. For Instance, reports from all other participants help the one who is facing problem to identify whether the problems are local or global. In addition RTCP helps third-party to monitor networks status and determine the problem occurred when an IP multicasting mechanism is being deployed.

RTCP packet format specifies following five kinds of feedback messages:

1. Sender Reports (SR), includes cumulative frame and byte counts.
2. Receiver Reports (RR), includes frame loss or delivery report.
3. Source Description (SDE), contains the user and hostname of a participant.
4. *Bye* Message finalizes an established connection.
5. Other application specific messagese which do not include the information enumerated above.

During a RTCP session both the sender and receivers continuously monitor the RTP transmission quality by measuring the transmission delay, jitter and packet loss. Sender calculates the transmission delay from the timestamp specified in the RR. Sender also calculates the total number of packets and bytes sent and specifies in the SR. The receivers measure the packet loss by calculating the gap in the sequence number space:

1. It finds the lowest the sequence number.
2. Calculates the expected number of packets = current - lowest.
3. Then counts the received number of packets which may include duplicates, out-of-order and late arrived packets.
4. Number of lost packets = expected - received. This value can be negative also.

In order to collect the jitter information, receiver relates the local real-time clock with the sender's packet time stamp at the very beginning when the first packet arrives. After that receveer derives the next packet reception and calculates the deviation upon packet arrival. Jitter is derived for each pair of successive received packet.

After collecting the transmission quality information both the sender and receiver exchange the reports. Here the sender report include the timing information to measure the RTT, to perform the playout adjustment by the receiver. SR also contains the number of packets and bytes transmitted to help the receiver to do proper packet loss calculation. On the other hand, receivers report contains the feedback timing for the RTT estimation, total number of packets lost, fraction of packets lost (in the units of 1/256), the highest packet sequence number and the jitter of the received packets. Based on these reports all the participants adapt the current QoS of the transmission.

RTCP reports are to be sent by all the participants. In case of multicasting the RTCP reports my cause congestion in the network. Therefore, the protocol must include some bandwidth management to reduce congestion. This is obtained by controlling the reports dynamically but scalable RTCP message or report interval depends on the group size, RTCP data rate and the average RTCP packet size. However, bandwidth usage by RTCP should not exceed 5%.

Session Description Protocol

While Streaming audio or video, it is necessary to carry media details, transport addresses and other session description metadata to other participants. SDP is here to provide such information irrespective of the transportation method. SDP provides the description of a multimedia session for the purpose of session announcement, session invitation or other form of multimedia session initiation. It does not incorporate any transportation protocol and eligible to use with other protocols like RTSP (for media streaming). In addition, it is not intended to use for the negotiation of session content or media encoding.

An SDP session description consists the following information:

1. Name and purpose of the session.
2. Time duration of an active session.
3. The type of media comprising the session.
4. Information like address, port, format, etc to receive media data.

2.4 Streaming Techniques

Despite the existence of some good streaming protocols, streaming long duration media objects on Internet is still challenging. If the server supports UDP and the client player is shielded by a NAT, the incoming UDP media traffic may not be able to pass through NAT. In addition, the media object delivery between client and server usually require high and stable bandwidth. But, the bandwidth fluctuation during a media session can degrade the quality of stream significantly. Besides, user connections can vary from dial-up to high speed Internet. Therefore, it is not desirable to have a fixed encoding or data rate for a particular media object. Furthermore, user can experience prolonged start-up delay due to the network bandwidth fluctuation, server load and client play-out buffer. Therefore, various techniques have been adopted by different streaming services (Windows, RealNetworks Media Services) to address above problems or challenges.

2.4.1 Protocol Rollover

Even though UDP is considered to be the default media streaming protocol, it is often blocked at the client side. Therefore, streaming media is often delivered over TCP or HTTP. At the beginning of the client request, UDP is tried. But if the UDP is not supported due to the client or server reasons TCP or HTTP can be used. This process is known as protocol rollover. However, protocol rollover plays an important role in media streaming due to the wide deployment of NAT routers/ firewalls in home or small business office networks.

While using RTSP, client specifies the transport protocol in the SETUP message. If UDP is supported, the port numbers for data receiving and sending feedback are also specified. In SETUP reply message the streaming server also specifies the transport protocol. If it selects UDP then also specifies two ports for data sending and receiving feedback, otherwise selects TCP. But if the server supports UDP and the client player is behind a NAT router/firewall, after a timeout period client terminates current RTSP connection and sends e new SETUP request with transport protocol TCP.

MMS with the latest Windows media player, if *fast cache* is activated (Section 2.4.2), server tries to deliver the media data with TCP based protocol RTSPT. If the player does not support TCP then server selects UDP based RTSPU. If the UDP based connection also not successful then the server tries to deliver the media over HTTP. If *fast cache* is not enabled, the server tries to send in a different order, first RTSPU, then RTSPT and finally HTTP [10].

2.4.2 Fast Streaming

In general a media object is streamed at its encoding rate and a small amount of buffer is used to smooth the streaming jitter. But the bandwidth between client and server may fluctuate and the play-out buffer may be exhausted. Window Media services use a group techniques called fast streaming in order to provide high quality of stream, which includes *Fast Start*, *Fast Cache*, *Fast Recover*, *Fast Reconnect* and *Advanced Fast Streaming* [8].

Fast Start helps client to buffer media data at a higher rate than the media encoding rate until the play-out buffer is filled. It also enables the client to reduce the startup buffering time. In addition, users can fast forward, rewind without any additional delay and rebuffering. However, the bandwidth initially used by *fast start* to send data can increase the congestion. Therefore,

in order to reduce the risk of congestion users can limit the amount of bandwidth to be used to stream to each player. *Fast start* only works with unicast TCP or UDP stream.

Fast Cache is used immediate after*fast start* to stream the entire media object or the media session is terminated by the user. It also streams media data at a higher rate than the encoding rate. While *fast cache* is running, client player also maintains a growing buffer for the early arrived data. The purpose of *fast caching* is to guard against network bandwidth fluctuations. But, *fast cache* works only run with TCP.

Fast Recovery is used by the windows media player to recover the lost or damaged media without again requesting to the Windows Media server. It is accomplished by using Forward Error Correction (FEC) on a Windows Media Services publishing point.

Fast Reconnect reduces the impact of a temporary stream disconnection period, by enabling the client to reconnect to the server automatically and resume the streaming. In case of *on demand* the stream is resumed from where it was interrupted. But for *live* or *simulated live* stream clients may experience gap.

Advanced Fast Streaming enables client to play a media stream before its play-out buffer is full and the content is also being streamed at a higher speed at the same time. The acceleration is stopped when the client buffer is full.

2.4.3 Rate Adaptation

In addition to the *fast caching* popular media services like RealNetworks and Windows Media supports three kind of rate adaptation techniques in order to adapt bandwidth fluctuation.

Stream Switch makes a media server to switch dynamically among several streams with different encoding rates of the same media object and allows to select a stream based on the available bandwidth between the client and the server. In order to enable *stream switch*, the media object must be encoded with multiple bit rates.

Using RTSP, when a media session is established, the client media player sends a DESCRIBE message to the server. In DESCRIBE response message server includes the description of each media stream encapsulated in the media object. Then client selects the desired media object and specifies in the SETUP command.

Now, during playback, if the available bandwidth drops down to the selected media encoding rate, the client play may generate a request to the server to switch to a lower stream rate. In Windows media service it is known as *Intelligent Streaming* [9] and *Sure Streaming* [6] in RealNetworks media service.

Stream Thinning can be thought similar as *Stream Switch*. If enough bandwidth is not available, the quality of video stream is degraded as only key frames are delivered to the client.

Video Cancellation is enabled only when the bandwidth is not even enough to send the key video frames and only audio stream is maintained. The client may send a RTSP TEARDOWN command to cancel the video and may again setup and request video stream later again only if the bandwidth increases.

However, *Stream Thinning* and *Video Cancellation* are applicable only for video media streaming.

2.4.4 Coordinated Streaming

Coordinated Streaming [38] works with the cooperation of *fast caching* and *rate adaptation*. A range is maintained for the client's play-out buffer. The lower bound of the buffer range helps to eliminate the latency for stream switching and the upper bound to prevent aggressive buffering. When a stream session starts, server starts sending data as fast as possible until the lower bound is reached. Immediate after that, client begins to play and buffer media data at the highest rate until the client's play-out buffer reaches the upper bound. At this point, client buffers at the media encoding rate and keeps the buffer always full.

When the transmission rate falls down due to bandwidth fluctuation, play-out buffer drains off and if the play-out buffer touches the lower bound, client requests the lower stream rate. But, current bandwidth must be able to maintain normal playback for the new lower streaming rate and transmit extra data to fill out the play-out buffer to its upper bound. If the bandwidth again increases before the play-out buffer reaches the lower bound, client can request higher stream rate until the buffer is full again.

Chapter 3

WNIC Power Consumption in Mobile Devices

This chapter describes IEEE 802.11 basic power saving mechanism and its performance with different kind of applications. After that we present brief reviews of previous works for power saving; that had been done for simple web access and media streaming over WLAN. Finally we explain, how our works differ from the other previous works.

3.1 IEEE 802.11 WNIC Opertaing Modes

An wireless network interface card has four states of operation:*active (transmit and receive)*, *waiting*, *sleep* and *off* state. Each of these states has different power requirements. *Active* state is the most power demanding state. In this state, the interface usually sends and receives data and majority of the power is consumed to power the radio subsystem. During *waiting* state card does not send or receive any data, but radio's receiver is turned on to watch the medium. When the radio is turned off, the card enters into *sleep* state. In *off* state power is completely removed from the card. Hence, the descending order of the states in case of power consumption: active, waiting and sleep state.

Therefore, the transition from one state to another state also requires additional power; lower the power state of the card, more power is required to switch from current state to higher state. In *idle* state the card periodically alternates between *waiting* and *sleep* states. The table 3.1 describes the operating states of the wireless network interface card.

State	Description
Active	Sending and receiving data.
Waiting	No data transmission or reception. But the radio is powered up.
Sleep	Card is turned off.
Off	The power to the entire card is removed
Idle	Periodic alternate switching between Idle and Sleep states. No data is exchanged.

Table 3.1: Power states of the wireless interface card.

3.2 IEEE 802.11 Power Management

The *power saving mode* (PSM) defined by the 802.11 standard works by the cooperation of the access point(AP) and the mobile clients. A mobile client sends it's intention to the access point to switch to the *power saving mode*. While being in *power saving mode*, a mobile client switches its WNIC to *sleep* state, and periodically wakes up to receive traffic indication map(TIM) frames (also known as *beacons*) from the access point. In response, the mobile client sends *PS-Poll* frames to the access point only if the previous TIM frame contains an indication that AP has buffered data for the client. The AP sends the buffered data directed to the client only if it receives *PS-Poll* frame from that mobile client.

For multicast and broadcast data packets access point transmits delivery traffic indication message (DTIM) frames; immediately followed by the actual data packets. In this case, no *PS-Poll* frame is required from the client. DTIM is configurable at the access point and can be any multiple of the *beacon* interval. Figure 3.1 illustrates IEEE 802.11 basic power saving technique.

3.3 Performance of 802.11 PSM

802.11 power management enables an AP to perform two tasks. First is to determine whether AP should deliver the frames to the wireless network or to buffer the frames because the mobile clients might be in sleeping mode. The second task is to periodically announce the mobile clients, which have buffered frames waiting for them. Mobile clients can choose to sleep for more extended period of time to avoid using of the wireless interface card. But

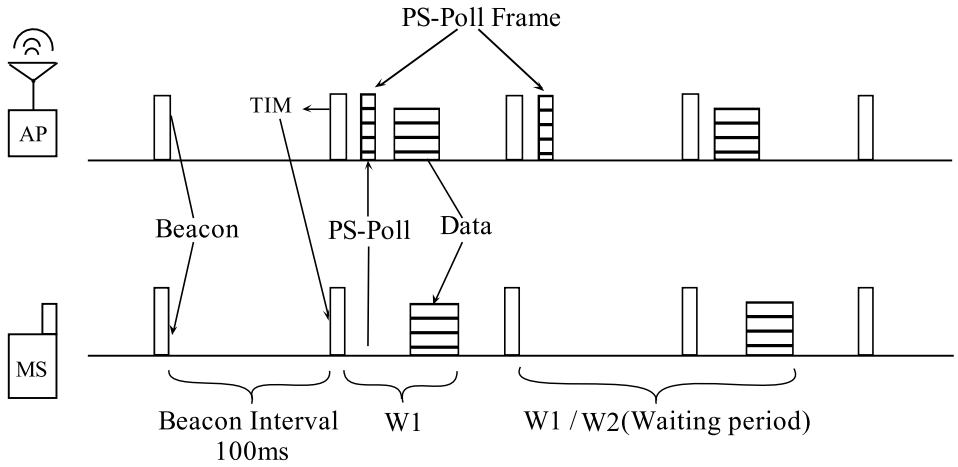


Figure 3.1: IEEE 802.11 Power Management.

this requires more buffering capacity at the access point.

It would seem that 802.11 *power saving mode* allows the mobile stations to save power by allowing them frequently to switch to the lower power consuming *sleep* mode or sleeping longer period. But, the potential power saving depends on the minimal *waiting* time interval, i.e. the time between the transmission of the *PS-Poll* packet and the reception of the data packets. However, 802.11 standard does not specify any bound for this *waiting* period.

In [32], *Chandra et al.* indentified several limiting factors of the 802.11 *power saving mode*. Table 3.2 few of them regarding streaming.

3.4 Related Works to Reduce WNIC Power Consumption

Along with other mobile Internet services, different kinds of multimedia streaming services are also getting popular. And a lot of works have been done to reduce the power consumption for these Internet dependent applications. In this section we classify the related previous works based on the application type; whether the user application is requires simple random web access or it is a multimedia streaming application.

Limitation	Explanation
Poor support for multiple clients	While multiple clients are accessing multimedia streams, even though all the clients send *PS-poll* requests at the same time, any of the clients might have to wait for the whole time needed to transmit the packets for all other clients.
PSM is not application aware	802.11 PSM behaves well if the distribution of traffic is periodic or regular as the client WNIC wakes up periodically. But the multimedia traffic tends to be very continuous and irregular. For instance, real media service sends smaller packets at very close intervals, therefore WNIC is always in *active sate* or spend very less time in *sleep state.*

Table 3.2: Limitations of 802.11 power saving mode.

3.4.1 Web Access

Wireless network is prone to the packet loss by its nature. This packet loss unnecessarily triggers TCP congestion control mechanism. Therefore, the throughput is degraded, as the total transmission delay is increased. *Stemm et al.*, in [44], found that this behavior increases the total energy cost. Besides, the dominant energy cost of any transport protocol is not the number of packets sent or received but the amount of time it takes to complete the transmission.

The very beginning work to reduce power consumption of WNIC was to improve the throughput at the transport layer. The work in [28], proposed Indirect TCP(I-TCP) architecture as an alternative to the classical TCP. I-TCP splits one TCP connection into two TCP connections. One connection is between the mobile client and access point, and the other connection is between the access point and the fixed host in the Internet. The access point relays the data between the fixed host and the mobile station. I-TCP avoids the throughput degradation caused by the combined effect of the packet loss in the wireless network and the congestion control mechanism. This basically reduces the total transfer delay and WNIC power consumption.

A few other solutions [26, 24, 23, 43] also use I-TCP scheme and turns off the WNIC during *idle* period. The work proposed in [26], defines new network architecture, named Power-Saving Network Architecture (PSNA) for accessing the Internet from the mobile client. Here, the connection between the

mobile client and the access point is based on a new protocol Power Saving Transport Protocol (PS-TP), whereas the communication between the access point and the fixed host is classical TCP. PS-TP is a reliable connectionless protocol. The main purpose of using connectionless approach is to reduce the amount of work performed by the mobile client and the energy cost associated with the set-up and tear down of a connection. In their next work [23], *Anastasi et al.* introduced an application independent approach, which is able to save energy around 70% without a significant degradation in the QoS perceived by the user.

However, potential energy is consumed when the WNIC is waiting for the data after sending the PS-poll packet to access point, as shown in [44, 32]. During this period no data is transmitted or received but drains a large amount of energy. The work in [22], proposes Self Tuning Power Management (STPM) protocol. It observes the network access pattern for different applications to predict the application behavior. Based on this prediction STPM decides whether to use PSM or CAM (Continuous Active Mode) and reduces power consumption.

3.4.2 Multimedia Streaming

However, the web access solutions are not suitable for power saving in multimedia applications due to the time constraints imposed by the applications and the quality of the stream. In addition, multimedia applications are considered to be resource hungry and most of solutions described below did not consider basic PSM.

Chandra et al. in their earlier work [31], explored WNIC power consumption implications for popular multimedia formats like Windows Media, Real Media and Quick Time. They also developed history based client side strategy to reduce the power consumption of the WNIC and techniques to switch the WNIC to the *sleeping* mode during the predicted *idle period*. The *idle period* is predicted by averaging the previous *idle periods*. In case of *too conservative idle period* there is no energy saving, whereas *too aggressive idle period* leads to packet loss. Using this approach it is possible to save 80% power for 28.8Kbps Windows Media stream and for higher stream rate it is 57%, where the packet loss is 2% and 0.3% respectively. The outcome seems positive, as windows media packets tend to arrive at regular intervals. However, this approach was not successful for Quick Time media streams, as the packet interval was irregular and such packets are harder to predict.

In their next work [32], *Chandra et al.* found that 802.11 power saving mode

is not suitable for multimedia streaming as the multimedia traffic tends to be irregular. Therefore, they proposed a server side proxy for shaping media streams. They showed that server-side application specific traffic shaping along with client-side history based prediction could save up-to 80% energy required for network reception.

In [53], *Chandra et al.* proposed a refinement to their earlier works [32, 31] by developing a statistical policy, where they found that *idle periods* have statistical correlation with the previously observed *idle periods*. In order to ensure that WNIC wakes up before the next packet arrival, a relatively small *negative bias* is added to the *sleep* interval. This approach yields lower packet loss and power consumption compared to the history-based prediction. Besides, it seems to be more robust to predict the *idle periods* for three media stream formats, namely Microsoft Media, Apple QuickTime and Real.

Shenoy et al. in their work [49] proposed a proxy-based solution. Proxy has two main functionalities: power friendly transcoding of media streams and an intelligent transmission technique to reduce the power consumption of the clients WNIC. The main goal is to reduce the energy consumption at the CPU and the wireless network interface card. The client sends its energy budget for stream decoding, network reception and the maximum spatial resolution it supports along with the request. In addition, client also specifies the type of its CPU and network interface. The proxy does transcoding based on these information received from the client.

For power friendly transmission *Shenoy* present two techniques: i) client-side history based prediction and ii) proxy-assisted prediction. For the former case, proxy does traffic shaping by converting VBR (Variable Bit Rate) stream to Constant Bit Rate, which helps the client to predict the idle interval to switch to the sleep mode. In case of proxy-assisted prediction, proxy sends a control packet to the client to indicate the arrival of the next set of frames and client WNIC goes to the *sleep* mode until that specified time. The researchers proved that their proposed solution saves 65-98% power during network reception.

In their latest work [25], *Anastasi et al* introduced a power saving streaming protocol to reduce the power consumption at the Wi-Fi interface of the mobile device, named Real Time Power Saving Protocol (RT_PS). Its works on the top of UDP and requires a proxy-based architecture. RT_PS enables proxy to schedule packet transmission by following an ON/OFF scheme. During ON period proxy transmits packets to the client with the highest possible rate allowed by the wireless link. Proxy decides the duration of the ON period by the current available bandwidth. Client switches off the WNIC during

the OFF period. The duration of the OFF period depends on the clients buffer level. When the buffer level falls down a dynamic level, client warns the proxy about the risk of the playback starvation and OFF period ends. In addition to the audio/video length, client buffer size, current available bandwidth, RT_PS power management policy also depends on the initial playback delay. Their implemented prototype shows that, this solution makes an energy saving ranging from 76% to 91% while maintaining the quality of the stream.

Two similar works [51, 56] focus on TCP based streams. They use server-side traffic shaping, which is implemented by exploiting TCP's flow control mechanism at the client side. At the beginning client sends the request with TCP receiving window size zero and later sends the request with appropriate window size to make the server to send the packets in a burst. Accordingly, client can predict the arrival and termination of a packet burst and turn off WNIC.

Gundlach et al. [37] proposed a technique to save energy for multiple clients by using a transparent proxy. Here, proxy schedules the packets to multiple clients on a TDMA approach and broadcasts a *schedule message* (a UDP packet). The *schedule message* defines the length of the burst for each client and the order of the burst. At the same time proxy transmits data to the clients according the order defined in the broadcast message. The proxy also sets the type of service bit of the IP header of the last packet of the data burst, so that the mobile client can make the transition of the WNIC to the *sleep* mode on time, therefore can save energy.

Janet at el. in [20] introduces a server side traffic shaping mechanism, called Adaptive-Buffer Power Save Mechanism (AB-PSM). Here, traffic is buffered at the server for certain period of time. This buffering period is selected as the multiples of the AP beacon interval. Unlike other techniques discussed above, AB-PSM assumes that basic PSM is active at the client side. However, AB-PSM is able to save a significant amount of energy by controlling the traffic transmitted over WLAN.

3.5 Differences with Previous Works

In Section 3.4 we have discussed existing solutions for normal web access (Section 3.4.1) and multimedia streaming(Section 3.4.2). Most of the energy management solutions for multimedia streaming based on the following approaches: i) switching the WNIC to the sleep mode during the idle periods,

and ii) reducing the total transfer delay. Client-side and proxy-assisted prediction or feedback was used to switch the WNIC to the *sleep* mode and total transfer delay of a multimedia session was reduced through the reduction of the stream fidelity.

However, prediction based solutions can suffer from packet loss if the prediction is not correct, thus the quality of the stream. In addition, they are client centric and require one client module, which causes extra energy cost.

Transcoding techniques can reduce transfer delay by introducing shorten audio/video size. However, while saving little energy, transcoding may cause degradation in the quality of the stream. In addition, only video streaming can be benefited from transcoding.

Besides, most of solutions are for RTSP-based Windows Media, RealNetworks and QuckTime streaming services. These services have their own extensions for RTSP protocol [38], which have to be implemented individually for a general purpose RTSP proxy.

In this thesis work, we propose a proxy-based solution for Internet radio streaming. The proxy works at the application layer and based on pure I-TCP. It is also independent of client and server. It receives stream traffic from the server on behalf of the client and relays the traffic to the mobile client at periodic intervals. Here, we assume that 802.11 PSM active at the client. Energy efficiency is achieved by increasing the periodic intervals while maintaining the stream encoding rate and quality of the stream perceived by the user. Our solutions do not require any modification to underlying protocol suite, 802.11b/g device driver or the AP.

We perform a complete study of this energy consumption behavior by increasing the periodic intervals while implementing the proxy locally and remotely. In our work, we also implement a transparent proxy locally and compare the performance with the application layer proxy. Furthermore, we analyze the 3G power consumption behavior with the application layer proxy.

Chapter 4

Power Consumption: Mobile Internet Radio

High profile mobile devices include WLAN, 3G interfaces to provide data services to the applications requiring high bandwidth. This integration pushed up the popularity of the Internet Radio streaming in mobile devices where the radio stations provide continuous broadcasting of simulated live or live streams to the users at different encoding rates.

This chapter describes our target architecture, application and power measurement tool. It also presents the mobile power consumption behavior while playing radio stations in WLAN condition with different stream rates. Finally we present our proxy mechanisms to save power for Internet radio streaming.

4.1 Mobile Device Architecture

Throughout our experiment we played and listened Internet Radio on Nokia E-71 [12] running Symbian OS S60 3rd Edition [16]. It is equipped with WLAN and 3G (up to 3.6Mbps) interfaces support to provide seamless data connectivity for applications like Mobile YouTube, Internet Radio, etc. Its key specifications are:

1. Operating System, Symbian S60 3rd Edition.
2. CPU, 369MHz ARM 11 Processor.
3. RAM, 128MB.

4. WLAN, 802.11b/g.
5. 3G, 3.6 Mbps.
6. Battery, 1500mAh.

Symbian is a mobile OS for ARM processors. The system includes a multi tasking microkernel OS, associated libraries, user interface and the reference implementation of common tools. The microkernel works in a server-based fashion where the access to the server is asynchronous and based on event passing. The selection of the server is limited to the four services: Generic OS Services, Common Services, Multimedia and Graphics, and Connectivity [52]. Figure 4.1 illustrates the layers of the Symbian OS System Model.

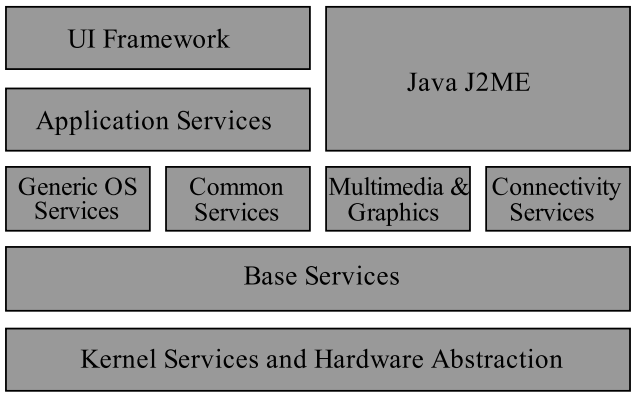


Figure 4.1: Symbian Architecture

The Base Services layer is the lowest reachable layer by an application that includes the File Server, User Library, cryptographic services, etc. It is also responsible for the basic network connectivity. The communication infrastructure is developed on this layer and two other protocol stacks: TCP and WAP. Besides, some other narrow band protocols, infrared and bluetooth are also supported.

The WLAN package comes with Symbian is hardware independent and implements the basic functionalities like access point selection and roaming, scanning logic, association state machine, admission control, etc. The WLAN chip vendors implements a hardware specific driver below the WLAN Hardware Abstraction Layer (HAL) API defined in the WLAN package [17].

4.2 Mobile Internet Radio Application

Internet Radio applications on mobile phones may use different streaming protocols and services. The application comes with Nokia E-71 uses SHOUTcast ICY HTTP streaming protocol (Section 2.3.2). When a user selects a radio station to play initially it takes five to ten seconds to fill out the application's play-out buffer and then starts playing. The application continues to play the stream until the user quits or stops the application. Figure 4.2 shows the format of a ICY radio stream request sent from E-71.

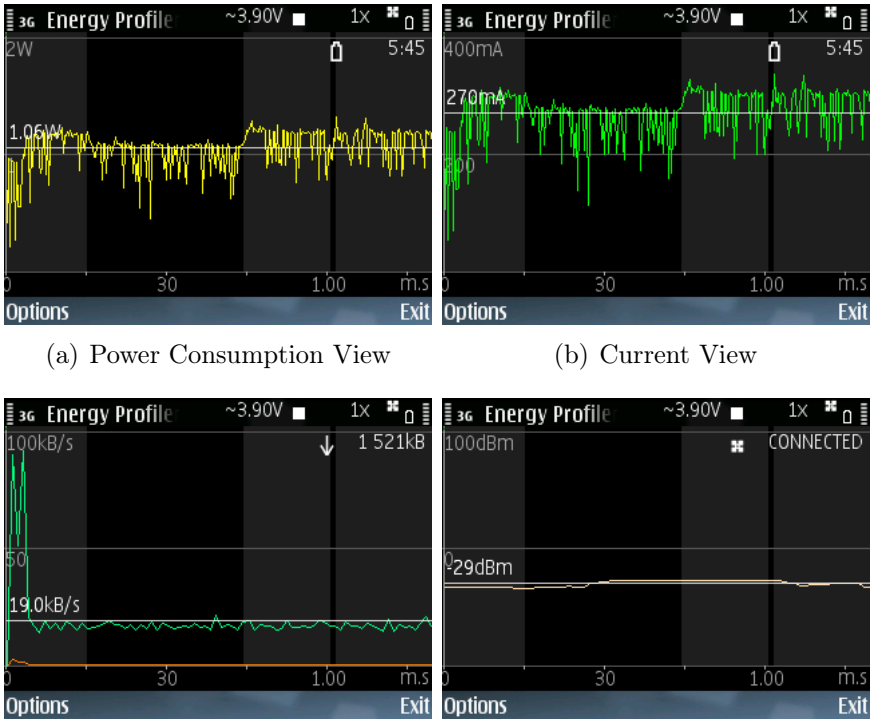
```
GET / HT TP/1.0
Host: ra dio.inte
rnode.on .net:810
0 User- Agent: i
RAPP/1.9 .0 Nokia
E71-1/20 0.21.118
; Series 60/3.1 P
Profile/ MIDP-2.0
  Configu ration/C
LDC-1.   Accept:
*/*  Icy -MetaDat
a:1  Con nection:
  close
```

Figure 4.2: ICY Request Format

At the beginning of the connection, the streaming stations use *fast streaming* (Section 2.4.2) to fill out the client application play-out buffer. Soon after the buffer is filled, server sends traffic at the stream-encoding rate. Here, the radio streaming stations support multiple encoding rates to enable users to listen with different bandwidth Internet connections. For instance, *radio.internode.on.net* radio station supports streaming at 32, 64 and 128 kBps.

In contrast to the *stream switching* (Section 2.4.3) for Windows media or QuickTime media services, it is not possible to switch to another encoding rate over the same TCP connection while listening radio. It requires closing the current connection and then establishing a new TCP connection to another port for a different streaming rate. For this reason, the streaming rate is generally decided by the application at the moment of the connection establishment based on the available bandwidth or the user can select to listen a particular streaming rate supported by the radio station.

4.3 Power Measurement Tool



(a) Power Consumption View (b) Current View

(c) Data Traffic Acitiviy (d) WLAN Signal Strength

Figure 4.3: Nokia Energy Profiler Views

In order to measure the power consumption we used Nokia Energy Profiler (NEP) [13] in our work. It is an application for Symbian S60 3rd Edition and later Nokia devices. It enables to measure the power being consumed by an S60 device and provides information on key power consuming tasks like CPU, WLAN, IP traffic activity, etc.

1. Power, current, voltage consumption
2. WLAN signal strength
3. 3G Timers
4. Data transmission speed
5. Memory usage

6. Processor activity.

While running an application, energy profiler can be run in the background to profile every power consuming events. NEP makes easy identification of the power consumption by screen shot or generating CSV output.

The figure 4.3(a) shows the average power consumption over a period of more than one minute. The timer at the upper right corner indicates the estimated lifetime of the battery before it discharges completely. The next picture 4.3(b) illustrates the average current consumption. As the battery drains out with time, the voltage decrease but the current increases and the power level remains constant. Figure 4.3(c) shows the upload and download link speed in kbps. The orange line indicates upload and green indicates download link. The last figure 4.3(d) indicates the WLAN received-signal strength when the device is connected to a access point and values are typically within the range of -25 to -90dBm, where the values closer to -40 is considered to be very good condition.

4.4 AP and Mobile Device Configuration

In our experiment, we have used one D-LINK DIR-301 [4] access point to support WLAN connectivity for devices with 802.11b/g interface. The access was restricted through the use of WPA enhanced security feature. Table 4.1 lists the parameters configured in the access point.

Parameters to Set	Values
TX Rates	54Mbps (Maximum).
Transmit Power	100%.
Beacon Interval	100ms.
RTS Threshold	2346 bytes.
Fragmentation	2346 bytes.
DTIM Interval	1.
CTS Mode	Auto.
WMM Function	Disable.
RF Channel	Auto.

Table 4.1: Wireless AP Configuration.

WMM (Wi-Fi Multimedia) [21] is an Wi-Fi Alliance interoperability certification based on 802.11e [50] standard to provide QoS (Quality of Service)

to 802.11 networks. It works by prioritizing the traffic according to four Access Categories: voice, video, best effort and background. It is suitable for applications like audio and video telephony over IP. However, it does not guarantee the required throughput.

In addition to the AP, Nokia E-71 was also configured. 3G, Bluetooth interfaces were turned off while using WLAN. The phone speaker volume was mute while running the Internet radio. The device WNIC was in PSM mode most of the time and the Energy Profiler was always running at the background.

Parameters to Set	Values
3G	Disabled.
Speaker Volume	Muted.
WNIC	PSM disabled - enabled.
Nokia Energy Profiler	Running.

Table 4.2: E-71 Configuration.

4.5 Mobile Power Consumption

In handheld devices, WNIC is the most power hungry component and streaming applications are the most power consuming applications. 802.11 PSM helps to reduce the power consumption by switching the WNIC into sleeping mode during the idle periods. *Chandra et al.* [32] showed that PSM is effective only when traffic is regular or periodic. Since most of the media traffic is irregular in nature, it is not suitable to use 802.11 PSM with streaming applications. Therefore, we have noticed that most of the solutions described in the previous chapter (Section 3.4) assumed that PSM is not active.

4.5.1 Mobile Internet Radio

We have worked with three different Internet Radio Stations with different media encoding rates from the different parts of the world. While listening to these radios on E-71 over WLAN we have noticed that battery drains out very quickly.

From Table 4.3 we can see that if WNIC is in continuous active mode (CAM) at the mobile client, the power consumption is almost same for all radio

Radio Stations	Data Rate	Power(PSM)	Power(CAM)
stream.rawfm.com.au:8004	8kBps	0.53W	1.06W
radio.internode.on.net:8100	16kBps	0.99W	1.07W
83.145.201.209:8000	24kbps	1.04W	1.07W

Table 4.3: Power Consumption of Internet Radio Sites

stations. It does not make any difference whether the media-encoding rate is lower or higher, because the WNIC is always on. On the contrary, we find that mobile power consumption is less with the lower encoding rates while PSM is active. But the effect is not significant for higher data rate radio streaming. The power consumption behavior is more clarified in figure 4.4

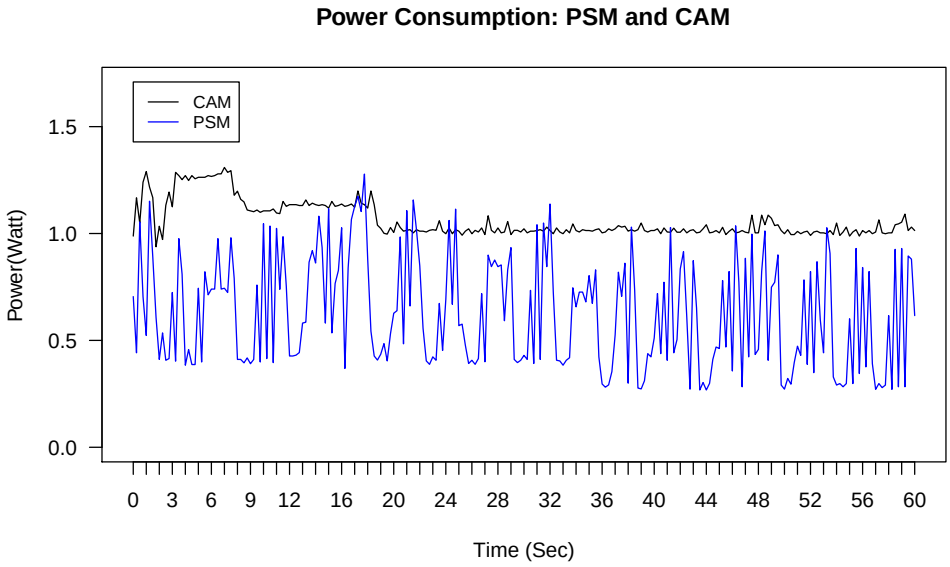


Figure 4.4: Power Comsumption: CAM and PSM

4.5.2 Reducing Power Consumption

In order to make PSM more effective while listening to a Internet radio station, we need to shape the traffic at the server end or somewhere else in between the radio station and the client. Instead of the server we have two separate proxy server implementations to shape the traffic, i.e. to generate

the periodic bursts. One of the proxies is transparent and the other one works at the application layer. Here, the proxy servers buffer the media stream, generate *bursts* periodically, and reduces the number of *bursts* by increasing the *buffering period* in order to reduce the power consumption by the mobile client. The properties of the bursts generated by the proxy after a buffering period:

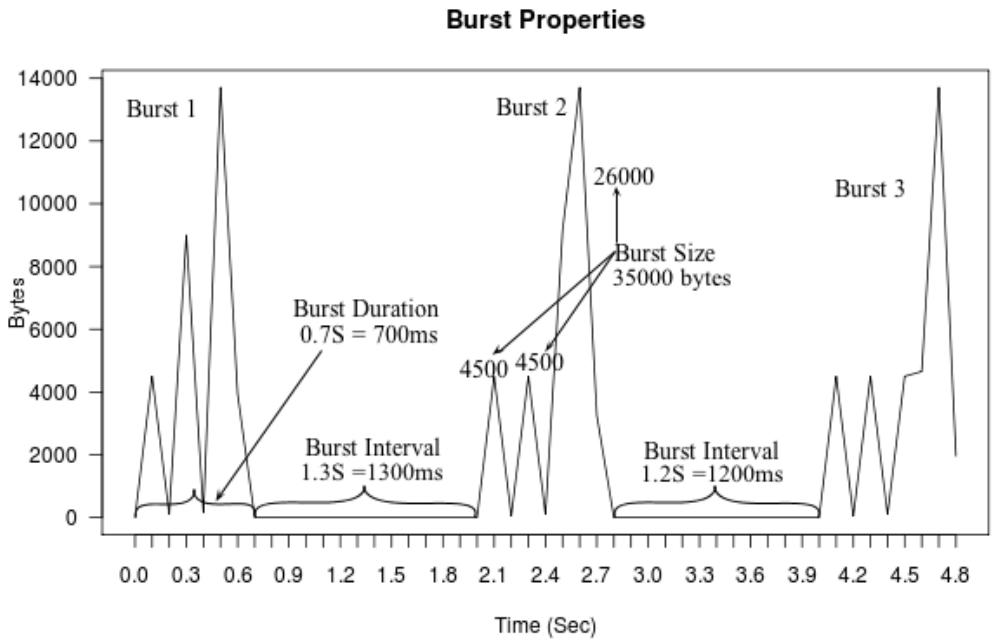


Figure 4.5: Burst Properties

1. *Burst:* The media data buffered by the proxy server for a period of time to sent to the mobile client.

2. *Burst Size:* The size of the buffered data in kilo bytes.

3. *Burst Duration:* Time required to complete a single burst i.e. to send the buffered data from the proxy server to the client for one time.

4. *Burst Interval:* The time period between two consecutive bursts. Most of the time during this period Proxy usually buffers media stream. It is the difference between the buffering period and the burst duration.

Figure 4.5 illustrates the burst properties in detail. There are three bursts. The duration of the second burst is 800ms and the burst size is 35kBytes. The burst interval or buffering period between the first and the second burst is around 1300ms.

4.5.3 Application Layer Proxy

This application layer proxy is based on I-TCP scheme. Client sends the radio-playing request to the proxy server using ICY protocol; proxy makes another TCP connection with the Radio Streaming server and relays the ICY response containing the radio server information followed by continues flow of MP3 or AAC media stream to the client player.

I-TCP mechanism is implemented using TCP server and client stream sockets at the proxy. Clients connect to the server socket and proxy connects to the radio server using the client socket. Afterward media data is simply forwarded from one socket to another socket. We used simple process forking mechanism to handle multiple clients connecting to the same radio server.

We introduce two timers for each client: *start-up* and *buffering period*. The first timer starts only when the proxy forwards the stream request to the server. Consequently, the streaming server sends traffic to the client using *fast streaming* to fill the play-out buffer. Here we used a fixed timer of ten to twenty seconds. During this period proxy simply forwards the traffic to the client and measures the amount of data transmitted to the client. Proxy also calculates the maximum amount of time that the client player can play, if the connection is disturbed or broken.

Immediate next the second timer is activated and proxy buffers until the timer expires. Then, proxy sends the buffered traffic in a single burst to the mobile client. Then again the second timer is initialized and this process continues until the connection is closed. In this case, we assume that proxy knows the media encoding rate of the radio stream from the server ICY's response. The values of the timers are constant throughout a connection.

If the encoding rate is E bytes per second and K bytes are transferred to the client during the initial *start-up* time T, then the maximum amount of time the proxy can buffer without affecting the stream quality and smooth playback:

$$n = \frac{K}{T} \tag{4.1}$$

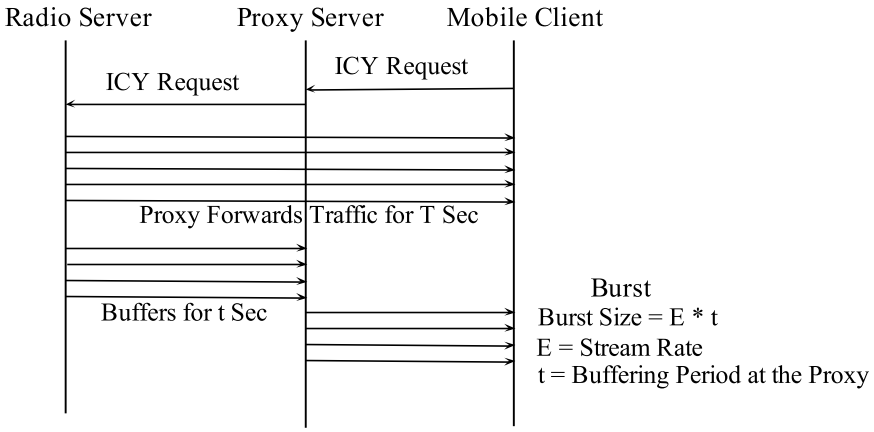


Figure 4.6: Proxy Mechanism

Here the amount of data to be sent in a single burst or burst size is the product of the encoding rate E and the buffering period t:

$$BurstSize = E \times t \tag{4.2}$$

The value of t ranges from *0* to n.

4.5.4 Transparent Proxy

The proxy is implemented using *IPQ*, a built-in packet filter in *iptables* [11], to catch all incoming and outgoing packets. Then we used *libipq* [45] to read take the packets from kernel space to user space and send back unmodified packets to the kernel space. However, the packets to be modified were dropped in the kernel-space and sent through raw socket after modification because the packets must be returned to the kernel space in the same order as they were retrieved.

The proxy is transparent, the clients and server do not know about the existence of the proxy. We found that it is convenient to implement the proxy at the gateway, as it does not require ARP poisoning. Proxy tracks the beginning of a TCP connection by seeing a SYN request to a particular IP address and port and keeps this *connection information* in a pool when the connection is matured, i.e. three-way handshake is completed. It also uses two timers, similar to the application layer proxy.

A complete *connection information* contains IP addresses, port numbers,

sequence and acknowledge numbers, the current state of a TCP connection and the connection type (whether the connection is a stream connection or not). A connection is removed from the pool as the connection is terminated either by the client or the server. However proxy does not generate any new SYN request or any other packets, it simply forwards, delays and modifies the IP packets.

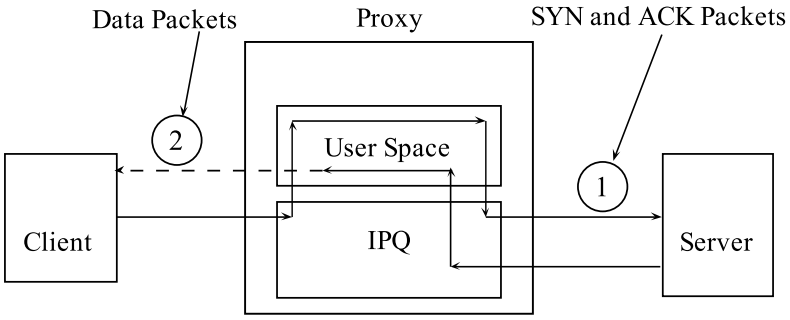


Figure 4.7: Transparent Proxy Mechanism

Proxy finds out a radio stream request by looking into the ICY header sent by the client immediate after the three-way handshake of the connection establishment. The connection type of the *connection information* is updated only when the server sends the response of the request with the header Content-Type of audio.

Chapter 5

Proxy Performance over WLAN

Like other streaming applications Mobile Internet Radio is also high power consuming. While listening to a Internet radio station, even with lower streaming rate over WLAN, it is possible to drain out the battery (1500mAh) within few hours. And battery drains out more quickly with higher streaming rate radio stations. In Chapter 4 we have observed that 802.11 power saving mechanism does not have significant impact on the total power consumption while running radio. For this reason, we proposed two proxy server mechanisms to work together with PSM.

In this chapter we present the combined effect of WLAN PSM and the proxy server on mobile power consumption with different streaming rate radio stations. We place the proxy at different locations, then observe and analyze the behavior of the mobile device power consumption. With local application layer proxy we also allow other clients with the same access point to download at higher data rate or to run other streaming applications.

5.1 Local Application Layer Proxy

At the beginning of our experiment we have hosted the proxy in a Linux machine in the same local network with the mobile client. The communication between the proxy and the AP is on the wired medium, whereas with the mobile client is over WLAN. Figure 5.1 describes the network configuration.

We have conducted our experiments on three Internet radio stations (Table 4.3). Two of them are hosted in Australia and one is in Finland. Each of these radio stations we have played around n times through the proxy server and every time the playing duration have been about 20 minutes. Then

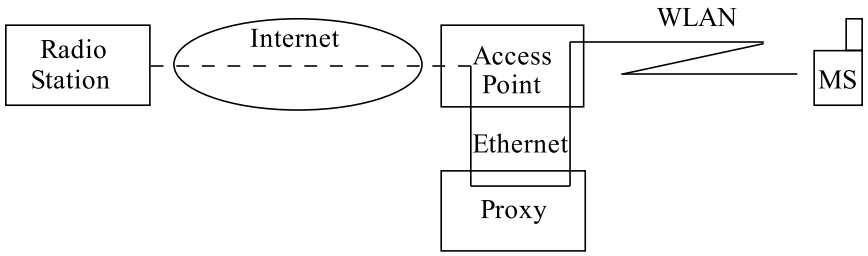


Figure 5.1: Local Proxy Scenario

we have taken the average power consumption from the NEP reading. This average power consumption includes the initial high power due to the *start-up* buffering at the application, display and back light.

5.1.1 Results

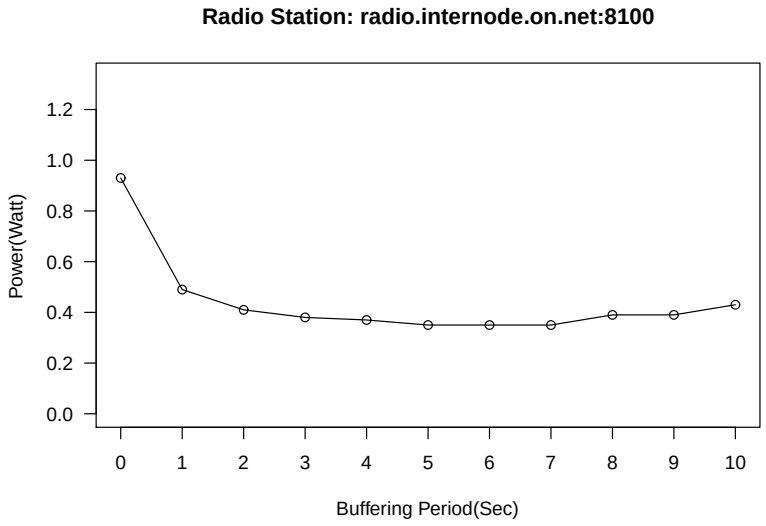
The initial *start-up* time for *radio.internode.on.net* is about 10 seconds. Before sending the stream request to the proxy server we have calculated this time by listening the radio directly for a while. Then this *start-up* time has been used at the proxy to forward the traffic without buffering. The amount of data transferred during this period to the client using *fast streaming* is 410016 Bytes (K). With a streaming rate of 16000Bps (E) the maximum buffering period at the proxy has been $n = K/E = 25$ seconds.

After that, proxy has buffered the radio traffic for a fixed period of t seconds (t=*1,2,3....10*<n) and then sent the buffered data to the client in a single burst. The Figure 5.2(a) shows the power consumption while listening this radio with different buffering periods.

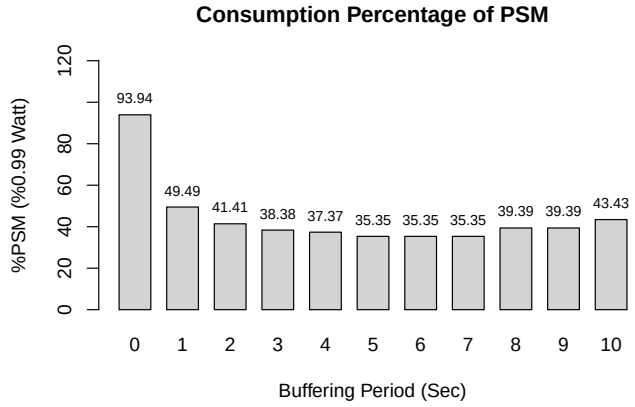
Radio Stations	Data Rate (E) kBps	Buffering Period (t)S	Burst Size (E x t) KBytes	Minimum Power (W)
stream.rawfm.com.au	8	12,13,14	96,104,112	0.32
radio.internode.on.net	16	5,6,7	80,96,120	0.35
83.135.201.209	24	4	96	0.37

Table 5.1: Minimum Power Consumption for Radio Sites

From the graph 5.2(a) we can see that buffering at the proxy even for a very smaller period of time contributes in over all power saving. The performance



(a) Power Consumption Graph



(b) Percentage of PSM (% 0.99Watt)

Figure 5.2: Mobile Device Power Consumption

can be further improved by increasing the buffering period at the proxy; meanwhile the quality of the stream is also maintained. The maximum power saving is achieved when the power consumption is reduced from 0.93 Watt to 0.35 Watt, while the proxy buffers for 5, 6 or 7 seconds (Table 5.1). If we compare with standalone PSM performance, this is about 35.35% percent of PSM (Figure 5.2(b)).

For two other radio stations power consumption is also reduced in similar fashion. In case of *stream.rawfm.com.au* the minimum consumption is 0.32 Watt which is about 60.38%of with basic PSM. On the other hand for *83.145.201.201* the minimum power consumption is about 37% of PSM.

5.1.2 Analysis

From a small snapshot of our results, Table 5.1, it is certain that, our application layer proxy is successful in reducing power consumption. In addition, it is noticeable that this reduction maintains a particular pattern. And this pattern is consistent for all radio stations; power consumption decreases as the buffering period increases at the proxy, but after a certain buffering period, again consumption tends to increase (Figure 5.2(a)).

Since PSM is active, most of the time client WNIC is in sleeping state during the *burst intervals*. And WNIC wakes up each time proxy sends a burst and then again switches to the sleep state once the burst is complete.

For instance, while listening *radio.internode.on.net*, with the buffering period of 1 sec, there will be around 30 bursts generated by the proxy within 30 seconds of playing and 5 bursts with 6 seconds of *buffering period*, hence WNIC will be awaking at least 5 times to receive the bursts (Figure 5.3). Therefore, the number of bursts decreases as the buffering period increases and hence power consumption decreases.

In order to have an insight about the effect of underlying transport protocol on the application layer bursts, for each of the buffering periods we have captured traffic of around 10000 packets. Then we have computed the number of bursts and the average burst duration, assuming the minimum burst duration of 50ms (Table 5.2). Two consequtive transmissions are considered two separate bursts if they are 200ms apart from each other. But this computation does not include the traffic for the initial *start-up* time.

However, we notice that, the number of application layer burst remains intact till with buffering period 6 seconds. After that number bursts begins to increase and therefore the power consumption (Table 5.2).

Because, here the bursts are delivered using TCP and it is observed that, TCP affects the proxy bursts if they overflow the clients receiving buffer. Consequently client TCP sends zero window advertisements (ZWA) to the sender to stop the transmission. Immediate after some buffer ($>$ 1 MSS) is freed and client sends the requests with proper receiving window size. Therefore, the original single burst is split into several bursts at the transport

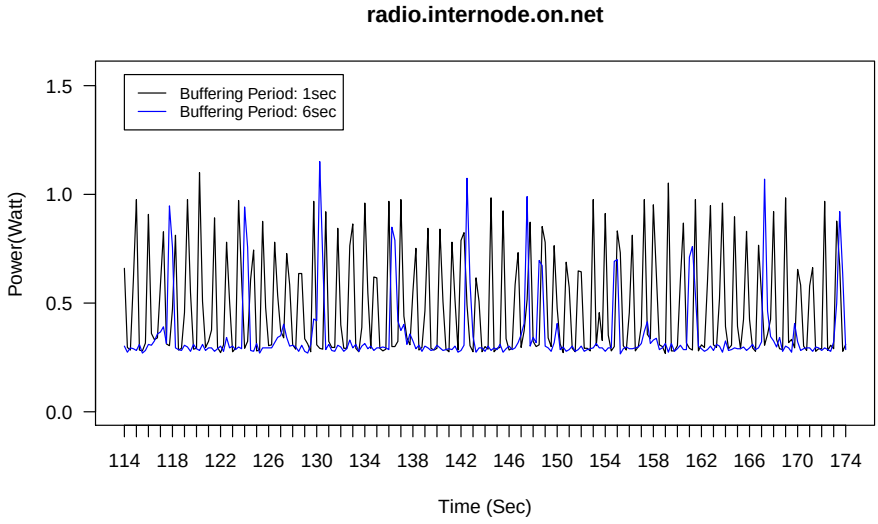


Figure 5.3: Power Consumption with Different Buffering Periods

Buffering Period (S)	Estimated Bursts	Realized Bursts	Zero Window Adv.	Power Consumption (W)	Power Saving (%)
1	600	599	0	0.49	50.51
2	300	290	0	0.41	58.59
3	200	199	0	0.38	61.62
4	150	145	0	0.37	62.63
5	120	120	0	0.35	65.65
6	100	99	0	0.35	65.65
7	86	97	11	0.35	65.65
8	75	88	21	0.39	60.41
9	67	190	215	0.39	60.41
10	60	227	288	0.43	57.57

Table 5.2: Bursts and Power Consumption (*radio.internode.on.net*)

layer (Figure 5.4). For Instance, while the buffering period is of 10 seconds, the burst size is about 160kBytes and the number of ZWA is 227 (Table 5.2). Therefore, even though the number of estimated burst was 60 but 288 burst was realized, which eventually helped to increase the total power consumption.

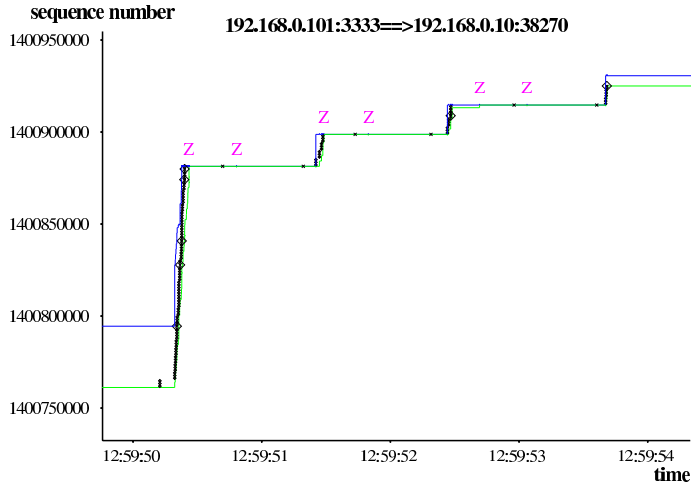


Figure 5.4: Time Sequence Graph(radio.internode.on.net): 10Sec Buffering Period

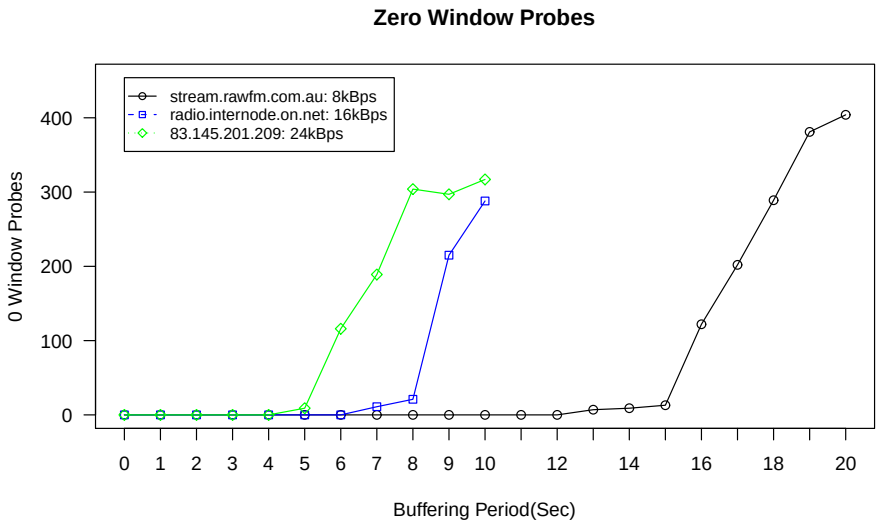


Figure 5.5: ZWAs with Different Buffering Periods

We find that there are three buffering periods (5, 6 and 7 seconds) for which power consumption is minimum (Table 5.2). But the transmission of zero window probes begins with buffering period 7 seconds. For two other radio stations zero window probes begin with 13 and 5 seconds (Figure 5.5) respectively. Hence it is always possible to find out the *optimam buffering period*

for all radio stations and it is the maximum buffering period for which the transmission of zero window advertisement is about to zero. In addition, the burst size also about 96KBytes (Table 5.1).

5.2 Remote Application Layer Prxoy

In chapter 3.4 we have outlined some solutions based on server side traffic shaping. In most cases, there is a proxy at the server end to do the shaping. But if the shaper is not closer to the client, there is a good probability that the bursts generated by the proxy might be affected by the Internet architecture or by the corresponding protocols behavior. In order to realize those effects on the proxy bursts and thus power consumption, we have done extensive lookout by setting up our proxy at different regional locations.

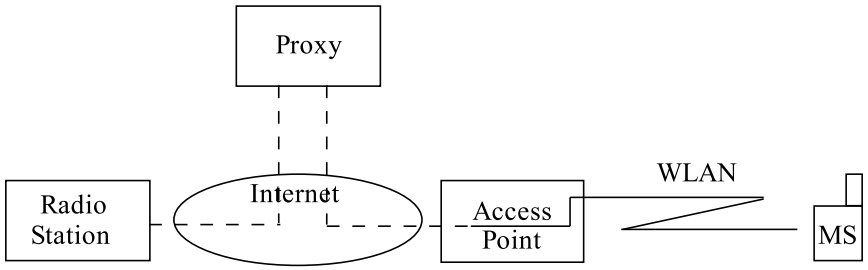


Figure 5.6: Remote Proxy Scenario

5.2.1 Results

Apart from the local network we also have deployed the proxy at four other locations, where two locations have been very close to the mobile client within 7 hops. For two other cases, we have used Amazon EC2 Linux boxes in UK and USA respectively. Similar to the local proxy setup all three radio stations have been played and tested. Likewise, the initial *start-up* time is 10 seconds for *radio.internode.on.net* and the maximum buffering period is around 25 seconds. But we have measured power consumption with maximum buffering period of 10 seconds.

The finding is similar to that of local proxy scenario. Power consumption decreases till some buffering periods and after that again aims to increase.

Therefore, the power consumption pattern is preserved (Figure 5.7) and consistent.

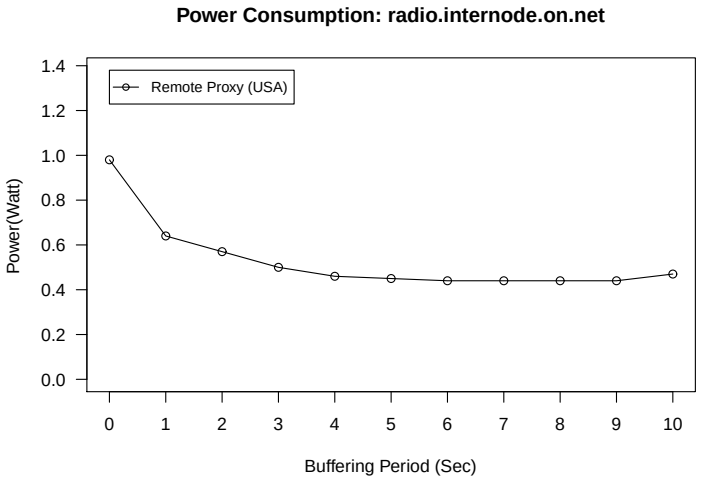


Figure 5.7: Power Consumption: Remote Proxy Location (USA)

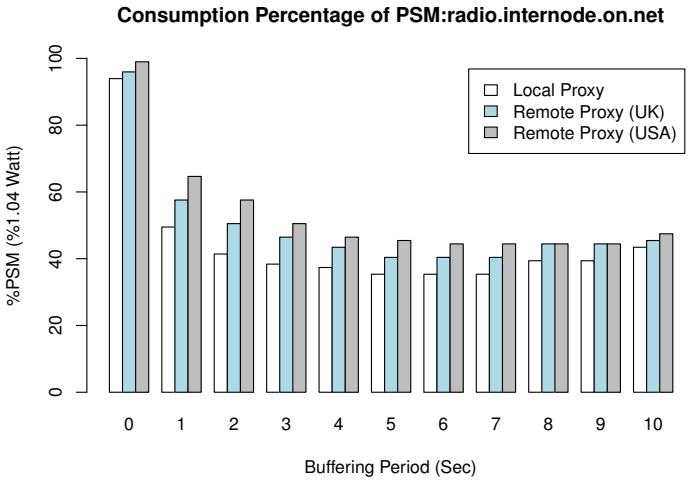


Figure 5.8: Power Consumption: Percentage of 0.99 Watt

While using local proxy, the RTT from the proxy to the client is about 4ms. The other proxy locations with very smaller RTT difference from the local

stream.rawfm.com.au				
Optimum Buffering Period(S)	Proxy Address	RTT (ms)	Burst Duration (ms)	Minimum Power(W)
12	192.168.0.101	3.01	289.03	0.32
	gul4.kyla.fi	4.3	259.13	0.32
	mh-a135-ext.cs.hut.fi	5.09	283.76	0.32
	79.125.8.152 (UK)	65.7	556.96	0.34
	174.129.12.210 (USA)	108.7	1101.85	0.37
radio.internode.on.net				
Optimum Buffering Period(S)	Proxy Address	RTT (ms)	Burst Duration (ms)	Minimum Power(W)
6	192.168.0.101	3.0	297.78	0.35
	gul4.kyla.fi	4.1	294.06	0.35
	mh-a135-ext.cs.hut.fi	5.3	282.18	0.35
	79.125.8.152 (UK)	52.1	594.54	0.40
	174.129.12.210 (USA)	108.6	1218.32	0.44
83.145.201.209				
Optimum Buffering Period(S)	Proxy Address	RTT (ms)	Burst Duration (ms)	Minimum Power(W)
4	192.168.0.101	3.7	269.3	0.37
	gul4.kyla.fi	4.29	276.53	0.37
	mh-a135-ext.cs.hut.fi	4.7	291.11	0.37
	79.125.8.152 (UK)	53.9	612.87	0.44
	174.129.12.210 (USA)	106.0	957.24	0.49

Table 5.3: Minimum Power Consumption at Different Proxy Locations

proxy do not exhibit significant difference with the local proxy scenario for at any buffering period (Table 5.3).

However, if the proxy moves further away from the server, we find that power consumption increases. For instance, out of these four remote locations the maximum consumption happens when the proxy is in USA. In this case, for *radio.internode.on.net* the average RTT is 108.6ms and minimum power consumption is 0.44 Watt which is 44.44% of basic PSM(0.99W). For the proxy setup in UK the minimum power consumption is 35.35% of PSM. So, from the result described in Table 5.3 and illustrated in Figure 5.8 we find

that mobile power consumption increases as the RTT or propagation delay increases between the client and the proxy.

5.2.2 Analysis

Since, the underlying transport protocol is TCP, the amount of data can be transmitted into the network is controlled by three important variables: congestion window (CWND) size, receiving window (RWND) size advertized by the client and another variable which is used to select the congestion control algorithm. CWND determines the amount of data that the TCP sender can send to the client before receiving an ACK, where RWND limits the outstanding data.

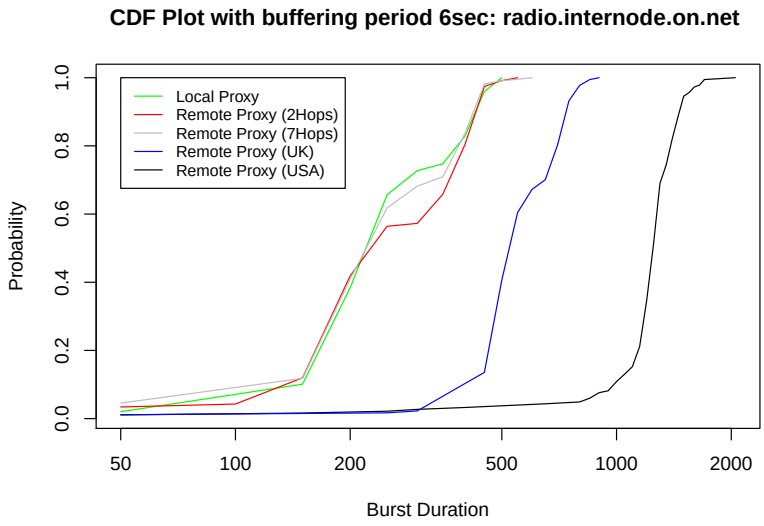
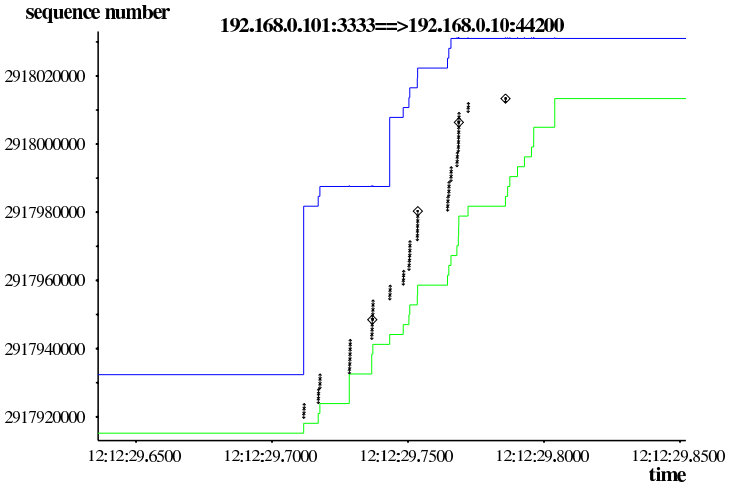


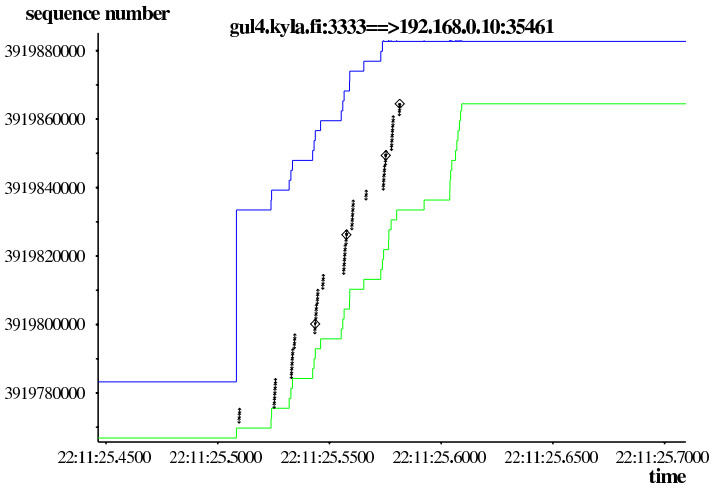
Figure 5.9: CDF of Burst Duration for All Proxy Locations

The size of the CWND is doubled once an ACK is received. But, if there is an increment in RTT in the network due to the distance or propagation delay, the ACK of a packet is delayed and therefore the TCP sender also delays the increment of the CWND. Since CWND is responsible for the amount of the data that can be transmitted in to network, the increase of RTT causes the sender to decrease the number of packets to be sent.

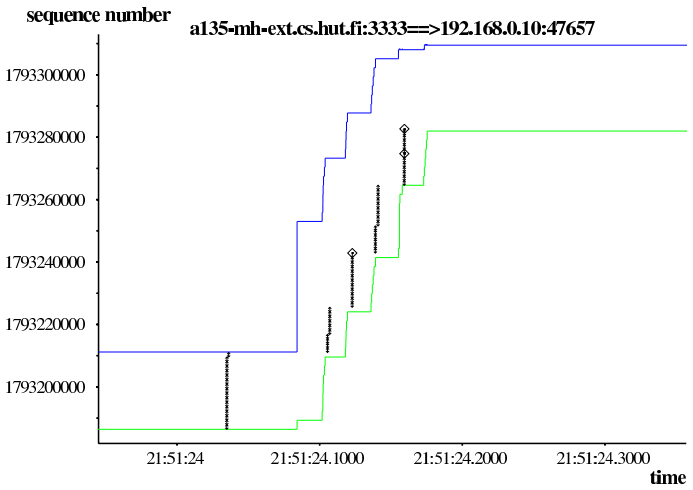
For this reason, the TCP sender on the proxies which are very close to the client with smaller RTTs takes very short to time fill out the congestion



(a) Local Proxy

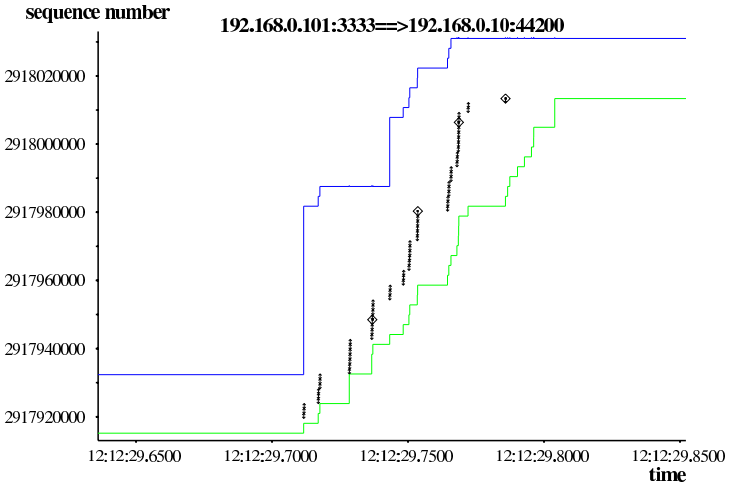


(b) Remote Proxy (Hop2)

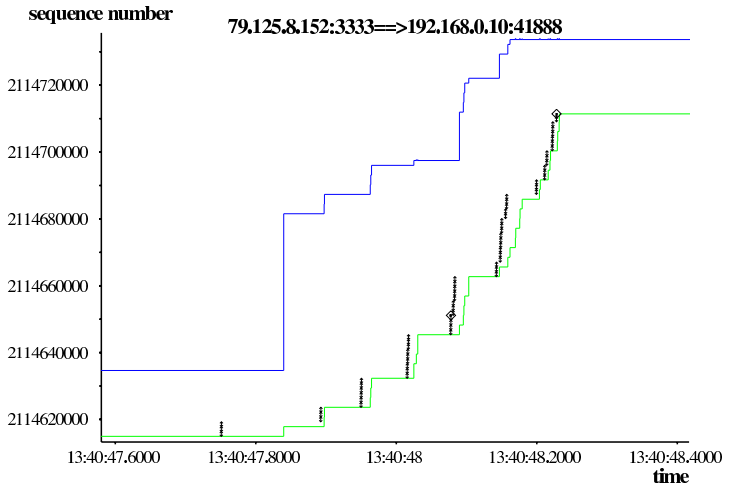


(c) Remote Proxy (Hop7)

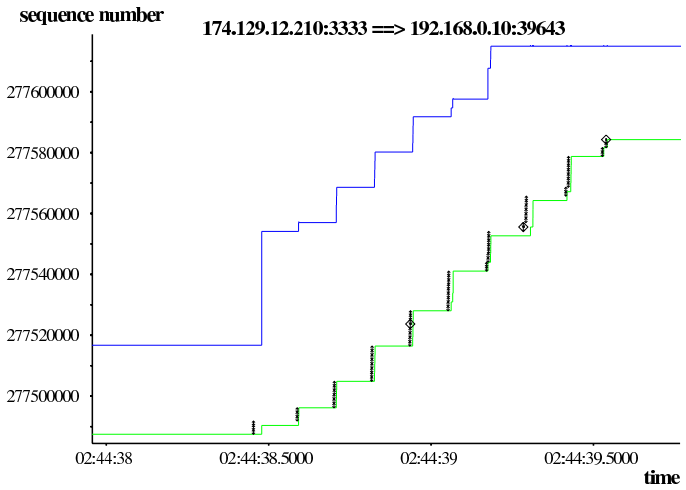
Figure 5.10: Time Sequence Graph: Scenario One



(a) Local Proxy



(b) Remote Proxy (UK)



(c) Remote Proxy (USA)

Figure 5.11: Time Sequence Graph: Scenario Two

window (Figure 5.10) and therefore the smaller burst duration. On the other hand if the proxy moves far from the client the RTT increases and it takes longer period for the TCP sender to fill the congestion window (Figure 5.11) and consequently burst duration increases.

The CDF plot (Figure 5.9) illustrates the burst duration with different proxy locations while the buffering period is of 6 seconds. We find that average *burst duration* for nearby proxy locations are comperatively less and very close of each other. On the contrary, for two other proxy locations the *burst duration* is high. Therefore, even though the buffering period is same for all proxy locations, the *burst interval* decreases as the *burst duration* increases. In other words, mobile client WNIC is spending more time in the *receiving state*, which eventually increases total power consumption.

5.3 Application Layer Proxy with Cross Traffic

radio.internode.on.net					
Buffering Period	Power Consumption at the Mobile Client				
	0KBps (Bytes)	100KBps (W)	200KBps (W)	520KBps (W)	1.2MBps (W)
1	0.49	0.50	0.50	0.58	0.53
2	0.41	0.42	0.42	0.43	0.46
3	0.38	0.39	0.39	0.38	0.41
4	0.37	0.37	0.37	0.37	0.40
5	0.35	0.36	0.36	0.36	0.38
6	0.35	0.37	0.36	0.36	0.38
7	0.35	0.37	0.37	0.37	0.40
8	0.39	0.40	0.39	0.42	0.42
9	0.39	0.41	0.42	0.43	0.45
10	0.43	0.42	0.45	0.45	0.47

Table 5.4: Mobile Power Consumption with different download rates at the Desktop

Besides serving a single mobile client we have also checked the power consumption behavior locally in cross traffic scenario with another client under the same AP. Here the second client is a desktop machine and connected to the AP with a DLink USB wireless interface. The desktop downloads from web at different rates; meanwhile the mobile client plays the radio stream.

We have used trickle to perform downloading at the desired rates on the second client. Table 5.4 shows the power consumption detail of the mobile client while the other client is downloading at different rates. Download rate 0KBps condition is same as the local proxy scenario (Section 5.1).

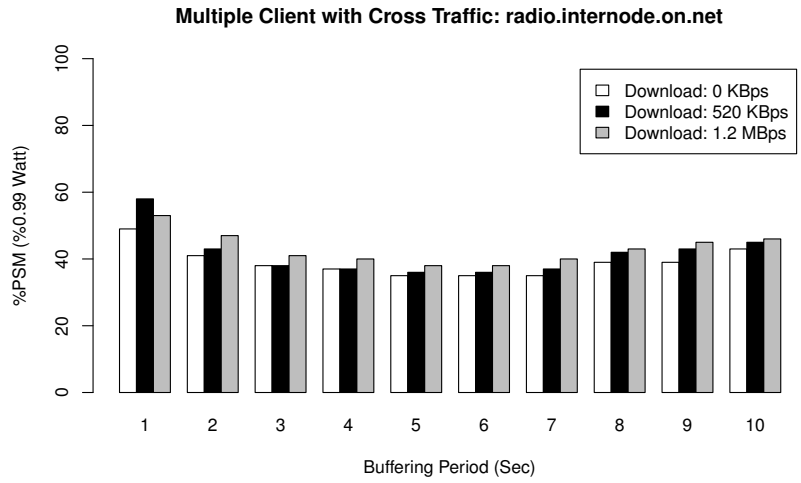


Figure 5.12: Local Application Layer Proxy Performance with Cross Traffic

radio.internode.on.net			
Buffering Period	0KBps (Bytes)	520KBps (Bytes)	1.2MBps (Bytes)
1	1432	38346	5504
2	0	3836	25837
3	5027	4473	28212
4	4945	10374	21877
5	16443	11584	2223
6	3027	15941	40553
7	5178	14480	36387
8	0	10498	11010
9	3992	11928	4133
10	8909	15928	11584

Table 5.5: Retransmission for the Mobile Client

We find that, power consumption at mobile client exhibits a small variation with lower downloading rates for any buffering period. But if the download

rate is around 520kBps or further beyond that (For instance 1.2MBps) then there is an indicative increment in power consumption as compared to the local proxy (single client) and other downloading conditions (Figure 5.12). With 520KBps downloading rate the minimum consumption is 36% of PSM and for 1.2MBps minimum power consumption is around 38% of PSM with buffering period of 6 seconds.

When the download rate is 0KBps, there is negligible packet loss in the network. The packet loss is not momentous in the network until the other client is downloading at the maximum rate caused by the implicit congestion in the network. Therefore, there is transmission of SACK packets and consequently retransmission in the network. These SACKs and retransmissions have effect either on the burst duration or on the creation of new more bursts and therefore power consumption. Table 5.5 describes the retransmission for the mobile client while the desktop client is downloading at 0KBps, 520KBps and 1.2MBps.

5.4 Transparent Proxy

In our thesis work, we have also implemented and deployed one transparent proxy server to shape traffic in the local netwok. Proxy is implemented at the gateway and therefore, parses every HTTP request it sees in order to find whether it is a radio stream request or not and then imposes the traffic shaping mechanism. Unlike, application layer proxy, TCP connection is not split. Proxy simply forwards the traffic for the initial *start-up* time, then buffers the incoming packets for the client radio application and next sends the packets to the client like the other proxy.

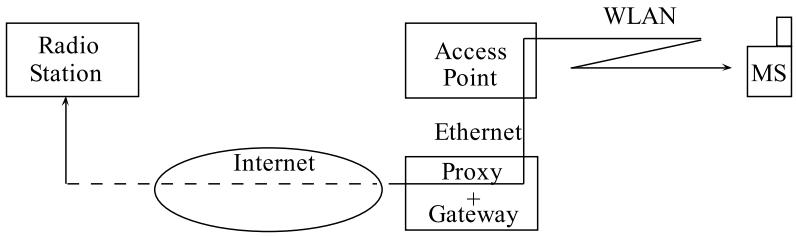


Figure 5.13: Network Configuration for the Transparent proxy

From Table 5.6 we notice that shaping traffic at the transparent proxy also reduces the power consumption at the mobile client. For *radio.internode.on.net*

Buffering Period (Sec)	Power Consumption for Internet Radios		
	stream.rawfm. com.au (Watt)	radio.internode. on.net (Watt)	83.145.201.209 (Watt)
0.0	0.57	1.00	1.07
0.5	0.49	0.71	0.73
1.0	0.46	0.53	0.53
1.5	0.43	0.48	0.5
2.0	0.43	0.43	0.45
3.0	NA	NA	NA
.	NA	NA	NA
.	NA	NA	NA
10.0	NA	NA	NA

Table 5.6: Power Consumption with Transparent Proxy

the amount of power saved is 57% and for two other radio stations the savings are 8.11% and 56.73% respectively.

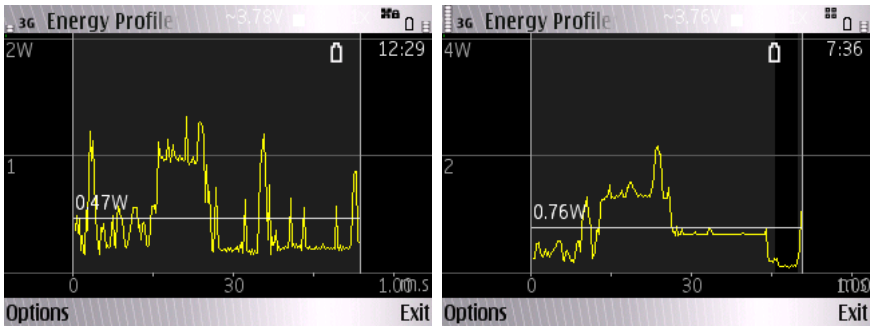
We observe that unlike the application layer proxy, we can not delay the packets more than 2 seconds. Otherwise, the smooth playback is suffered. Since the proxy mechanism is not I-TCP and packet receving and sending are implemented under one thread, more delaying the packets at the proxy causes TCP to expire the retransmission timer and to trigger the congestion control mechanism. However, up-to 2 seconds buffering period, the performance is same as the appliation layer proxy.

Chapter 6

Proxy Performance over 3G

Along with WLAN, 3G networks also provide high-speed data service to the users for applications like Mobile TV, etc. It allows the applications to use data service at a rate of maximum 14.0 Mbps through download link and 5.4Mbits on uplink [1]. It also allows simultaneous data and voice call. But, like WLAN while using data service application over 3G, it is possible to drain out battery very quickly, may be even quicker than WLAN. Figure 6.1 shows the NEP snapshot of power consumption while sending one HTTP GET request via WLAN and 3G respectively.

In this chapter, we present the performance of our application layer proxy over 3G networks. We begin with a brief overview on 3G power saving mechanism. After that we introduce our exprement result and analysis while playing radio in 3G network condition.



(a) Local Proxy (b) Remote Proxy (Hop2)

Figure 6.1: WLAN and 3G Power Consumption

6.1 3G Power Consumption

Mobile power consumption, when using 3G, depends on the transmission length and efficient power management by RRC (Radio Resource Control) protocol. Here, RRC in 3G(WCDMA) maintains a number of states [57]. Figure 6.2 illustrates the state machine for the RRC states which is implemented by 3G(WCDMA) networks and follows 3GPP standard.

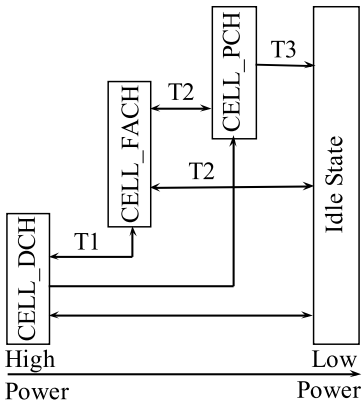


Figure 6.2: 3G(WCDMA) RRC State machine

In CELL_DCH state the mobile client is assigned a dedicated channel for specific data traffic requiring high throughput and very less transmission delay. Therefore, it is the most power consuming state. *Wigard et al.* [54] reports typical current consumption in this state is 200-400mA. In CELL_FACH state, the assigned channel is generally shared with other phones and used when the amount of traffic is very less and discrete. The power consumption in this state is about half of the CELL_DCH state. There is one optional state called CELL_PCH, which is used for paging. In this, state the mobile station is unable to send or receive any data but can be paged. Then after, the mobile station can switch to CELL_FACH or CELL_DCH state. Consequently the current consumption is also low and it is around 2 to 4mA. Finally, in idle state the mobile station is disconnected from the RRC but still it can have the IP address so that it can be paged and power consumption is same as in CELL_PCH state.

During each of these above-mentioned states the mobile station is assigned a particular channel. If one channel is not busy for a specific period of time than the mobile switched to other lower power consuming states. This timers are also state specific and informally named as T1, T2 and T3 and maintained

by the RNC (Radio Network Controller). T1 timer is for the CELL_DCH state and is reset on the availability of traffic. The value may depend on the data rate of the DCH and value decreases as the data rate increases. T2 timer is for CELL_FACH state and most cases the default value is 2 seconds. The third timer (T3) is for CELL_FCH state and its value can be several minutes [40].

6.2 Mobile Internet Radio and Proxy

Radio Station	Data Rate KBps	Power (3G-384Kbps) W	Power (3G-2.0Mbps) W
stream.rawfm.com.au	8	1.30	1.30
radio.internode.on.net	16	1.30	1.30
83.145.201.209	24	1.27	1.35

Table 6.1: 3G Power Consumption for Internet Radios

While playing Internet radio over 3G we have found that power consumption is very high (Table 6.1). Here, streaming rate of the radio stations do not have indicative effect on power consumption. So, similar to the remote proxy scenario we also have deployed the proxy in the Internet. Since, in case of 3G *idle state* and CELL_PCH are the lowest power consuming states (Figure 6.2), the idea is to keep the mobile idle during the buffering period at the proxy server and therefore minimizing the power consumption. We have used 3G with 2.0Mbps and 348kbps data rate services.

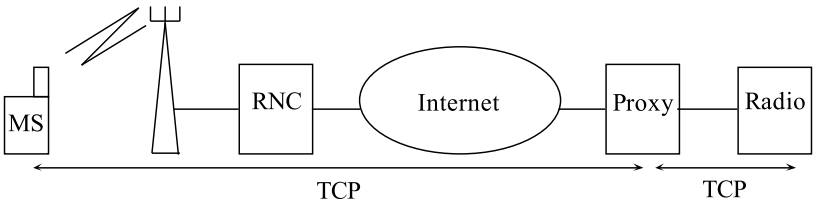
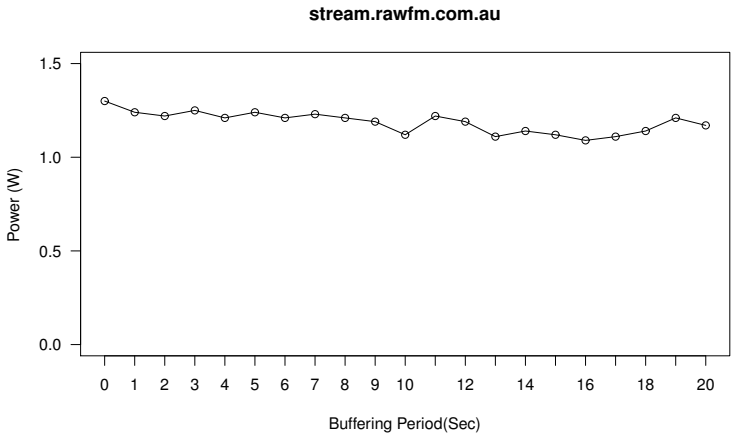
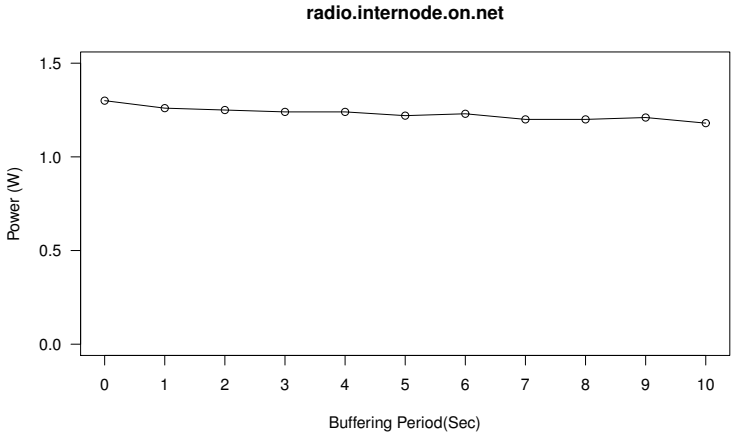


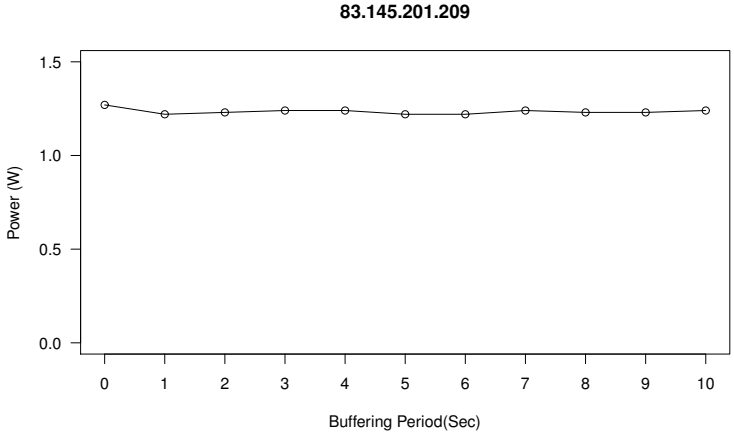
Figure 6.3: Proxy in 3G Network



(a) Audio Stream Rate 8KBps

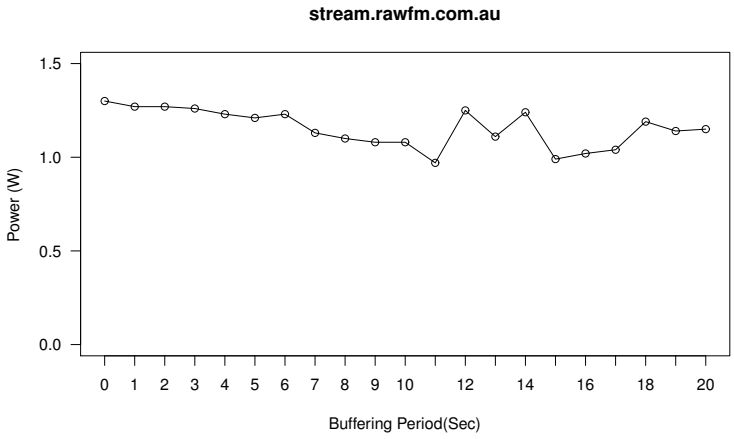


(b) Audio Stream Rate 16KBps

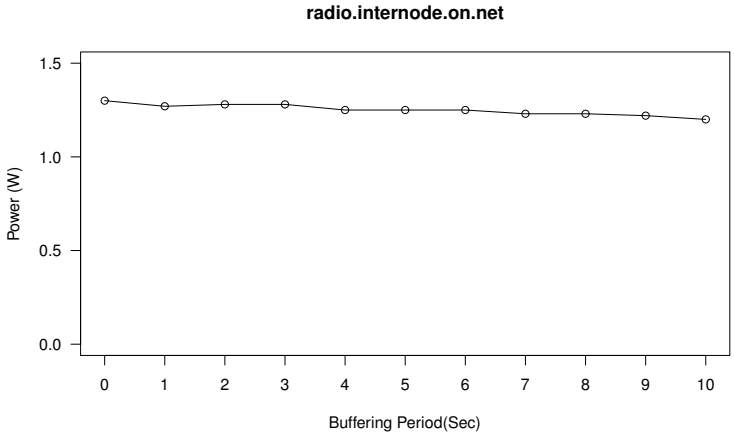


(c) Audio Stream Rate 26KBps

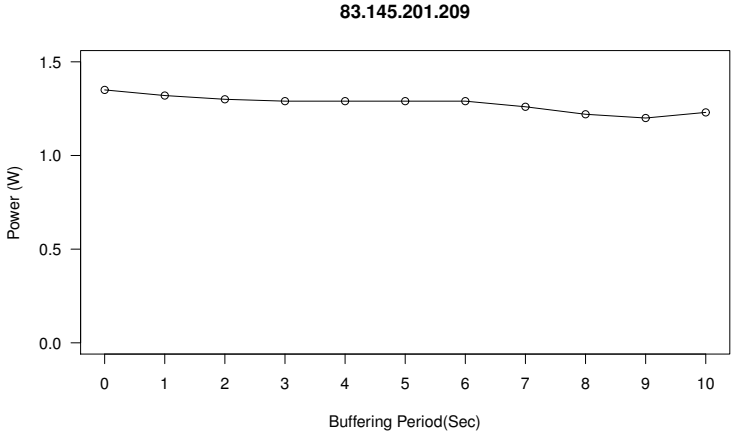
Figure 6.4: Power Consumption: Scenario One (3G:348kbps)



(a) Audio Stream Rate 8KBps



(b) Audio Stream Rate 16KBps



(c) Audio Stream rate 24KBps

Figure 6.5: Power Consumption: Scenario Two (3G:2Mbps)

6.3 Analysis

From Section 6.1 we find that inactivity timers are the key parameters which control the power consumption for 3G networks in a way that when a mobile station is idle in a channel for a certain time, the associated timer is expired and the mobile station releases the associated channel and transits from one state to another state. So, if the length of the timers are shorter then there will be very frequent reconnection to the RRC and longer inactivity timers will result higher energy consumption and inefficient use of radio resources.

Mobile Internet radio application receives continuous flow of audio stream from the station server and hence the mobile station is always in CELL_DCH state. For instance, if the streaming rate is 16KBps and T1 timer value is 2 seconds, then there is very less chance that the associated inactivity timer T1 will ever expire and the mobile station will switch to the idle mode. So, the power consumption will be always high.

For this reason, our intention behind using the proxy was to enable the T1 timer to expire during the *burst interval*. So, the mobile station will be in the *idle state* or in CELL_PCH state at least for *(Buffering Period - T1)* seconds. After that again the mobile station will be in the CELL_DCH state. Here, we do not know the exact configuration of the inactivity timers in the RNC of the BSs those our mobile client was connected to during the experiments.

Neverthless, from figures 6.4 and 6.5 we can infer that the effect of the proxy was very negligible irrespective of the data rate subscription. Unlike WLAN case, power consumption is almost same for all buffering periods and for the higher streaming rate radio station. But for the lower streaming rate station (*stream.rawfm.com.au*), there are some discrete behavior in power consumption.

From the above pattern of power consumption we realize that TCP might behave different in cellular networks, others than in the wired and WLAN environments. According to the RFC [42] 3G link layer characteristics have significant impact on the underlying transport protocol TCP. Consequently traffic, inactivity timers and hence, can be also on power consumption.

Latency is high in the 3G networks, mostly for the processing in the physical layer, i.e. FEC (Forward Error Correction), interleaving and transmission delays in the radio access network. So, the value of a RTT ranges from few hundred milliseconds to one second.

Besides, variable data rate can also contribute in RTT increment where data

radio.internode.on.net				
	3G (384Kbps)		3G (2.0Mbps)	
Buffering Period	ZWA Packets	RTT (ms)	ZWA Packets	RTT (ms)
1	202	364.2	0	225.3
2	200	352.9	0	251.3
3	194	341.8	0	183.6
4	195	373.3	226	210.4
5	158	353.5	239	201.8
6	201	396.9	181	217.7
7	131	555.5	242	194.0
8	103	689.9	228	198.9
9	102	542.7	208	204.7
10	132	576.3	227	209.2

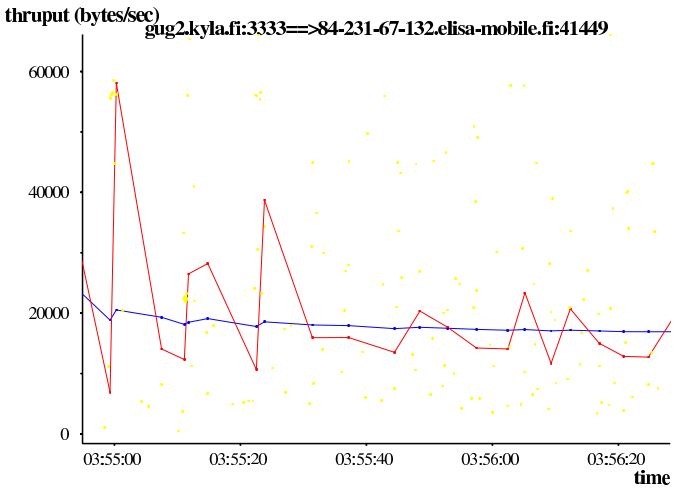
Table 6.2: RTTs of the mobile client and Transmission of ZWA over 3G

rate varies due to the effects from other users. Departing an arriving of users under the coverage of a cell may increase and decrease the available bandwidth of the cell. Therefore, a sudden increase in RTT may cause TCP timeout and retransmission. In our case, variable rate may contribute to the retransmission as the 3G networks were used in the university campus.

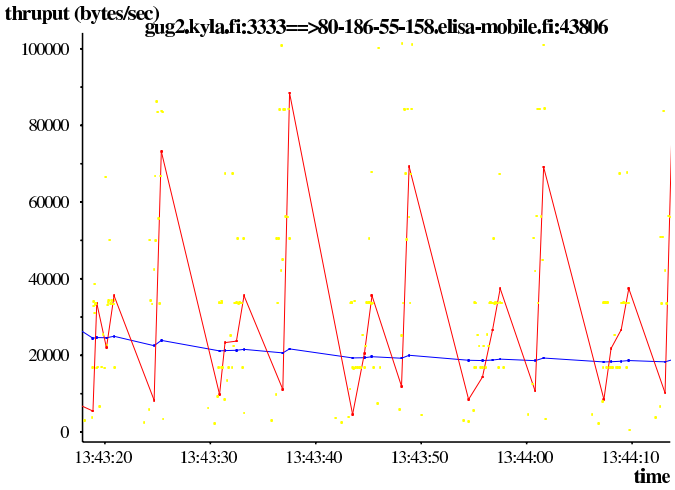
Retransmissions can also happen due to the delay spikes (sudden increase in the latency), which can happen due to the handover and suspension of the lower priority data transfer by the radio network scheduler in order to allocate that channel to higher priority users. These delay spikes may cause TCP timeouts and retransmissions. Besides, the implementation of various scheduling techniques may allocate and de-allocate radio channels in order to ensure the proper utilization of radio resources. Therefore, periodic allocation and de-allocation of channels may cause retransmission of packets [34].

But, the most significant consequence of retransmission on a TCP connection is the exponential back off the retransmission timer and reduction in the congestion window. And this repeated error may result small congestion window and very lower throughput throughout the connection. Even though the FEC may be sucessfull in combating high BER (Bit Error Rate) but the bandiwdth will be underutilized.

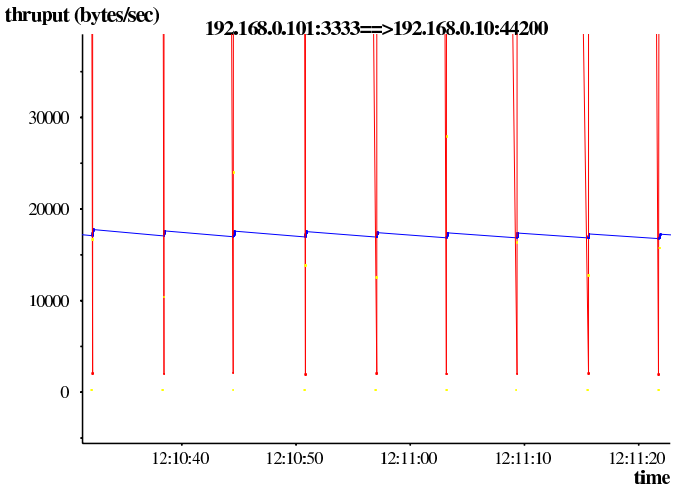
Some remarkable works have been done to imporve the performance of TCP by exploiting basic TCP flow control mechanism: Freeze-TCP [35, 33], M-



(a) Scenario: 3G (384Kbps)



(b) Scenario: 3G (2.0Mbps)



(c) Scenario: Local Proxy (WLAN)

Figure 6.6: TCP Throughput Graphs

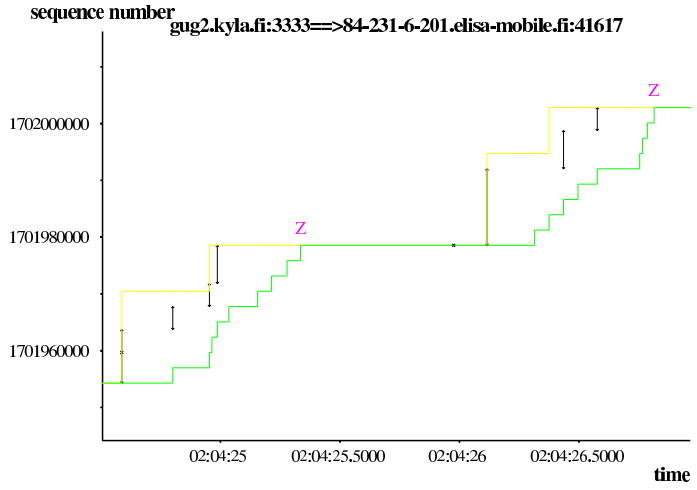


Figure 6.7: TCP Time Sequence Graph 3G(384Kbps)

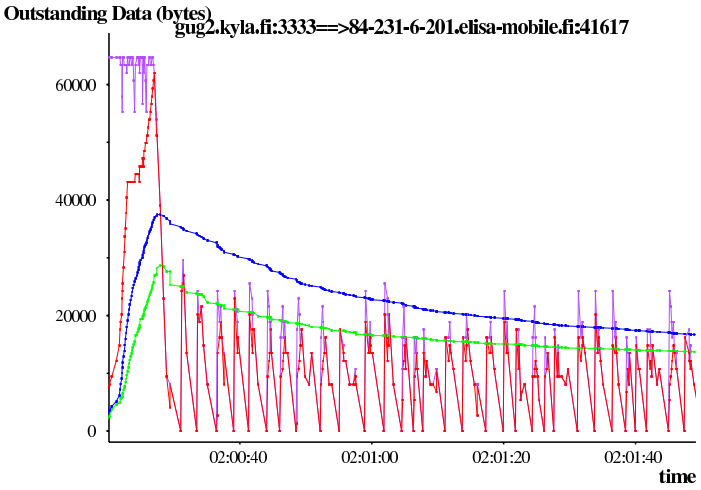


Figure 6.8: TCP Outstanding Graph: 3G(384Kbps)

TCP [29] and window regulator [30]. Freeze-TCP requires certain modification at the client TCP. Here, the mobile client monitors the signal strength and sends ZWA to the sender if it senses any disconnection. After reconnection, receiver sends triple ACKs for the last segment it received before the disconnection. On the contrary, ZWA is sent by the BS in M-TCP and acknowledges last (N-1) bytes. The last byte is acknowledged only after reconnection with simple Window update message. Where as, window regulator ensures that TCP sender operates in the window-limited regeion.

However, in case of M-TCP and window regulator, RNC maintains a buffer for each connection.

Therefore, the captured traffic at the proxy for each of the buffering period for the radio stations have been analyzed and found abnormal transmission of zero window advertisements (ZWA). Besides the average RTTs of the mobile station from the proxy were also about few hundred milliseconds (Table 6.2).

From section 5.1.2 we find that the effect of ZWA and retransmission on application layer bursts, where ZWA makes proxy bursts more bursty. From the throughput graph we find the in case of WLAN, this traffic behavior is according to the proxy burst generation while the proxy buffers for six seconds. On the other hand, in case of 3G throughput never goes to zero with the same buffering period, hence the traffic channel is always busy and the inactivity timers never expire. Where retransmission contributes either to increase the burst duration or creates smaller bursts (Section 5.3) and higher RTT increases burst duration as the CWND increases slowly (Section 5.2.2).

In our case, we could not find any triple ACKs after ZWAs from the mobile client. In addition, we have not observered any such situation where ZWA is sent while acknowledging N-1 bytes or window update with single byte acknowledgement after any ZWA (Figure 6.7). Rather, most cases we have observed dynamic oscillation in the receving window size advertised by the client. Hence, it is possible to have very frequent and a lot of ZWA due to the very limited recevier buffer size. Figure 6.8 illustrates the outstatindig data graph with buffering period of one sec where the purple color indicates the receiver advertised window size.

Chapter 7

Conclusion

In this thesis work we have developed two different proxy servers: Application layer proxy and Transparent Proxy. We have implemented them for Linux OS and deployed them at different network locations, conditions and evaluated their effect on mobile power consumption for one kind of *live* or *simulated live* audio streaming (Mobile Internet Radio).

This chapter presents a summery of our work and final outcomes achieved through different experiments. It also illustrates the limitations of our present solutions and some suggestions for our future work in this area.

7.1 Thesis Contribution

This thesis work proposes proxy-based power saving solutions for Internet radio audio streaming where the application layer proxy works on I-TCP scheme and transparent proxy simply delays the packets at the user space received from the kernel.

We have done extensive study and analysis the effects of our proxies on mobile power consumption. Initially we have tested the mobile client in the WLAN scenario. We have deployed the proxy in the same local network with the mobile client. We have discovered that power consumption on the mobile client can be reduced to a significant amount of 65% as compared to the power consumption with 802.11 PSM. We also tested multiple client scenario in the local network and found that congestion in the network threats power saving.

Likewise, we also have setup the proxy at some far remote locations (UK,

USA) and observed that power consumption increases at mobile client. We have found that RTT from the proxy to the client is a function of CWND of the sender's TCP, so burst duration and finally power consumption at the client.

Additionally, we have also used our proxy through 3G networks. However the achievement was not significant as were in WLAN scenarios. We have observed that traffic flow behavior in 3G networks has been very discrete due to the link characteristics and consequently the proxy burst intervals did not have significant impact on power inactivity timers and therefore power consumption.

Furthermore, we have also explored the underlying transport layer protocol TCP response on proxy traffic shaping mechanism in both WLAN and 3G network and the consequences on proxy performance and mobile power consumption.

7.2 Limitations of the Proxy Approaches

The performance of our proxies has been significant in case of WLAN. In case of application layer proxy, we have observed that the minimum power consumption is restricted to the burst size of around 96kBps. This is irrespective of the streaming rate and *burst interval*. Beyond above this size, power consumption again increases. We have also shown that this phenomenon can also be detected by observing the flow of ZWA. But our current proxy does not include this mechanism to always operate on the optimum buffering period. In addition, this proxy receives media data from the streaming server for a buffering period, accumulates in a single buffer and then writes the buffer on to the client sockets. But this receiving from server and sending to the client are implemented under single thread. Therefore, delay in reading from the server or writing buffer to the client socket may affect the overall performance.

Likewise, the transparent proxy was also effective in power saving but not so much successful like the other proxy. Receiving packets from the kernel, buffering and sending back are bind together in a single thread. Therefore, if the proxy buffers more than 2 seconds then TCP congestion control mechanism triggers due to the RTO expiration.

7.3 Future Work

In section 7.2 we have identified several limitations of our current solutions. The performance can be further improved by separating the receiving and sending tasks by two threads. In addition, one monitor module can be added with the proxies to make sure that they always operate at the optimum buffering period.

Apart from those, we have also noticed that underlying protocol TCP also have effect on power consumption. If case of WLAN, we have observed how RTT, RWND size and retransmission of packets affect mobile power saving. In 3G networks, TCP has been further optimized to increase the throughput, which makes it impossible to save power over 3G networks for real time applications like audio streaming. Therefore, studying the effect of TCP behavior on mobile power consumption can be the potential candidate for our further research and study.

Bibliography

[1] 3g wiki. urlhttp://en.wikipedia.org/wiki/3G.

[2] Apple quciktime streaming server modules programming guide. http://developer.apple.com/legacy/mac/library/documentation/QuickTime/QTSS/Concepts/QTSSConcepts.html.

[3] Apple-quicktime. http://www.apple.com/quicktime/.

[4] D-link dir-301 wireless router. http://developer.symbian.org/wiki/index.php/SHAI_WLAN_HAL_API_Specification.

[5] Microsift windows media service http streaming. http://msdn.microsoft.com/en-us/library/bb905764.aspx.

[6] Microsoft streaming media basics:real media surestream. http://service.real.com/help/library/guides/producerplus8/htmfiles/preparin.htm.

[7] Microsoft windows media. http://www.microsoft.com/windows/windowsmedia/default.mspx.

[8] Microsoft windows media: Fast streaming. http://www.microsoft.com/windows/windowsmedia/technologies/bettertogether.aspx.

[9] Microsoft windows media:intelligent streaming. http://www.microsoft.com/windows/windowsmedia/howto/articles/intstreaming.aspx.

[10] Microsoft windows media:protocol rollover. http://msdn.microsoft.com/en-us/library/dd757582(VS.85).aspx.

[11] Netfilter/iptable project home page. http://www.netfilter.org/.

[12] Nokia e-71 business phone. http://www.nokia.co.uk/find-products/all-phones/nokia-e71.

[13] Nokia energy profiler: Quick start. `http://www.forum.nokia.com/Technology_Topics/Application_Quality/Power_Management/Nokia_Energy_Profiler_Quick_Start.xhtml`.

[14] Real media. `http://www.real.com/`.

[15] The shoutcast radio. `http://www.shoutcast.com/`(visited on 14/07/2009).

[16] Symbian s60 3rd edition. `http://developer.symbian.org/wiki/index.php/Symbian\%5E3`.

[17] Symbian wlan hardware abstraction layer api specification. `http://developer.symbian.org/wiki/index.php/SHAI_WLAN_HAL_API_Specification`.

[18] The shoutcast streaming standard. `http://forums.radiotoolbox.com/viewtopic.php?t=74`(visited on 16/07/2009).

[19] Rtp control protocol extended reports (rtcp xr), 2003.

[20] Adams, J., Gabriel-Miro, and Muntean. Adaptive buffer power save mechanism for mobile multimedia streaming. Performance Engineering Lab, Performance Engineering Lab.

[21] Alliance, W.-F. Support for multimedia applications with quality of service in wi-fi networks, 2004. urlhttp://www.wi-fi.org/.

[22] Anand, M., Nightingale, E. B., and Flinn, J. Self-tuning wireless network power management. In *MobiCom '03: Proceedings of the 9th annual international conference on Mobile computing and networking* (New York, NY, USA, 2003), ACM, pp. 176–189.

[23] Anastasi, G., Conti, M., Gregori, E., and Passarella, A. Balancing energy saving and qos in the mobile internet: An -independent approach. In *HICSS '03: Proceedings of the 36th Annual Hawaii International Conference on System Sciences (HICSS'03) - Track 9* (2003).

[24] Anastasi, G., Conti, M., Gregori, E., and Passarella, A. Performance comparison of power-saving strategies for mobile web access. *Perform. Eval. 53*, 3-4 (2003), 273–294.

[25] Anastasi, G., Conti, M., Gregori, E., Passarella, A., and Pelusi, L. A power-aware multimedia streaming protocol for mobile users. In *Proceedings of the International Conference on Pervasive Services 2005* (2005), pp. 371–80.

[26] Anastasi, G., Conti, M., and Lapenna, W. Power saving pollicies for wireless access to tcp/ip networks.

[27] Apple. `http://www.apple.com/`.

[28] Bakre, A. V., and Badrinath, B. R. Implementation and performance evaluation of indirect tcp. *IEEE Trans. Comput. 46*, 3 (1997), 260–278.

[29] Brown, K., and Singh, S. M-tcp: Tcp for mobile cellular networks. *ACM Computer Communication Review 27* (1997).

[30] Chan, M. C., and Ramjee, R. Improving tcp/ip performance over third generation wireless networks, 2004.

[31] Chandra, and Surendar. Wireless network interface energy consumption: implications for popular streaming formats. *Multimedia Syst. 9*, 2 (2003), 185–201.

[32] Chandra, Surendar, Vahdat, and Amin. Application-specific network management for energy-aware streaming of popular multimedia formats. In *ATEC '02: Proceedings of the General Track of the annual conference on USENIX Annual Technical Conference* (Berkeley, CA, USA, 2002), USENIX Association, pp. 329–342.

[33] Costa, G., and Sirisena, H. Freeze tcp with timestamps for fast packet loss recovery after disconnections. 1792–1799.

[34] F. Kahfizov and M. Yavuz. TCP over CDMA2000 networks. Internet-Draft draft-khafizov-pilc-cdma2000-00.txt.

[35] Goff, T., Moronski, J., Phatak, D. S., and Gupta, V. Freeze-tcp: A true end-to-end tcp enhancement mechanism for mobile environments. In *In Proceedings of IEEE INFOCOM'2000, Tel Aviv* (2000), pp. 1537–1545.

[36] Group, A.-V. T. W., Schulzrinne, H., Casner, S., Frederick, R., and Jacobson, V. Rtp: A transport protocol for real-time applications, 1996.

[37] Gundlach, M., Doster, S., Yan, H., Lowenthal, D. K., Watterson, S. A., and Chandra, S. Dynamic, power-aware scheduling for mobile clients using a transparent proxy. In *ICPP '04: Proceedings of the 2004 International Conference on Parallel Processing* (Washington, DC, USA, 2004), IEEE Computer Society, pp. 557–565.

[38] Guo, L., Tan, E., Chen, S., Xiao, Z., Spatscheck, O., and Zhang, X. Delving into internet streaming media delivery: a quality and resource utilization perspective. In *IMC '06: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement* (New York, NY, USA, 2006), ACM, pp. 217–230.

[39] Handley, M., and Jacobson, V. Sdp: Session description protocol, 1998.

[40] Haverinen, Henry; Siren, J. E. P. Eenergy consmption of always-on applications in wcdma networks.

[41] Helmbold, D. P., Long, D. D. E., and Sherrod, B. A dynamic disk spin-down technique for mobile computing. In *MobiCom '96: Proceedings of the 2nd annual international conference on Mobile computing and networking* (New York, NY, USA, 1996), ACM, pp. 130–142.

[42] Inamura, H., Montenegro, G., Ludwig, R., Gurtov, A., and Khafizov, F. TCP over Second (2.5G) and Third (3G) Generation Wireless Networks.

[43] Kravets, R., and Krishnan, P. Power management techniques for mobile communication. In *MobiCom '98: Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking* (New York, NY, USA, 1998), ACM, pp. 157–168.

[44] M, S., P, G., D, H., and RH, K. Reducing power consumption of network interfaces in hand-held devices. In *MoMuc-3:3rd International Workshop on Mobile Multimedia Communications (MoMuc-3)* (Princeton, NJ, USA, 1996).

[45] Morris, J. Manpage of libipq: iptables userspace packet queuing library. `http://www.cs.princeton.edu/~nakao/libipq.htm`,.

[46] R. Pantos, Ed. HTTP Live Streaming. Internet-Draft draft-pantos-http-live-streaming-02, October 2009. Work in progress.

[47] Roy, S., Acquaviva, A., Acquaviva, A., Simunic, T., Simunic, T., Deolalikar, V., and Deolalikar, V. Server controlled power management for wireless portable devices, 2003.

[48] Schulzrinne, H., Rao, A., and Lanphier, R. Real time streaming protocol (rtsp), 1998.

[49] Shenoy, J., P., Radkov, and Peter. Proxy-assisted power-friendly streaming to mobile devices. vol. 5019, SPIE, pp. 177–191.

[50] Society, I. C. Amendment 8: Medium access control (mac) quality of service enhancements. Tech. rep., 3 Park Avenue,New York, NY 10016-5997, USA, September 2005.

[51] Tan, E., Guo, L., Chen, S., and Zhang, X. Psm-throttling: Minimizing energy consumption for bulk data communications in wlans. In *ICNP* (2007), pp. 123–132.

[52] Tarkoma, S. *Mobile Middleware: Architecture, Patterns and Practice.* John Willy, 2009.

[53] Wei, Y., Chandra, Surendar, Bhandarkar, and Suchendra. A statistical prediction-based scheme for energy-aware multimedia data streaming. vol. 4, pp. 2053–2057 Vol.4.

[54] Wigard, J., Madsen, N. A. H., Gutiérrez, P. A., Sepúlveda, I. L., and Mogensen, P. Packet scheduling with qos differentiation. *Wirel. Pers. Commun. 23*, 1 (2002), 147–160.

[55] Xiao, Y., Siekkinen, M., and Yläď-JÃďääski, A. Framework for energy-aware lossless compression in mobile services: the case of e-mail. In *ICC2010:To appear in the Proceedings of the IEEE International Conference on Communication.*

[56] Yan, H., Krishnan, R., Watterson, S., Lowenthal, D., Li, K., and Peterson, L. Client-centered energy and delay analysis for tcp downloads. In *Quality of Service, 2004. IWQOS 2004. Twelfth IEEE International Workshop on* (June 2004), pp. 255–264.

[57] Yang, S.-R., Yan, S.-Y., and Hung, H.-N. Modeling umts power saving with bursty packet data traffic. *IEEE Transactions on Mobile Computing 6* (2007), 1398–1409.

[58] Zorzi, M., and Rao, R. Energy constrained error control for wireless channels. In *Global Telecommunications Conference, 1996. GLOBECOM '96. 'Communications: The Key to Global Prosperity* (Nov 1996), vol. 2, pp. 1411–1416 vol.2.