2008

AdamRTP: Adaptive multi-flow real-time multimedia transport protocol for Wireless Sensor Networks

Aniss Zakaria
University of Windsor

AdamRTP
Adaptive Multi-flow Real-time Multimedia Transport Protocol for Wireless Sensor Networks

By

ANISS M. ZAKARIA

A Thesis
Submitted to the Faculty of Graduate Studies
Through the School of Computer Science
In Partial Fulfillment of the Requirements for
The Degree of Master of Science at the
University of Windsor

Windsor, Ontario, Canada

2008

Your file Votre référence
ISBN: 978-0-494-57556-7
Our file Notre référence
ISBN: 978-0-494-57556-7

AUTHOR DECLARATION OF ORIGINALITY

ABSTRACT:

Real-time multimedia applications are time sensitive and require extra resources from the network, e.g. large bandwidth and big memory. However, Wireless Sensor Networks (WSNs) suffer from limited resources such as computational, storage, and bandwidth capabilities. Therefore, sending real-time multimedia applications over WSNs can be very challenging. For this reason, we propose an Adaptive Multi-flow Real-time Multimedia Transport Protocol (AdamRTP) that has the ability to ease the process of transmitting real-time multimedia over WSNs by splitting the multimedia source stream into smaller independent flows using an MDC-aware encoder, then sending each flow to the destination using joint/disjoint path. AdamRTP uses dynamic adaptation techniques, e.g. number of flows and rate adaptation. Simulations experiments demonstrate that AdamRTP enhances the Quality of Service (QoS) of transmission. Also, we showed that in an ideal WSN, using multi-flows consumes less power than using a single flow and extends the life-time of the network.

DEDICATION

ALLAH (ALMIGHTY) FOR ALL HIS MERCY AND BLESSING

PARENTS

COMMITTEE

WIFE AND KIDS

FRIENDS

ACKNOWLEDGEMENTS

Dr. Randa El-Marakby, my supervisor, Thank you very much.

My Parents, although they are overseas, their continuous prayers encouraged and held me up. Thank you.

Manar Al-Smoudi, my beloved wife, she was patient, and helped me a lot to complete this work. Thanks

Dr. Richard Frost and Dr. A. Hussein, Thank you for your valuable assistance.

Walid, Maunzer and Colleagues, thank you for your patience and support.

My friends: Salah, Ashraf, Tarik and Raghad. Without you, this could have never been done.

Thank you everyone.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

Abbreviation	Full Name
AdamRTP	Adaptive Multi-flow Real-time Multimedia Transport Protocol for Wireless Sensor Networks
AODV	Ad hoc On-Demand Distance Vector
AOMDV	Ad-hoc On-demand Multipath Distance Vector
ARP	Address Resolution Protocol
ATM	Asynchronous Transfer Mode
B-frame	Bi-directional predictive pictures
CNAME	Canonical Name
Codec	COmpression DECompression
CSRC	Contributing source
CTS/RTS	Clear to Send / Ready to Send
DiffServ	Differentiated Services
DivX	Digital Video Express
DSCP	Differentiated Services Code Point
DSDV	Destination Sequenced Distance Vector
DSR	Dynamic Source Routing
DSSS	Direct-Sequence Spread Spectrum
ESRT	Event-to-Sink Reliable Transport in Wireless Sensor Networks
FLAC	Free Lossless Audio Codec
FTP	File Transport Protocol
HTML	HyperText Markup Language
IEEE	Institute of Electrical and Electronics Engineers
I-frame	Intra coded frames

IntServ	Integrated Services
IP	Internet Protocol
J	Joules unit of energy
MAC	Media Access Control
MANET	Mobile Ad hoc Network
MAX_TH	Maximum Threshold
MDC	Multiple Description Coding
MEMS	Micro-Electro-Mechanical Systems
MIN_TH	Minimum Threshold
MMSPEED	Multi-Path and Multi-SPEED Routing Protocol
Monarch	Mobile Networking Architectures
MP3	MPEG-1 Audio Layer 3
MPEG	Moving Picture Experts Group
MRTP	Multi-flow Real-time Transport Protocol
NAM	Network Animator
NS	Network Simulator
NTSC	National Television System Committee
O.O.	Object Oriented
OSI	Open Systems Interconnection
oTCL	Object Oriented version of Tool Command Language
P-frame	Predicted pictures
PHB	Per-Hop Behavior
PSFQ	Pump-Slowly, Fetch-Quickly
PT	Payload Type
QoS	Quality of Service
RF	Radio Frequency

RFC	Request For Comments
RMST	Reliable Multi- Segment Transport
RR	Receiver Report
RTCP	RTP Control Protocol
RTP	Real-time Transport Protocol
SAR	Sequential Assignment Routing
SDES	Session Description
SEER	Secure and Energy- Efficient multipath Routing protocol
SPEED	A Stateless Protocol for Real-Time Communication in Sensor Networks
SR	Sender Report
SSRC	synchronization source
TAG	a Tiny AGgregation Service for Ad-Hoc Sensor Networks
TCL	Tool Command Language
TCP	Transport Control Protocol
TORA	Temporally Ordered Routing Algorithm
ToS	Type of Service
UDP	User Datagram Protocol
VoIP	Voice over Internet Protocol
WLAN	Wireless Local Area Network
WMV	Windows Media Video
WSN	Wireless Sensor Network

CHAPTER I

INTRODUCTION

1.0 General

Recent advances in Internet technologies produce enormous varieties of applications that are used by an increasing number of users around the globe. Applications such as dynamic HTML, Animations, web-based video sharing (e.g. YouTube) and Video conferencing are examples of many rich implementations.

This chapter gives an overview of Multimedia Applications, their nature and requirements, followed by an introduction to real-time applications and why they are different from regular multimedia applications. A detailed overview of the Real-Time Transport Protocol (RTP) follows. A brief introduction to Quality of Services (QoS), Audio and Video Codecs, and Multiple Description Coding (MDC) is also included in this chapter. This chapter ends with our statement of research which explains the major problem that we are trying to solve, what are our goals, and an overview of our solution.

1.1 Multimedia Applications

Multimedia can be defined as a way of delivering information by using audio, video, pictures, and animations instead of just regular text. Multimedia applications such as Flash Animations, YouTube Video sharing, Video Conferencing, and RealPlayer Audio streaming are some of the applications that have been enjoyed by millions everyday around the world for many years. But what makes such applications different from regular HTML WebPages which usually contain text and some pictures only?

Internet applications can be divided into two main groups in terms of packet loss sensitivity, and delay tolerance, as shown in table 1. Network applications which are based on text (such as HTML, E-mail, and FTP) are delay tolerant, meaning they do not need to reach the destination right away, but, they have to reach it in full, without any type of loss, even if it is a single byte! On the other hand, applications that use multimedia like digital audio and video (e.g. video conferencing) are delay sensitive. However, they can accept some packet loss, because multimedia applications have the ability to reconstruct or predict the missing packets then play back without (significant) disruption.

Table 1: Requirements of network applications

	Traditional Network Applications	Real-Time Multimedia Applications
Applications	Hyper-text, E-mail, FTP, File Sharing	MM Streaming, Conferencing, VoIP
Delay	Delay-tolerant	Delay-sensitive
Packet Loss	Loss-sensitive	Loss-Tolerant (not significant loss)
Bandwidth	Small	Huge
Protocol	TCP	RTP

Multimedia applications are usually huge in size. Multimedia files get even bigger if we want a better audio or video fidelity. For example, ten seconds of raw, uncompressed NTSC video (which is the standard for television) occupies as much as 300 MB of storage space. A multimedia file size is considered to be the main problem especially when transmitting over any type of network. It gets worse when sending multimedia over wireless network because of the extra limitations of such networks, e.g. bandwidth limitations and packet loss due to signal fading.

The next subsections explain some of the techniques developed to ease and maintain an acceptable multimedia delivery over wired and wireless networks. But before that, a special type of multimedia applications that are time-sensitive is introduced. These applications are known as Real-time Multimedia applications.

1.2 Real-time Multimedia

Real-time multimedia is usually a multimedia application that has one strict rule: play on time or it will be dropped. Real-time multimedia applications such as Video Conferencing and On-Demand Video require timely delivery of packets. But, unless we have a network that can give some sort of guarantees such as IntServ or DiffServ-enabled networks (details in section 1.3), running real-time multimedia applications over regular networks may suffer from delays based on network state and congestion.

The Internet (TCP/IP model) is "best effort" by design, where the network does not provide any guarantees on timely packet delivery. This comes mainly from the inability of core routers to identify the type of packet they are routing. They do not differentiate between a packet that belongs to very low priority E-mail and a very important real-time packet that belongs to a video conference session. The network ability to classify packets is part of what is called Quality of Service (QoS), which will be discussed in the next section.

Real-time Transport Protocol (RTP) is a widely-used real-time multimedia protocol that works mainly on UDP. It assists in delivering real-time multimedia packets by using timestamps and statistical feedback information generated by a Real-Time Control Protocol (RTCP) to achieve a better QoS. RTP and RTCP are discussed in more details in chapter 2.

1.3 Quality of Service (QoS)

Quality of Service (QoS) aims at providing better networking services over current technologies such as ATM, Ethernet, and others. The main three parameters for multimedia QoS delivery on the Internet are latency (delay), jitter, and loss. Delay is the total amount of time a network spends to deliver a frame of data from source to destination. Jitter is the variation of delay between two consecutive packets during a given period of time. Loss determines the maximum packet loss that a stream of data can tolerate to provide good quality. Each parameter has been investigated thoroughly in the literature and many solutions are proposed such as forward error correction and interleaving [Kur03]. Other QoS parameters include reliability, network availability and bandwidth.

Providing hard guarantees, as in Integrated Services (IntServ), or soft guarantees, as in Differentiated Services (DiffServ), are the two main approaches to QoS in the Internet. IntServ [Bra04] establishes a virtual dedicated link between source and destination. The Resource Reservation Protocol (RSVP) is a signaling protocol that is responsible for checking the network desired bandwidth and delay requirements. IntServ provides per-flow reservations. Therefore, every node on the path needs to maintain state information about every flow. As a result, IntServ suffers from a scalability problem. DiffServ [Nic08] offers different levels of service classes. It employs Differentiated Services Code Point (DSCP–6 bits) field which exist in Type of Service (ToS) byte in the Internet Protocol (IP) header, to assign a different class to each flow. In turn, each network node treats every flow differently which is known as the per-hop behavior (PHB). Therefore, state information about every flow is not needed along the network path. A third model of QoS in the Internet is known as Adaptive Applications that adapt to network congestion based on QoS feedback, for example, sender can adjust the transmitting rate based on the level of network's congestion. Bolot [Bol94] has proposed a set of feedback mechanisms for use in adaptation of the output rate of video coders according to the state of the network, more in section 2.3.4.

Extending QoS to wireless networks presents new challenges due to radio channel characteristics, mobility management [Gar03], higher packet loss ratio than wired network, battery power constrains, and low bandwidth [Mah99]. However, most current QoS protocols can be implemented in wireless local area networks (WLAN) with some modifications because the last hop is the only wireless stage in these networks. In wireless networks, like Ad hoc Wireless Networks (MANET) or the new emerging Wireless Sensor Networks (WSNs) which are totally wireless, a new set of QoS parameters, mechanisms, and protocols are needed.

1.4 Audio and Video Codecs

As mentioned earlier, multimedia files are huge. Therefore, digital compression is essential to reduce the number of bytes needed to form a given media presentation. Compression and Decompression protocols are usually called CODECs. A CODEC usually takes advantage of the information redundancy in the data file. As a result of reducing the number of bytes used to represent a set of data, we reduce the physical space required to store it, thus reducing the bandwidth needed for transmitting the data over any type of network.

Usually, digital audio and video are data intensive. Audio files are very large, up to 10 MB for ten minutes of plain speech. Multimedia files that contain music are larger. Whereas files that contain digital video can be very huge. For example, an uncompressed video file with lots of motion can be 1 Gigabyte for 5 minutes of video. As a result, compression is required basically to make computer handling of video possible [URI03].

There are two main types of CODECs: lossy and lossless. Lossy codecs cause a slight degradation in quality. But, lossy codecs are faster than lossless coding techniques and can produce smaller file size. A lossless codec keeps the original quality, but is slower to encode and decode. Examples of some widely-used audio codecs are: FLAC (Free Lossless Audio Codec) and MP3 (MPEG-1 Audio Layer 3), while some video codecs are: WMV (Windows Media Video) and DivX (Digital Video Express).

1.5 Multiple Description Coding (MDC)

Multiple Description Coding (MDC) [Goy01] is a coding technique that divides a single media stream into n independent sub streams ($n >= 2$) referred to as descriptions. Generally, a multiple description (MD) coder generates two equal importance descriptions so that each description provides low but acceptable quality. Subsequently each description is routed over multiple joint or/and disjoint paths.

Therefore, the coder at the destination node can decode any arrived description independently, however, the quality of media improves with the number of descriptions received simultaneously. MDC provides error resiliency to media streams. Since an arbitrary subset of descriptions can be used to decode the original stream, network congestion or packet loss, will not interrupt the stream, but only cause a (temporary) loss of quality. MDC is gaining popularity because it can provide acceptable quality without the need for retransmission of lost packets (unless there is high packet loss). This can provide a better service for real-time applications where retransmissions of packets are not acceptable because it causes more delays. And it simplifies the design of network protocols where no need to employ feedback or retransmission mechanisms. MDC has many advantages such as:

- *Error resilience*: any description can be used to produce an acceptable quality media without the need to retransmit the lost packets.
- *Less bandwidth and buffer requirements*: the large source of media is divided to smaller flows that can fit in smaller buffer and require less bandwidth when transmitted over the network.
- *Reduction of network congestion*: media transmission will be shared among more nodes, each will send smaller amount of packets, therefore, reducing the packet congestion per node which leads to less congestion throughout the network.
- *Load balancing*: more intermediate nodes will participate in packet delivery causing distributed effort to deliver the data.
- *Save power*: since more intermediate nodes are sending the data, number of data transmission per node is fewer leading to overall energy saving.
- *Security*: in case of an attacker who is sniffing the network, mostly, he will get one description out of the whole data.

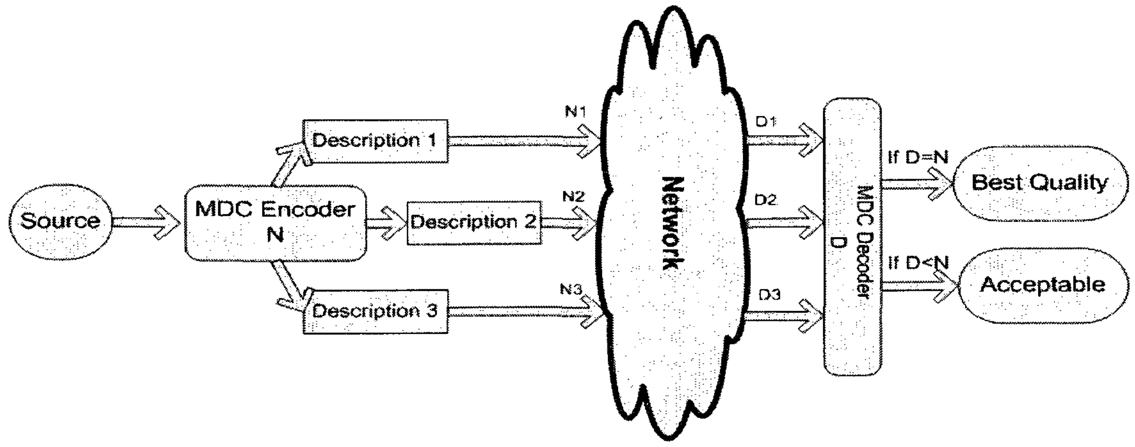


Figure 1: Multiple Descriptions Coding

Figure 1 describes an MDC encoder that divides the main media source into 3 descriptions, and sends each description to the destination using a separate route (N1, N2 and N3). If all descriptions arrive at the destination, the Decoder at the receiver will merge them into one stream again then play it back with best possible quality. But if one or two descriptions do not arrive, it will play the rest with degraded quality.

1.6 Statement of Research

In this section, we describe the problems encountered when sending multimedia over WSN. Why we do not use an existing protocol to do so? What are we trying to accomplish here? Then, we present a quick description of our proposed solution. At the end, we list our contributions.

1.6.1 Problem Description

There is an increasing demand for multimedia applications over all types of wireless networks. But with all of the limitations that WSN has naturally inherited from wireless networks, e.g. signal attenuation, sending multimedia data can add extra load greater than what a WSN is capable of, due to the fact that WSNs have limited processing power, limited energy source, and limited bandwidth. Multimedia applications require extra memory and cache. In addition, they require larger bandwidth for acceptable quality of media.

Currently, there is no real-time multimedia protocol solution available for multimedia delivery over WSNs. Such a protocol could facilitate multimedia availability and delivery to assist in some urgent situations e.g. natural disasters or war. Current solutions use mainly a single flow (path) to send multimedia data between sender and receiver. But, this can exhaust the intermediate nodes' buffer and power leading to degraded QoS.

Although, current real-time multimedia protocols such as RTP [Sch96], are designed to send real-time multimedia data, implementing such protocols directly over wireless sensor networks is not practical for the following reasons:

1. The RTP protocol is not power-aware. It does not consider the amount of power any node has in the decision of joining or leaving a session or being a part of the path that forward the traffic.
2. RTP was designed to work for IP based networks, such as the Internet, WLAN... etc.
3. RTCP, the control protocol of RTP, specifies all five types of packets (Sender Report (SR), Receiver Report (RR), SDES, BYE and APP) to be fully functional. This could be inconvenient for limited-resource networks such as WSNs.
4. RTP, in some situations, requires a large amount of memory. Koistinen [Koi00] stated that it may require up to 4 MB in certain cases.

For all these reasons a new protocol that can overcome some, if not all of these problems is necessary.

1.6.2 Objectives and Goal

Our main goal is to send Real-time Multimedia over Wireless Sensor Networks while preserving good Quality of Service and extending the lifetime of the network. However, achieving good QoS comes at the price of network resources such as bandwidth and power. To enhance the QoS without consuming more network resources, we can use some adaptation techniques as discussed in the next subsection.

In this thesis, we modify the RTP protocol to achieve good QoS over WSNs and to suite the limitations of these networks.

1.6.3 The Proposed Solution

We assume that nodes in WSN are densely deployed. A WSN has limited resources, e.g. bandwidth and power. We use multi-flows over multi-paths. The sender divides a large stream of data into smaller ones and sends each portion over a separate path (if possible). By doing this, we can relieve intermediate nodes from the burden of transmitting a large amount of data. Also multimedia traffic is distributed over more nodes. Moreover, we use an adaptation technique to adjust transmitted media dynamically according to the current network state.

By using both techniques, the proposed protocol AdamRTP helps alleviate congestion, provide better QoS, and provide a good distribution of power usage.

1.6.4 Contributions

Our contributions in this thesis are:

- Modified RTP protocol that uses MDC coder to split a stream of multimedia into two or more flows.
- Adaptive protocol that can provides dynamic adjustments e.g. number of flows and transmission rate, according to the state of the network.
- A lighter version of RTP protocol where we eliminate some of its features e.g. Sender Report (SR), and Source Identification (SDES), that can overwhelm the WSN with extra unwanted packets, but enough to provide an acceptable QoS.
- Facilitating some techniques to overcome some of WSN limitations like power constrains and low bandwidth, this can be done by distributing the load of data delivery over more sensors, and generating smaller flows out of one large stream of multimedia to be sent over a low bandwidth network.
- Offering enhanced QoS vehicle for real time multimedia applications over wireless sensor networks by employing multipath and adaptation techniques.

- Developed a complete implementation (using C++ and oTCL programming languages) of AdamRTP using Network Simulation NS-2.

1.7 Thesis Organization

This thesis is organized as follows. Chapter 2 gives an overview of the Real-time Transport Protocol (RTP). The RTP protocol is used to send multimedia data with real time characteristics. Also, in Chapter 2, we survey Wireless Sensor Networks (WSN) with discussion of related work. Chapter 3 is our main contribution. It discusses, in details, our Adaptive Multi-flow Real-time Multimedia Transport Protocol (AdamRTP) designed for WSN, starting with some definitions and design architecture, and then, it explains adaptation techniques used in AdamRTP protocol. Chapter 4 describes our simulation, starting with an introduction to the Network Simulator (NS), then, it explains the simulation environment that we used to test AdamRTP. Chapter 5 presents the conclusions and future work.

CHAPTER II

LITERATURE REVIEW

2.0 General

This chapter presents a background of the Real-time Transport Protocol (RTP), which is used to send real-time multimedia applications over best-effort networks such as the Internet, followed by a presentation about RTP Control Protocol (RTCP). Then, it surveys the new emerging networks, Wireless Sensor Networks (WSNs), explaining some of their applications and limitations. Afterwards, some related work is presented with their limitations in delivering real-time multimedia over WSNs.

2.1 RTP and RTCP protocols

The Real-time Transport Protocol (RTP) provides end-to-end delivery services for data, such as interactive digital audio/video, with real-time characteristics. RTP is a standard specified in RFC 1889 [Sch96]. More recent versions are RFC 3550 [Scf03] and RFC 3551 [Sch03].

RTP does not provide any mechanism to ensure timely delivery of packets or provide any sort of QoS guarantees. It relies on lower-layer services to do so. RTP does not guarantee delivery of packets or prevent out-of-order delivery. RTP protocol uses sequence numbers and timestamp for each packet that allows the receiver to determine the appropriate location of a packet for play out.

According to RFC 3550 [Scf03], the RTP suite (RTP/RTCP) provides the following services:

1. *Payload-type identification*: specifies the type of content being carried.
2. *Sequence numbering*: sequence number of packets.
3. *Time stamping*: used for synchronization and jitter calculations.
4. *Delivery monitoring*.

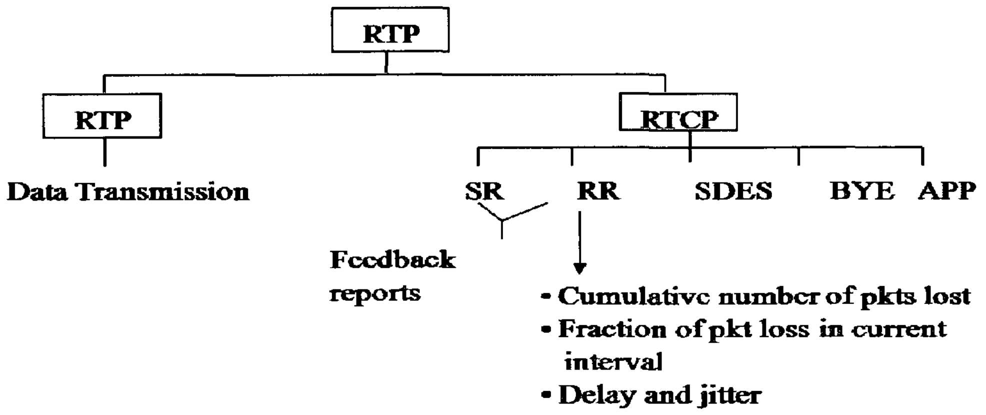


Figure 2: RTP and RTCP

As shown in Figure 2. The RTP suite consists of two closely-linked parts:

1. *The Real-time Transport Protocol (RTP):* RTP data protocol is used to carry data that has real-time characteristics.
2. *The RTP control protocol (RTCP):* used to monitor the delivery to provide quality of service and used to collect statistical information about the participants in the session.

2.1.1 The Real-time Transport Protocol (RTP)

The RTP data protocol is used for real-time data transmission. The application layer at the sender partitions the data and adds its own header information. Then it forwards the packet to the lower layers, where each layer (transport, network, and data link layers) add their own header, until received by the receiver which strips out all extra headers until the application gets the original data back. RTP header information is discussed next.

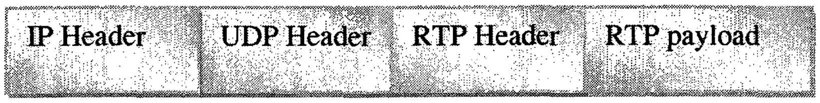


Figure 3: RTP payload and header

RTP Payload forms the actual real-time multimedia data that is being transferred. RTP header contains information related to the payload e.g. payload type, sequence number, timestamp, etc. as shown in table 2.

The RTP header has the following format:

Table 2: RTP Protocol Structure [Scf03]

2	1	1	4	1	7-bits	16-bits
V	P	X	CSRC count	M	Payload type	Sequence number
Timestamp						
Synchronization source (SSRC)						
Contributing source						

Version (V): 2 bits. Version of RTP. The newest version is 2.

Padding (P): 1 bit. If this bit is set to 1, this means that this packet contains one or more additional padding at the end.

Extension (X): 1 bit. If set, the fixed header is followed by one header extension.

CSRC count (CC): 4 bits. The number of CSRC identifiers. This number is more than one if the payload of the RTP packet contains data from several sources.

Marker (M): 1 bit. The marker is used to allow significant events such as frame boundaries to be marked in the packet stream.

Payload type (PT): 7 bits. It identifies the format of the RTP payload. E.g. PCM, MPEG2 video, etc.

Sequence number: 16 bits. Initially the sequence number is randomly set and then it will be incremented by one for each RTP data packet sent. It can be used to detect packet loss.

Timestamp: 32 bits. It represents the time that the packet has been created. Timestamp can be used for synchronization and jitter calculations.

SSRC: 32 bits. Each source will choose a unique random number to identify itself from the others.

CSRC list: 0 to 15 items, 32 bits each. Contributing sources for the payload contained in this packet. The number of identifiers is given by the CC field.

2.1.2 RTP Control Protocol (RTCP)

RTCP works in conjunction with RTP to monitor data delivery and to send periodical statistical information about each and every participant in the current session. Its main functions are:

- *QoS Monitoring:*

The main function of RTCP is to monitor data delivery and to convey identification information about the participants in the on-going session. The monitoring function if performed by employing feedback reports. RTCP uses two main reporting packets. The first called Sender Report (SR) and the other is the Receiver Report (RR). SR carries information about transmission and reception statistics from active senders. Whereas, RR carries information about reception statistics from active receivers.

- *Source Identification:*

RTCP Source DEScription (SDES) packet contains an identifier called canonical name or CNAME, something like user@host. It is used to identify a participant during the session in case of a conflict with synchronization source identifier (SSRC). SDES may contain also extra information about the participants like email, phone, location and notes.

- *Bandwidth & Transmission interval:*

RTCP control packets can overwhelm network resources if not managed properly, to prevent that, and to allow RTP to scale up to a large number of session participants, control traffic is limited to a maximum of 5 percent of the overall session traffic, out of this 5 percent, RTCP allocate 75% to receivers and 25% to senders. This limitation is enforced by regulating the rate at which RTCP packets are transmitted. Each participant sends, via multicast, control packets to others, so they can keep track of the total number of participants and use this number to verify the rate at which to send RTCP packets.

RTCP Packet types

RTCP identifies five packet types to carry a variety of information as follows:

1. *Sender Report (SR):* Issued by the data sender. It includes the synchronization source identifier (SSRC) of the creator of SR packet. Also SR contains the total number of packets and byte sent.
2. *Receiver Report (RR):* Issued by the receiver. It contains information about:
    - Fraction lost (8 bits): fraction of packets lost since issuing previous RR. It is calculated as the number of packets lost during any given interval divided by the number of packets expected during the same interval.
    - Cumulative number of packets lost (24 bits): total number of packets lost since the beginning of the session. It is calculated as the number of packets expected minus the number of packets actually received since the beginning of the transmission.
    - The extended highest sequence number received (32 bits): A random number generated by the sender at the beginning of the session, then incremented by one for each generated RTP packet.
    - Interarrival jitter (32 bits): an estimate of the statistical variance of RTP data packets interarrival time (more in subsection 3.3.1).
    - Timestamp (32 bits): can be used to estimate the round-trip delay between a sender and the receiver.
3. *Source Description (SDES):* contains the canonical name (CNAME) that identifies the source, more information in subsection 2.1.2.
4. *Goodbye (BYE):* When a source is no longer active or decides to leave the session, it sends an RTCP BYE packet. The BYE notice can include the reason for leaving the session.
5. *Application-Defined (APP):* provides a mechanism for applications to define and send new application definitions (not defined in the standard).

2.2 Wireless Sensor Networks (WSNs)

The rapid advancement in Micro-Electro-Mechanical Systems (MEMS) and wireless communication technologies have enabled the integration of sensing, actuation, processing, and wireless communication capabilities into tiny sensor devices. This sub-section presents and surveys WSNs and their limitations.

2.2.1 Introduction

Wireless Sensor Networks (WSNs) are composed of many tiny, low-cost, low-power and scattered devices called sensor nodes. Each node integrates a processor, memory, transceiver, and power source in one small device that has the ability to observe, process, and send data about observed phenomenon to its neighboring nodes destined to a central processing unit sometimes referred to as a sink. A sensor node should have the ability to process as much information locally as possible instead of just disseminating raw data to save energy, because radio frequency (RF) communication is the key energy consumer [Est01]. Usually the main source of energy in a sensor node is a battery. Therefore, the lifetime for any node depends on the lifetime of the battery itself. For these reasons, many Media Access Control (MAC) protocols have been proposed to turn radio communication on and off periodically instead of just listening to the channel all the time e.g. SMAC [YeW02]. Energy conservation is one of the main obstacles to any proposed protocol in sensor networks, while maintaining high QoS measurements is the main goal in traditional networks [Aky02].

Sensor nodes are densely and randomly deployed. This can provide better accuracy and more energy saving since nodes can use short-range communication. However, if not managed properly, data redundancy and collisions may occur. For example, in a forest, hundreds of nodes can be programmed to inform a central sink if the temperature exceeds 45° C. When the event occurs, many nodes may disseminate at once the same information to the sink, resulting in data redundancy and implosion at the sink. To solve this problem, while maintaining a degree of reliability, data aggregation techniques can be used to combine and summarize the data coming from different sources into one data stream [Mad02].

2.2.2 Applications

Wireless Sensor Networks enjoy a large variety of applications. Because the sensor network is wireless, it can be deployed almost anywhere and anytime. One sensor network cannot be designed to fit all type of applications. They are application-specific networks, which are capable of achieving specific set of operations like weather condition monitoring. For these reasons, Sensor Networks applications are categorized into five main categories [Aky05]: environmental applications, health, home, military and other commercial applications.

1. *Environmental:* weather (temperature), natural disaster (flood), and natural species (animal and plants)
2. *Health:* patients (monitor condition 24/7), doctor (locating), and drugs
3. *Home:* home automation and security
4. *Military:* track enemy movements
5. *Commercial:* factories (track inventories), traffic (traffic control and safety)

2.2.3 Communication Protocols of a WSN

WSNs, like any other network architectures, share almost all OSI layers, but with slight differences. In this section, we discuss the two most important layers that concern us, application and network layers.

2.2.3.1 Application Layer:

QoS may be interpreted in two different perspectives [Chen04]. The first perspective defines QoS as quality perceived by the user or application. Another perspective defines QoS with respect to the network and how the network is able to provide QoS to users or applications. We can redefine the first type as a set of rules or parameters set by a user or an application to get the desired service from the network. For example, the user can ask the network to send the data in pairs to achieve higher reliability.

In a user/application perspective, many parameters can be defined by the user to achieve some QoS in WSNs:

1. *Fidelity:* A user can instruct the network to send their queries back to the sink in pairs, or do not accept any event that has been seen by a number of nodes only.
2. *Update (Refresh):* Sensors should send data to the sink every period of time, even if there are no events.
3. *Mode:* A user/Application defines how the sink will interact with events. In general, four data delivery models are defined: event-driven, query-driven, continuous, and hybrid [Til02].

From a network perspective, providing QoS to an application or a user defines new QoS parameters:

1. *Query processing:* It is the ability of a WSN to perform in-network processing instead of sending raw data to the sink. For example, a sink may send a query "What is the highest temperature in the forest?" In response to this query, each sensor node sends back the temperature to the sink which in turn calculates the highest temperature or let the nodes in the network find it out themselves and then send the result only. This can be accomplished with the help of aggregation mechanisms. A Tiny AGgregation Service (TAG) [Mad02] is one approach to combine related data sent by the nodes into one compact record based on a set of aggregation values specified by queries.

2. *Coverage:* High coverage is a key factor to a robust sensor network and is considered one of the QoS measures [Meg01]. It discusses the ability to provide the largest area of coverage possible using the lowest number of sensor nodes. Generally, nodes are deployed either randomly or based on a predefined map. Random deployment usually suffers from lack of coverage. This can be solved by allocating some extra nodes manually during network runtime.

   Having good coverage algorithms can save power and improve sensor network connectivity. In an area that is covered by multiple sensors we can turn some sensors off (save power) or instruct one or two sensors only to sense the environment (less redundant data). k-UC and k-NC are two algorithms proposed to determine how well each sensing area is covered [Hua03]. A related problem to coverage is *exposure* that measures the ability of a given network to observe an object over a period of time [MKQ01].

3. *RTP (multimedia streaming over WSNs):* The Real-time Transport Protocol (RTP) [Scf03] provides end-to-end delivery service for real-time audio or video. RTP adds timing and sequence information to every packet allowing the reassembly of packets so that it reproduces real-time audio or video.

   The Real-time Control Protocol (RTCP) is responsible to maintain, control, and provide QoS feedback in an RTP session. In addition, both senders and receivers send reports to each other to synchronize the delivery of packets.

   Implementing RTP as is in a WSN can suffer from some problems. First, it requires high caching capabilities to save state information at end sensors. Second, scalability can be another problem as a WSN may consist of hundreds or even thousands of nodes and it has scarce bandwidth. Sending "high quality" audio or video streams is usually not required in WSNs. However, some modifications to RTP are essential before implementing it in a WSN. More discussions about RTP limitations are in section 2.4.1.

2.2.3.2 Network Layer:

The network layer deals mainly with determining the route from source to destination and managing traffic problems. Generally, the network layer is responsible for end-to-end packet delivery, whereas the data link layer is responsible for node-to-node (hop-by-hop) packet delivery. Routing in sensor networks is different from routing in traditional network because each sensor does not necessarily have a global unique ID thus selecting the next hop node becomes harder. Moreover, in a WSN, each node acts as a sensing and routing node at the same time. Routing protocols in a WSN can be categorized as in [Akk05], as follow:

1. *Data-Centric:* Data is disseminated among sensors without the need for global unique ID. It depends on the naming of desired data.
2. *Hierarchical:* Sensors are controlled by a sensor (cluster-head) to aggregate data. A cluster-head is either a special (more powerful) node or an elected sensor in each cluster.
3. *Location-based:* These protocols are location-aware usually by utilizing a GPS. The ability to find the location makes it easier to route data to a single and specific region instead of broadcasting traffic to all regions.
4. *QoS based:* Protocols that ensure some QoS requirements such as minimum cost path: in terms of energy, low throughput, or delay.

Our concern here is QoS based routing protocols. Relatively few protocols have been proposed. A recent survey [Akk05] identifies three protocols only as QoS-aware, as follows:

1. *Sequential Assignment Routing (SAR)* [Soh00]: is a QoS routing protocol that builds a table of paths between the source and destination. SAR creates multiple trees each rooted from 1-hop neighbor node of the sink, it tries to avoid nodes with very low QoS and energy reserve. The table is used for multi-path, energy efficiency, and fault tolerance. But SAR suffers from the overhead of maintaining the tables and states of the network inside each sensor. When WSN becomes very large, there is a scalability problem.
2. *SPEED* [HeT03]: This is a real-time communication protocol for a WSN that provides soft real-time end-to-end guarantees. It uses location-based mechanisms to find the route to the sink. By employing location awareness, SPEED can calculate distance. Thus, it can find out the time it takes to deliver packets to a destination prior to admission (end-to end delay). In addition, it can handle congestion. SPEED maintains a table for immediate neighbors only. It does not maintain a routing table or per-destination state. Therefore, its memory requirements are minimal. It does

not provide any energy-awareness mechanisms other than spreading traffic uniformly through the entire network.

3. *Energy-Aware QoS Routing Protocol* [Akk03]: This protocol is concerned mainly with power. It finds a least cost and energy efficient path that meets certain end-to-end delay requirements during the connection. Additionally, a class-based queuing model is employed to support both best effort and real-time traffic simultaneously. The protocol captures the nodes' energy reserve, transmission energy, error rate, and other communication parameters to calculate the communication link cost. However, it is based on the concept of end-to-end applications, which may not be appropriate in some WSNs and it is too complex [Che04].

2.2.4 QoS in WSNs

Regular wired networks send data between nodes mainly without the knowledge of the nature of the carried data (*data transparency*). They use the end-to-end communication model mainly. Therefore, parameters like delay, bandwidth, jitter, and loss can provide acceptable QoS if managed properly. However, in WSN, these parameters are not fully applicable because sensor nodes communicate mostly using non-end-to-end model. That is to say, each node communicates only with its neighboring nodes (hop-by-hop model). This means that no connection needs to be established between source and destination at the beginning of the transmission process. Another problem arises from the fact that intermediate sensor nodes have the ability to generate data in addition to routing it. Along with the most challenging problem which is energy consumption, all these factors generate new QoS parameters like coverage, exposure, energy cost, and network life time.

The problem of coverage can happen when no sensor node observes and informs the sink about an event. This may happen because of noisy channels, deployment location, or network management [Che04]. Exposure is related to coverage that provides measures of how an object can be observed by a sensor over a period of time. Energy cost defines the process of finding the best route to destination according to energy conservation, while network lifetime is the total time of a WSN until it is unable to satisfy user's needs.

Implementing the two QoS models of the Internet on a WSN is not practical. IntServ [Bra04] depends mainly on reserving the bandwidth between source and destination while saving state information on each intermediate node (more discussion about Intserv can be found in 1.3). This is impractical in a WSN for three main reasons. The first reason is the complexity to achieve such service. The second reason is the limited memory capability in each sensor node that can't save per-flow state information. The third reason is because the route usually is not known between the source and destination at the beginning of the transmission process. Implementing DiffServ [Nic08] in a WSN faces another problem beside complexity. That is to say, the core ideas behind DiffServ is queuing and prioritizing packets based on a service priority level. Queuing requires large memory which normally a sensor node does not have.

Reliability, as a QoS parameter in WSNs, refers to the ability to correctly sense events and to successfully deliver data from sensors to sink and vice versa. Reliability protocols are divided into two groups: Event-to-Sink and Sink-to-sensor.

Event-to-Sink transport carries information usually about observed phenomena. In most cases it might be very critical data which needs to be reliably communicated to the sink. Several protocols have been proposed such as Reliable Multi-Segment Transport (RMST) [Sta03] and Event-to-Sink Reliable Transport (ESRT) [San03]. Sink-to-Sensor transport usually carries queries or update control information. A protocol such as Pump Slowly Fetch Quickly (PSFQ) [Wan02] is proposed for reliable transfer of tasks and reprogramming the WSN nodes.

What makes QoS in WSN different?

Sensor nodes are small in size. Therefore, each node is equipped with limited batteries, a processor, and a transceiver that leads to restricted power source, slower processing capabilities, and constrained communication power. These limitations have caused new challenges that are discussed briefly as follows:

1. *Power:* This is considered the most critical limitation. Therefore, almost every protocols proposed considers the energy problem. The main power consumer as discussed earlier is communications, so high compression and local data processing should be done on each node before dissemination. Achieving a better service (QoS) is always at the price of energy [You04].
2. *Bandwidth:* it is one of QoS parameters. The lack of bandwidth presents more difficulties in achieving QoS in WSNs. Using data compression and utilizing different bandwidth capabilities based on the nature of the stream are two approaches to overcome the scarce of bandwidth [Kim05].
3. *Memory size:* The limitation of memory (cache) size affects most proposals to enhance WSN networking capabilities. In some cases, local memory is not enough to load the whole operating system, and to implement extra QoS measures.
4. *Standardization:* The lack of standardization in WSNs makes it hard to implement a QoS solution. ZigBee[1] may be considered to be a first attempt.
5. *Lifetime:* The nature of a WSN lifetime is limited because of the fact that most nodes operate on nonchargeable power sources like batteries. Another reason is the ease of node damage.
6. *Density:* It leads to data redundancy. Although it may help to achieve reliability but it may add also overhead and consume power to aggregate traffic to the sink. In addition, it may add some sort of latency and complexity to QoS design [Che04].
7. *Application diversity:* A WSN is consider to be application specific rather than general purpose, carries hardware and software needed only for a certain application. The vast number of applications in a WSN offers different QoS requirements.

---

[1] ZigBee is a low-cost, low-power, wireless mesh networking standard (http://www.zigbee.org)

2.2.5 WSNs Limitations

WSNs inherit almost all challenges from regular Wireless Local Area Networks (WLAN) and Mobile Ad hoc Networks (MANET) in addition to the following [Est01] [Aky02]:

1. Each sensor node suffers from a very limited power source, not like PDAs or laptops which are usually recharged.
2. A sensor network topology faces frequent changes due to external forces like animals, vehicles, or humans. It faces also internal reasons like power or software failure.
3. In most cases, a sensor node does not have a global ID, which makes most of the current network protocols inapplicable to a WSN.
4. Sensor networks operate mainly without any human intervention and they should be self-configurable.
5. Sensor nodes are densely deployed. This increases redundancy and collisions.
6. Sensors have the ability to know the nature of information they are carrying, unlike traditional networks where intermediate nodes only forward packets of data.
7. Sensor nodes normally use the broadcast communication model, while traditional networks use point-to-point communication.

For all of the above reasons, implementing QoS in Sensor Networks differs from regular QoS implementations in other types of networks.

2.3 Adaptive Applications

Adaptive applications are applications that can adapt to current network state (e.g. congestion and channel error) by changing some of its behaviors, e.g. encoding rate and/or media quality.

Adaptations of multimedia applications can be implemented at a number of layers of the network stack. For example, at the physical layer, an adaptive power control is used to equalize the received power for all nodes in the wireless network. This problem is caused by the attenuations that a signal transmitted through radio channels may face, such as path loss due to the long distance, and shadowing caused by a shielding obstacle.

At the data link layer, error control protocols are used to recover from wireless channel error. Two main error recovery mechanisms are used: Automatic Repeat Request (ARQ), and Forward Error Correction (FEC). ARQ used to retransmit the lost packets while FEC transmits some redundant data with the original packet. In most cases, the use of hybrid ARQ/FEC protocol improves the performance of error control schemes for wireless links. An adaptive combination of both protocols helps the network to send adjustable amount of redundant information according to channel error rate. However, in fixed coding scheme, a high waste of bandwidth may occur during normal behavior of the network since the frequent redundant information is not necessary due to the low bit error rate of the channel.

At the network layer, a routing protocol can adapt either to network's needs or user's needs. That is to say, if a network is suffering from limited energy source, in this case an adaptive power-aware routing protocol is necessary to avoid draining nodes with limited power. Moreover, if a network is suffering from high error rate, so according to the level of the error rate, a routing protocol can duplicate sending messages to ensure reliability. At the user level, a user can choose between different levels of reliabilities. Accordingly, the routing protocol can send more than one copy of the packet from source to destination to ensure a sort of reliable service.

At the transport layer, current transport protocols, e.g. TCP, do not provide smooth congestion control for multimedia applications. Thus, the need for an adaptive congestion control that suits both the networking and video coding communities is necessary. The need for an adaptive congestion control that understands and differentiates between different applications and user needs and adjusts sending rate accordingly to deliver maximum congestion-free throughput. As an example, the Reception Control Protocol (ReCP) [Kyu05] which is a TCP clone in its general behavior. It uses an adaptive congestion control algorithm that monitors the wireless random loss rate and delay dynamically, then adjusts its congestion control adaptation parameters to compensate the loss rate and delay components introduced by the wireless link.

At the application layer, the application can adapt to changes in the network state by several techniques such as media encoding, compression, media quality, or rate shaping. Our main concern in this survey is about adaptive applications. Therefore, a detailed discussion about each technique is to follow.

2.3.1 Encoding

Multimedia encoding is a process of digitizing a sample of any media presentation such as audio, video, and graphics. As discussed earlier in section 1.4, an encoder usually compresses the media file. The media file is decompressed by a decoder. Encoders have the ability to produce different quality samples out of one media presentation.

Adaptation to network condition changes can be accomplished by a number of techniques at the compression level (encoder) such as layered encoding [Cha03]. In layered encoding, the encoder divides the media into several layers. The base layer carries the most important data beside some critical timing information, while the higher layers add progressive level of quality to the original media file. Therefore, the receiver can get an acceptable quality when receiving only the base layer. But quality improves with the reception of more packets that belong to higher layers. In adaptive multi-layered media encoding, the encoder at the sender node can add or drop layers based on network feedback.

2.3.2 Compression

Multimedia files are generally huge. That's why we need to compress these files to reduce storage and transmission requirements. There are several aspects of the compression techniques that can be used for adaptations. For example a technique that can be used for adaptation is using different quantization levels and encoding rate. Quantization is a digital signal processing method used to convert discrete signal into digital signal. The encoding rate is the rate at which the original media file is sampled in order to create the digital file. A higher sampling rate produces higher quality media files. For example, in a compact disc (CD), an analog recording is converted to a digital signal sampled at 44,100 Hz (44 KHz) and quantized with 16-bits (2 bytes) of data per sample.

Another technique that can be used for adaptation is by using different levels of Discrete Cosine Transform (DCT) [Ahm74] methods. DCT helps separate the media file into parts (or layers) of different importance (priority). The base layer carries the most important media information and additional layers improve the quality. In the event of congestion, lower priority layers can be dropped to reduce the sending rate.

Compression techniques are categorized into two groups: Lossly and lossless compression. Lossy compression methods produce a much smaller compressed file than lossless methods but with less media quality. A user or encoder can decide which compression method to use and how much data loss to introduce and make a trade-off between file size and image quality.

2.3.3 Media Quality

Multimedia files consist usually of audio or video or a combination of both. In a media file that consists of both audio and video media, an encoder encodes usually each media separately. Humans are more sensitive to audio than video, therefore, choosing a lossless compression method to encode audio while using a lossy compression for video can help produce a better quality multimedia files while preserving a smaller file size. The quality of media file is represented by a number of bits that are used to sample a unit of time. An encoder that uses a lower bit rate produces a lower media quality file and therefore creates lower file size. However, a higher bit rate produces higher quality and bigger file size, e.g. "An MP3 file that is created using the mid-range bit rate setting of 128 kbit/s will result in a file that is typically about 1/10th the size of the CD file created from the original audio source" [2].

[2] http://en.wikipedia.org/wiki/Mp3

Therefore, adaptation of quality can be done by reducing the bit rate when the network suffers from congestion resulting in lower quality until the network state goes back to normal where it increases the bit rate to produce a better quality.

2.3.4 Rate Shaping

Encoders can encode media file using Constant Bit Rate (CBR) and Variable Bit Rate (VBR) [Lak98]. In CBR, the encoder uses the same speed rate to encode the whole media file. This can be simpler but may waste some space when encoding simple sections (e.g. audio pause segments). In VBR, the encoder uses different encoding speed rate for different segments of the same media file. The advantage of VBR is that it produces a better quality-to-space ratio when compared to a CBR-based file of the same size.

Rate shaping techniques try to adjust the rate of traffic generated by the encoder according to the current network conditions. A rate control mechanism for video in the Internet [Bol94] was developed to provide a simple feedback control mechanism that can be used to prevent video sources from swamping the resources of the Internet. The rate shaping is achieved by adjusting some parameters of the video encoder, specifically the refresh rate, the quantizer, and movement detection threshold.

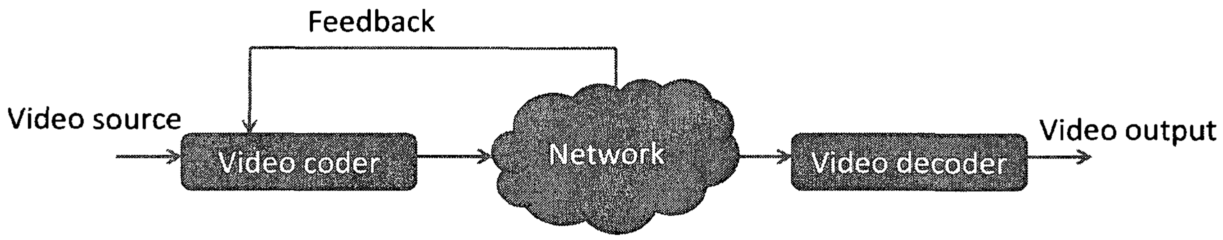


Figure 4: Network feedback control

Receivers send feedback information (e.g. packet loss rate) to the sender's coder periodically, see figure 4. The coder computes the average loss rate to estimate the network load and selects a maximal output rate value according to it.

2.4 Related work and comparison

Sending real-time multimedia over wireless sensor networks is a new topic that has not yet been fully explored. So, the process of finding a closely related work to AdamRTP was hard. Many approaches are available for sending real-time multimedia over wired or wireless network e.g. Ad hoc network. These approaches are concerned mainly on number of methods such as modifying the coder, either by using adaptation techniques [Ati01] or by using Multi-Layer and Multiple Description Coding techniques [Wan05], or a combination of both [Dur08]. Other methods are by using a real-time transport or routing protocol e.g. RTP [Scf03] and SPEED [HeT03]. Further proposals take the Cross-layer approach e.g. FireFly [Man07]. In this section we will discuss in details some approaches than uses a combination of these approaches to facilitate sending real-time multimedia over wireless network and if possible WSN. One protocol which is somewhat close to AdamRTP is called (MRTP) which stands for Multi-flow Real-time Transport Protocol for Ad Hoc Networks [Mao06] (subsection 2.4.1), it uses a combination of an MDC and transport protocol techniques to operate over ad-hoc networks. Another protocol is SPEED (subsection 2.4.2) which is a routing protocol that helps with real-time data delivery over WSN. Other protocol called Adaptive Multiple Description Coding for Internet Video (subsection 2.4.3) that uses adaptation and MDC coder to help in sending video over the Internet.

2.4.1 MRTP: Multi-flow Real-time Transport Protocol for Ad Hoc Networks

MRTP [Mao06] is designed to allow real-time application the ability to send real-time data over Ad hoc networks. It uses an MDC encoder and decoder to split a stream of data into several flows then sends each flow on a separate path. MRTP is a transport protocol that could be implemented in the user space of the host's operating system. "Given multiple paths maintained by an underlying multipath routing protocol, MRTP and its companion control protocol, the Multi-flow Real-time Transport Control Protocol (MRTCP), provide essential support for multiple path real-time transport. This includes session and flow management, data partitioning, traffic dispersion, timestamping, sequence numbering, and Quality of Service (QoS) reporting" [Mao06].

MRTP protocol is a session-oriented protocol, this means a session needs to be established prior to any data transmission. A three-way handshaking is initiated from the source node (either sender or receiver) using *HelloSession* and *ACKHelloSession* messages, this gives both parties an opportunity to agree on some parameters such as number of flow to be used. During the session, some flows may be unavailable, due to node failure or severe congestions, in this case, the receiver sends a *DeleteFlow* message to the sender, the sender in return delete the affected flow and redistribute the assigned remaining packets over other active flows. When a new path is found, the routing protocol informs MRTP protocol about this new path, accordingly, a new flow can be added to the session by initiating an *AddFlow* message. MRTP generate a periodical QoS reports using Sender Report (SR) and Receiver Report (RR) that carries both per-flow and per-session statistics. When packets of different flows arrive at the destination, the receiver stores these packets in a buffer for reordering (according to sequence number and timestamp found in the header) and jitter compensation.

AdamRTP looks similar to the MRTP. However, AdamRTP is designed for real-time multimedia applications while MRTP is for any real-time application. AdamRTP eliminates many of the RTP functionalities to simplify the process of delivery, while MRTP uses the whole RTP suite besides adding some extra features such as flow splitting and flow managements. Moreover, MRTP is designed for Ad hoc networks, not Wireless Sensor Networks, and, as we discussed in section 2.2.5, a WSN has extra new requirements that any new proposed protocol should consider. Furthermore, MRTP is a session-oriented protocol, although we believe in multimedia streaming this is not necessary. Finally, MRTP does not consider power consumption at all.

2.4.2 SPEED: Stateless Protocol for Real-Time Communication in Sensor Networks

SPEED [HeT03] is a location-aware and QoS routing protocol that is designed for real-time communication in WSNs. SPEED provides soft-realtime end-to-end guarantees. The protocol requires each node to maintain information about its neighbors and uses geographic forwarding to find paths. The core component of SPEED protocol is The Stateless Non-deterministic Geographic Forwarding algorithm

(SNGF) which is responsible for collecting information about neighboring nodes, such as location, fast and slow neighbors, and delay estimation. Based on this information, SPEED can decide the path between the source and destination. Delay estimation is calculated by counting the time elapsed when a node receive an ACK packet in response to a previous transmitted packet. Accordingly, SNGF selects the next hop node based on this delay value. SPEED performs better in terms of end-to-end delay when compared to other Ad hoc routing protocols such as Dynamic Source Routing (DSR) and Ad-hoc On-Demand Vector routing (AODV). Moreover, SPEED claims that by selecting the shortest path between the source and destination in terms of number of hops, it can reduce the energy consumed for transmission. SPEED provides adaptations only at the MAC layer by locally dropping or buffering packets at congested areas. SPEED is not a multi-path aware protocol. It does not have timestamping and packet sequencing capabilities. A newer version of SPEED that supports multipath is called MM-SPEED [Fel06].

2.4.3 Adaptive Multiple Description Coding for Internet Video

Adaptive Multiple Description Coding for Internet Video [Lot03] is an adaptive scheme that uses Multiple Description Coding (MDC) techniques to send video over the Internet. The main feature of this scheme is to alternate between two simple MDC schemes depending on the network state and the underlying visual content. The video content can change from low to moderate or high motion scene. Therefore, it is important to design an MDC scheme that intelligently adjusts the coding scheme according to visual contents in addition to network conditions.

The two simple MDC schemes used are temporal sub-sampling and spatial sub-sampling. The concept of sub-sampling is often used in image or video compression where information is sampled at lower sampling frequency to reduce the amount of data to be stored [Won06]. Temporal sub-sampling used in this scheme splits the original video sequence into even and odd frames with equal quality. On the other hand, spatial sub-sampling constructs even and odd lines of the video image with different quality. Then, each stream could be decoded independently using two separate codecs or serially using a single codec. After transmitting these descriptions over packet lossy networks, some descriptions or part of it may be lost. Therefore, a state recovery algorithm that has access to both descriptions at the receiver tries to reconstruct the best quality video out of the received descriptions.

This scheme is mainly designed for video transmission over the Internet. Furthermore, it does not consider real-time multimedia transmission or address the special requirement of WSNs.

2.5 Summary

In this chapter, we surveyed the most important topics that we think are related to our proposed protocol. Starting with a detailed study about the Real-time Transport Protocol (RTP) which provides a set of mechanisms for delivering multimedia data with real-time characteristics such as online conferencing. Moreover, we discussed Wireless Sensor Networks (WSNs) with details about their architectures, applications, and limitations, emphasizing on techniques that have been developed to achieve QoS over such limited networks. Also, a study about Adaptive Applications is presented discussing some of the limitations that current proposed solutions are facing when delivering real-time multimedia applications over WSNs. Finally, some related works are presented such as MRTP, SPEED, and Adaptive Multiple Description Coding for Internet Video with comparison against AdamRTP

CHAPTER III

Adaptive Multi-flow Real-time Multimedia Transport Protocol for WSNs: AdamRTP

3.0 General

This chapter describes our main contribution. We start with an overview, then provide detailed description of AdamRTP and its design architecture. Then, the adaptation techniques are presented. At the end is our summary.

3.1 Overview

Consider an area that is covered by many sensors that are equipped with audio and video capabilities. Generally, when an event occurs, the sensor that sensed the event generates a media representation of the event and divides the stream into many flows using an MDC encoder. Then, it forwards these flows to the sink over multiple paths. The sink merges the media, then decodes the media by a decoder and makes it available to the user directly or probably over the Internet, as shown in figure 5.

To accomplish that, we assume that each sensor has a fair size of internal cache which can be used to store continuous multimedia (audio/video) files of the surrounding environment. The sensors do not send continuous streams of data to the sink unless an event triggers the source/sender sensor to do so. This prolongs the life of each sensor node. For example, a triggered event could be the hearing of the word "HELP" or the detection of a waiving sign. The sender sensor is equipped with a coder that is capable of generating multiple flows out of one stream called Multiple Description Coding (MDC).

A flow consists of data packets transferred from the sender to the receiver using a certain path, while a session consists of one or more flow which carry a single real-time multimedia stream between sender and receiver and necessary control packets. A path/route is the physical route that a flow uses to reach the destination.

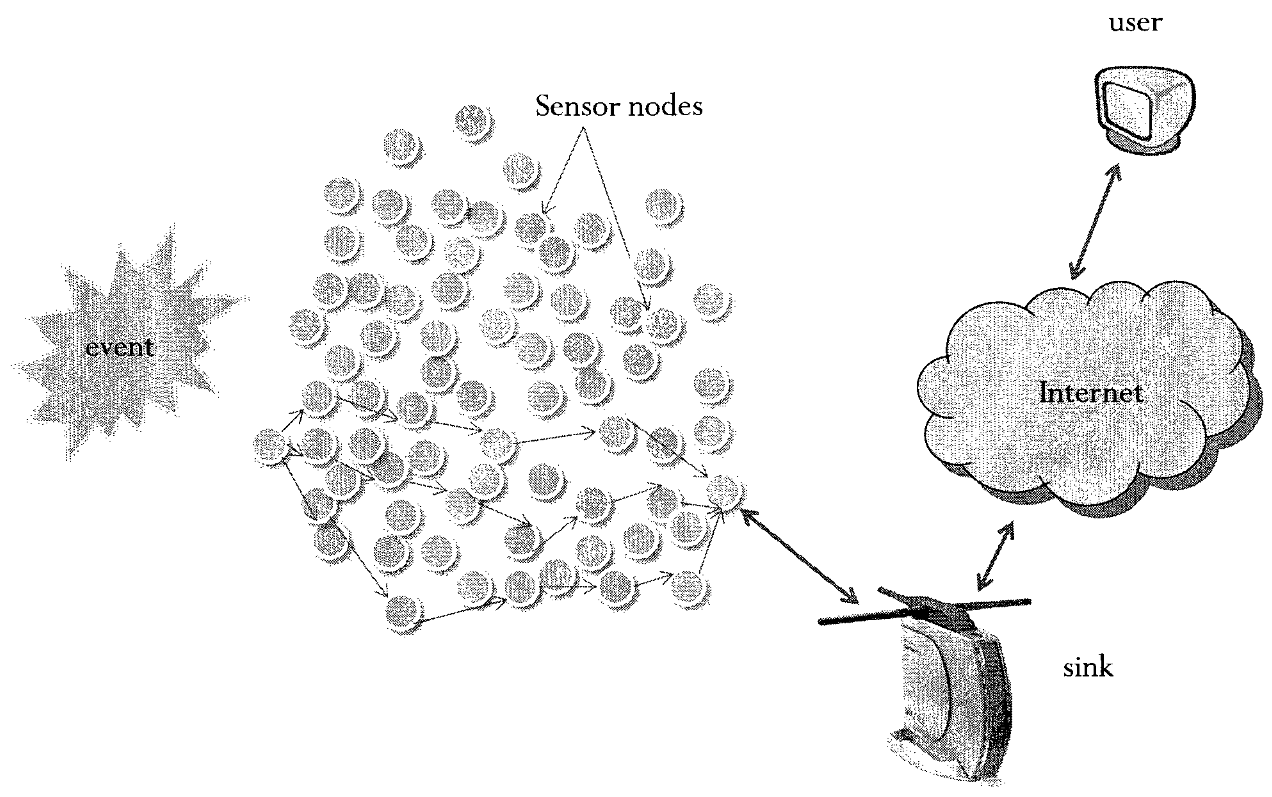


Figure 5: AdamRTP, sender generates media presentation of an event and sends it to user using multi-flows

Each flow routes to the sink using a different path (if possible). We can always start with one or two flows, then increase the number of flows (up to **n** flows) until something wrong happens, e.g. severe packet loss or failure of an intermediate sensor on the path used. AdamRTP may function over any multipath routing protocol such as the Highly-Resilient, Energy-Efficient Multipath Routing in WSNs [Gan01] or SEER [Nas07].

The receiver generates a Receiver Report (RR) packet every time unit e.g. every one second, combining statistical information about all previously received flows in the current interval. Interval is the time between issuing the previous RR from the sink and the current RR. Based on the RR feedback, the sender tries either to increase the number of flows if network state is uncongested or minimizes the number of flows or lowers its transmission rate if there is congestion until the network state goes back to normal. Detailed discussion about network adaptation is given in section 3.4.

3.2 AdamRTP Description

AdamRTP employs no connection establishment mechanism between the sender and the receiver at the beginning of the session. The sender simply sends the real-time multimedia data to the sink as soon it detects something. In general, the multimedia stream goes through the following 5 stages: (AdamRTP implements stages 1, 4, and 5)

1) *Encoding and partitioning:* sensors are equipped with a small and moderate quality camera. For example, a camera which can capture video of resolution 640x480 pixels at 30 frames-per-second can be considered acceptable, adequate, and small to fit in one small sensor node. Moreover, this camera is enough to fulfill the job of delivering a good quality of video. The stream of video generated by the camera passes into a coder that compresses it and divides it into **n** flows using an MDC aware coder. AdamRTP has the ability to compress and partition the video stream using some partitioning methodologies such as thinning and striping [Bus02]. Once partitioned, each portion is assigned a flow number and a sequence number, then the flow is sent to the sink by a separate route (if possible).

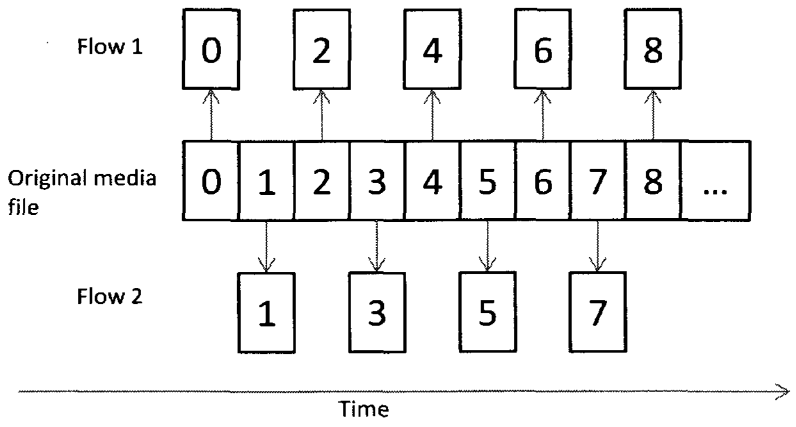


Figure 6: Thinning a media stream into two flows

In thinning way of partitioning, given in figure 6, a media file is divided into equal-size blocks, each block contains a number of multimedia frames (audio or video). Then, each block is assigned to a path in a round-robin scheme. Each flow contains less number of flow than the original media file, this relieves the nodes on each flow from exhausting its bandwidth, power, and computational resources. Additionally, the pauses found between blocks on each flow, can give intermediate nodes more time to receive, queue, then transmit packets that belong to certain block of media.

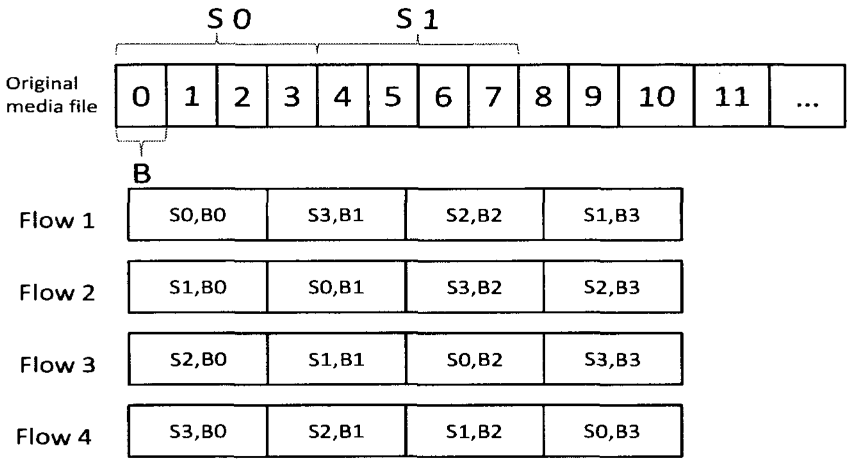


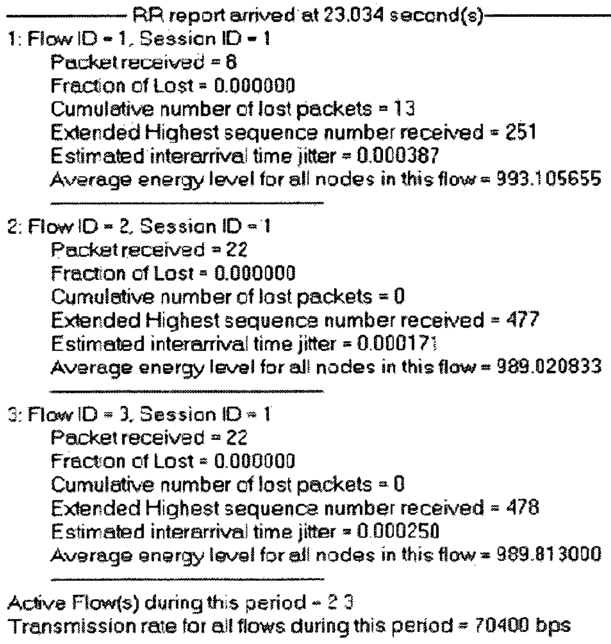
Figure 7: Stripping media file into 4 flows

Whereas stripping divides the main media files into equal size blocks (B), then according to the number of the flow to be used (S), the coder fills each flow with blocks (B) from different substreams in interleaved fashion. Interleaving is a process of rearranging blocks of a stream in non-adjacent manner to increase performance and mitigate the effect of packet loss. However, it increases latency because blocks that belong to a certain multimedia frame will not arrive at the same time [Kur03]. In Figure 7, an original media stream is divided into a number of blocks of size (B). Since the number of flows to use is 4, we arrange every 4 blocks into *stripped substream* (S). The coder then chooses first block from the first 4 stripped substream (S0, S1, S2, and S4) and sends each block to the destination on a separate flow.

2) *Path Selection and assignment:* usually finding the best route to the sink is the job of the routing protocol. AdamRTP assigns only its own header information (such as flow id, session id, sequence number, and a time stamp), then passes this information to the lower layer.

3) *Packet routing:* It is preferable to find disjoint paths back to the sink, so we can distribute the stream of data into flows all over the WSN, thus saving more power and facing less congestion. But, this requires a huge amount of overhead trying to find totally disjoint paths from source to destination resulting in a waste of bandwidth and memory.

4) *QoS feedback:* AdamRTP generates periodical statistical QoS reports summarizing the quality received by all previous flows since the last QoS report. However, unlike RTP, where the sender and receiver must generate these reports, in AdamRTP only the receiver generates the report. This will free the network for more data. The receiver report (RR) is almost similar to RR in RTP, where it has information about the fraction of packets lost in the current interval, accumulation of packets lost since the beginning of the session, highest sequence number received, the interarrival jitter, and some other values as shown in figure 8. Based on these RRs, the sender acts accordingly. That is to say, it may increase or decrease quality of media and number of flows.



```
------------ RR report arrived at 23.034 second(s)------------
1: Flow ID = 1, Session ID = 1
    Packet received = 8
    Fraction of Lost = 0.000000
    Cumulative number of lost packets = 13
    Extended Highest sequence number received = 251
    Estimated interarrival time jitter = 0.000387
    Average energy level for all nodes in this flow = 993.105655
    ----------------------------------------
2: Flow ID = 2, Session ID = 1
    Packet received = 22
    Fraction of Lost = 0.000000
    Cumulative number of lost packets = 0
    Extended Highest sequence number received = 477
    Estimated interarrival time jitter = 0.000171
    Average energy level for all nodes in this flow = 989.020833
    ----------------------------------------
3: Flow ID = 3, Session ID = 1
    Packet received = 22
    Fraction of Lost = 0.000000
    Cumulative number of lost packets = 0
    Extended Highest sequence number received = 478
    Estimated interarrival time jitter = 0.000250
    Average energy level for all nodes in this flow = 989.813000
    ----------------------------------------
Active Flow(s) during this period - 2 3
Transmission rate for all flows during this period = 70400 bps
```

Figure 8: AdamRTP Receiver Report (RR) including QoS summary about 3-flows

5) *Reassembling at the sink:* Once the sink receives a packet, it extracts the session and flow id from its header and queues it until a complete set of packets received up to a certain time threshold called the playout delay. Then, the sink decodes the received stream back to its original multimedia file. We assume the sink is more powerful and has more computational capabilities in terms of processor, memory, and storage. Thus, the decoder is more complex than a coder in normal sensor nodes. Because we can have more than one flow in one media session, each may take a different route, this can generate two types of jitter. The first type is jitter between two consecutive packets in the same flow and the second type of jitter is across the flows. This is caused when different flows arrive at different times. So, we need to have a large buffer to store the received packets and then wait some time before we play the media back.

3.3 AdamRTP Architecture

Unlike network architectures in almost all other type of networks, e.g. the Internet, the OSI model of layers in WSNs is mixed up. Recently, a cross-layer design for WSNs [Aky06], where two or more layers are merged has been proposed. The supporters of the cross-layering think that sensor nodes are designed usually for specific tasks, as we explained earlier in section 2.2. Because sensors are small in size, they think it will be more convenient to unify layers to simplify the design. However, it is not practical to implement the whole seven OSI layers in a single sensor that is only tasked to sense the temperature!

For these reasons, we somehow tried to implement AdamRTP to operate over the network layer directly. Although it is not considered to be a pure transport layer or an application layer, it tries to unify the tasks of both layers (See figure 9).

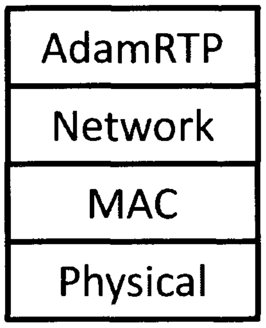


Figure 9 AdamRTP Layer

Normally, a transport layer is responsible for four main tasks:

1. Segmentation/Reassembly
2. Error Recovery (using checksum)
3. Reliable transport service
4. And in case of Internet layer model, TCP provides congestion control

AdamRTP is a connectionless, non-reliable protocol that fragments, then reassembles packets. Large datagrams are broken into small ones to fit the frame size of AdamRTP, which is 200 bytes. When these datagrams arrive at the receiver, the receiver reassembles them and merges back to the original packet format. AdamRTP can (optionally) provide a simple error recovery mechanism using checksum but we did not implement it. By using adaptation techniques, AdamRTP can avoid congested or dead intermediate sensors and it can adjust the sending rate which can be considered as a simple form of congestion control.

3.3.1 AdamRTP Packet Format

AdamRTP adds its own header information to the "payload data" to be sent. Then, network protocol adds in turn its own header information, as shown in figure 10.

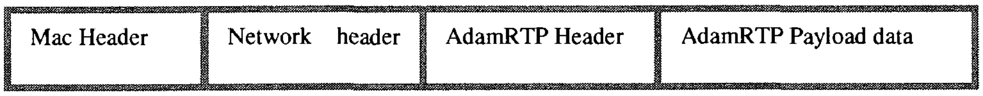


Figure 10 AdamRTP Payload and Header

AdamRTP Header Format:

0 1	2 3 4 5 6 7 8	9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24	25 26 27 28 29 30 31
V	PT	Session ID	Flow ID
Node's counter			
Energy level			
Extended Sequence Number			
Timestamp			

Figure 11: AdamRTP header format

V: AdamRTP version. 2 bits
AdamRTP version number. It is set to 1.

PT: Payload Type. 7 bits
Specifies the format of the AdamRTP payload. This field contains the standard audio/video encoding used, e.g. JPEG, H261.

Session ID. 16 bits
A random number specifying the session used to carry all the multimedia flows belonging to the steams in the session for a specific event. This can be useful if we have more than one source of AdamRTP traffic.

Flow ID. 7 bits
A number specifying the flow number which is used to carry the payload from source to destination. e.g. if the current number of the flows in the session is 3, then the Flow ID can be either 1, 2, or 3.

Node's Counter. 32 bits
Sender initializes this field with 0. Then, each node forwarding an AdamRTP packet along the path between the sender and the receiver increments this by 1. When the packet arrives at the sink, it will divide the total energy level stored in the next field by the node's counter resulting in a value that estimates the average energy level of all nodes participating in delivering this packet.

Energy level. 32 bits
Each node along the path between the source and the destination adds its own energy level to the value of this field.

Extended Sequence Number. 32 bits
The extended sequence number is incremented by 1 for each n flows. For example, if we are using 3 flows in the session, then the extended sequence number is incremented by one after the transmission of every 3 packets, each belonging to a different flow. More details in section 4.3. Sequence number can be used to arrange out-of-order packets arrived at the destination.

Timestamp. 32 bits
Used for synchronization and calculation of delayed packets and jitter.

RR Header Format:

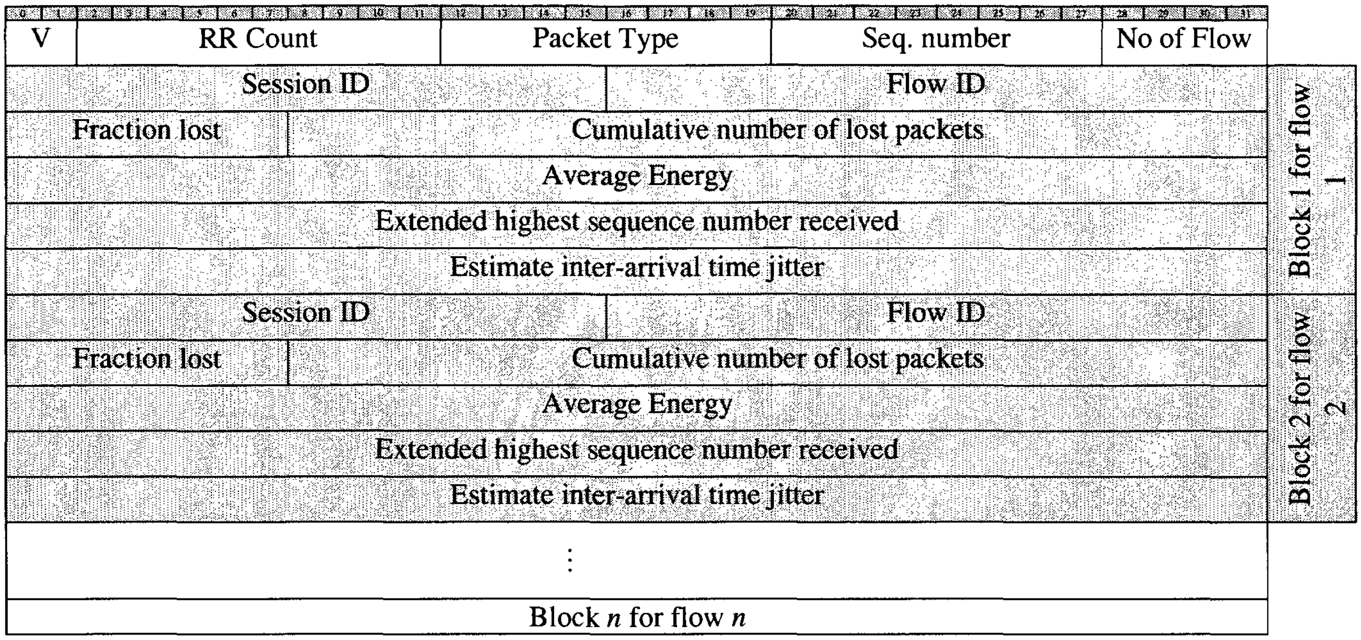

V
RR Count
Packet Type
Seq. number
No of Flow
Session ID
Flow ID
Fraction lost
Cumulative number of lost packets
Average Energy
Extended highest sequence number received
Estimate inter-arrival time jitter
Block 1 for flow 1
Session ID
Flow ID
Fraction lost
Cumulative number of lost packets
Average Energy
Extended highest sequence number received
Estimate inter-arrival time jitter
Block 2 for flow 2
⋮
Block *n* for flow *n*

Figure 12: RR header format

V: AdamRTP version. 2 bits
AdamRTP version number, set to 1.

RR Count. 10 bits
Sequential Number for number of RRs that have been sent.

Packet Type. 8 bits
A field specifies the type of reporting packet. So far, we have one type as we are using RR only.

Sequence Number. 8 bits
Specifies a general sequence number for the whole session.

Number of Flows. 4 bits
How many flows are active and used during the current session. This can help determine number of blocks required.

Session ID. 16 bits
A number specifying the session used to carry all the multimedia flows belonging to one stream for a specific event.

Flow ID. 16 bits
A number specifying the Flow number which is used to carry the payload from source to destination, e.g. if we use 3 flows, Flow ID will be 0, 1, or 2.

Fraction Lost. 8 bits
A percentage of packets lost since last RR issued by the receiver.

$$fraction\ lost = \frac{number\ of\ packets\ lost}{number\ of\ packets\ expected}$$

Where:

$$number\ of\ packet\ lost = number\ of\ packet\ expected - numer\ of\ packet\ received$$

$number\ of\ packets\ expected = (specific\ hightest\ sequence\ number\ received - base\_seq\_no + 1 - previous\ value\ of\ expected\ value$ in previous RR

Cumulative number of lost packets. 24 bits

Total number of packets lost since the beginning of the session.

Average Energy. 32 bits

The average energy level of all nodes in a particular flow delivered on a path calculated by the sink.

Extended highest sequence number received. 32 bits

This field indicates the highest sequence number received using a specific flow.

Estimate inter-arrival time jitter. 32 bits

Jitter estimation for a specific flow.

$$jitter = |\,(rec_i - rec_{i-1}) - (sent_i - sent_{i-1})\,|$$

where i denotes current packet

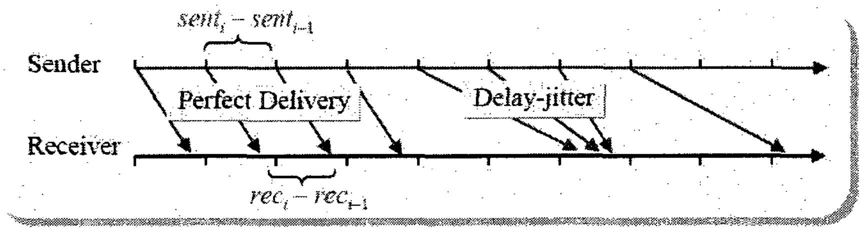


Figure 13: Jitter calculations

Figure 13 illustrates how jitter can be occurs. In a perfect network, packets arrive at the receiver in a timely fashion, where the difference between the arrival time of previous packet and the arrival time of current packet is equal between all delivered packets. But in congested network, some packet may arrive late than others causing a jitter. Jitter can vary from time to time according to receiving packet time.

3.4 Adaptations

The other main building block of the AdamRTP protocol is the adaptation mechanism. AdamRTP provides a number of adaptation mechanisms to enhance QoS and to extend the life time of the WSN. Statistical QoS feedback information helps the protocol acquire information about the state of the network, then allows it to act accordingly on time using adaptation mechanisms.

AdamRTP provides three different adaptation mechanisms:

1. *Number of flows:* the sender adjusts the number of flows dynamically to suite the current WSN state. When the receivers detect that a certain active flow is delivering late real-time multimedia packets (which we consider lost), or no packets at all, the receiver informs the sender about this incident in the next RR. When the sender receives the RR, it deletes this flow. The sender adds another flow only to maintain a minimum number of flows. Moreover, AdamRTP adds a new flow even if there is no packets loss at all. This happens if the network is uncongested. That is to say, it adds a new flow if the number of flows is less than the maximum allowed. More details about changing number of flow according to network state are to follow.

2. *Rate adjustment:* adjusting the sending rate can ease the delivery of packets when the network is facing congestion. If the network state is congested and deleting an affected flow does not help alleviating congestion, AdamRTP will decrease the sending rate i.e. decrease media quality. By decreasing the media quality, we decrease the number of packets that represent the number of frames used to play that media. However, if the network reports no significant congestion (according to the value of fraction of lost that each individual flow is reporting) for 3 consecutive RRs, AdamRTP increases sending rate up to a threshold.

   In the NS simulator, RTP's default transmission interval rate is 3.75 ms. This means every 3.75 ms a new RTP packet will be issued by the sender. Since RTP's packet size, in default NS implementation, is 210 bytes, this means RTP default transmission rate is 448 kbps. Experiments showed that this transmission rate is not achievable on multi-hop wireless networks. The best we can achieve using our grid topology was around 65Kbps. Therefore, we set the default transmission rate to this value and define two rate thresholds, the upper-bound rate threshold MAX_RATE set to a certain value, e.g. 70 kbps, and the lower-bound rate threshold MIN_RATE, set to another static value, e.g. 60 kbps. Therefore, according to the network state, we increase or decrease the transmission rate by a small value, e.g. 0.1 until reaching either threshold or network state changes. More details can be found in the pseudo code below (see page 46).

3. *Power:* AdamRTP checks continuously for cumulative power used by each path separately. If power is draining fast from a certain path used by a flow, AdamRTP instructs the sender about this incident. Sender sensor, in turn, instructs routing protocol to use different path if possible. Each node along the path adds its own energy level to a field in the header and increment another field that act as a node's counter. When the sink receives this packet it divides the total number of energy level over number of node's count to get the average of power level for all nodes that participate in the process of forwarding the packet from the source to the destination. The receiver sends the average power value to the sender in the next RR for each flow.

   It is important to mention that AdamRTP does not adapt the power level, but it reacts upon the level of power of all nodes along the path and accordingly instruct routing protocol to change the path for a newer one to distribute power usage over more sensors before exhausting the power of all the nodes.

Our adaptation algorithm, at the sender, checks continuously for RRs that have been issued periodically from the receiver, and examine each flow separately. It calculates the fraction of packets lost for each flow found in the RR received. If the loss is less than the minimum threshold (MIN_TH), it will mark the network as "GOOD". When receiver receives two RRs stating that network state is good for every individual flow, then rate adjustment adaptation algorithm will increment the sending rate by a small value, e.g. 0.1, until reaching a maximum sending rate threshold (MAX_RATE). On the other hand, if the loss is greater than minimum but less than maximum loss threshold (MAX_TH), it will consider the network as "ACCEPTABLE". Checking for packet loss between minimum and maximum threshold allows the sender not to change the number of flows right away (to prevent oscillations). However, if the network loss is greater than the maximum loss threshold (MAX_TH), the network will be marked as "BAD" and actions should be taken. The number of flows is decreased until reaching a single flow. But, before deleting a flow, AdamRTP checks for current active number of flows, if number of flows is greater than one, it will delete the affected flow right away. But, if the number of flow is equal to one, AdamRTP cannot delete the affected flow right away otherwise we will end up with no active flow at all. Therefore, AdamRTP adds a new flow and instructs the routing protocol to find a new path for the new flow. Once successful, AdamRTP deletes the affected flow. As well, when network is marked as "BAD", the sending rate is decreased until reaching a minimum sending rate threshold (MIN_RATE).

The following adaptation algorithm shows a pseudo code for the "number of flows" and "rate adjustment" adaptation techniques.

```
1.  numFlows = MIN_FLOWS
2.  FOREVER
3.  Wait for new RR summarizing quality from different flows in the current interval
4.                                            /*note that RR contains statistics about each flow*/
5.  Calculate PktLoss for all active flows  individually /*calc. average of fraction of packet loss for each flow*/
6.  /* Derive current network state */
7. For all current flows
8.   IF (flow[numFlows].PktLoss < MIN_TH) {            /*if PktLoss for each flow is less than minimum threshold*/
9.      network_state = GOOD
10.   } ELSE {
11.    IF (flow[numFlows].PktLoss < MAX_TH)            /*if PktLoss > MIN_TH, but < MAX_TH*/
12.       network_state= ACCEPT                       /* network state is acceptable*/
13.   } ELSE {
14.       network_state = BAD    }
15.  IF (network_state = GOOD) {
16.     IF (previous_network_state = GOOD) {               /* increase transmission rate if Network was good
17.       IF (transmission_rate < MAX_RATE) {               for more than specific number of RR intervals and
18.           Transmission_rate += 0.1 }                    current transmission rate is less than MAX_RATE */
19.       }
20.   IF (numFlows < MAX_FLOWS) {
21.      numFlows++
22.      Instruct routing protocol to make a new path for the new flow
23.  }
24.  ELSE
25.   IF (network_state = BAD) {                          /*  if network state is BAD, decrease transmission rate
26.       IF (transmission_rate >MIN_RATE) {                  delete the affected flow */
27.              Transmission_rate -= 0.1
28.       }
29.      IF (numFlows > MIN_FLOWS) {
30.         Delete flow with largest pktLoss and free its path
31.         numFlows--      /*delete most suffering flow and redistribute remaining packets on remaining flows*/
32.      } ELSE {
33.       /* We have to add a new flow first so we do not go below the value of MIN_FLOW /*
34.        Add a new flow; numFlows++
35.       Delete affected flow; numFlows--
36.       }
37.  }                               /* end of network_state = BAD */
38. END LOOP
```

3.5 Summary

AdamRTP is a real-time multimedia transport protocol for wireless sensor networks. It uses two techniques to achieve good performance and to enhance quality of service. The first technique is the multi-flow technique that works by dividing one stream of multimedia into two or more flows and then sending each flow on a separate path/route to the sink, if possible. The Multi-flows technique takes advantage of the fact that WSNs are usually densely deployed creating many routes between any source and destination. The second technique is the adaptation technique to alleviate congestion. AdamRTP provides three different adaptation techniques: number of flows, rate adjustments, and power adaptation.

AdamRTP provides the following advantages:

1. Distributes the load of multimedia traffic over many sensor nodes instead of a specific set of nodes.
2. Helps increase the lifetime of the whole WSN.
3. Reduces packet loss (congestion).
4. Provides set of adaptation techniques to reduce congestion and save energy.
5. Using multi-flow (with multiple paths) adds additional benefits such as:
    - Error resiliency: if one flow does not arrive, we can construct the media from the other received flow(s).
    - Requires less bandwidth and buffer due to the fact that we divide the large stream into smaller flows.

CHAPTER IV

SIMULATIONS

4.0 General

Simulations were conducted with the well-known, open source Network Simulator (NS) [NS2]. This chapter starts with an introduction of NS and lists some of its limitations. This is followed by a description of our simulation environment and configuration. Afterwards, different simulation experiments are described. We concluded with a summary.

4.1 Network Simulator (NS) overview

NS [NS2] is a discrete event simulator designed for networking research. NS provides essential support for simulation of TCP, routing, and multicast protocols over wired and wireless (local and satellite) networks. NS is a free network simulation application that can be downloaded from the web [ISIUSC]. NS is compatible with many operating systems e.g. MS Windows, and UNIX. NS has an open architecture that allows users to edit existing or to add new functionality.

NS is an object-oriented network simulator that uses two programming languages, C++ and oTCL. C++ is used for programming the core functionalities of NS, like nodes, protocols, links, queues, etc. The other language is oTCL (TCL "Tool Command Language" + O.O. "Object Oriented" = oTCL) which is considered to be an easy and flexible scripting language. This separation has created a more powerful and faster core functionally, while enjoying flexible topology script writing.

The functionalities of a TCL script are to:

1. Define a network topology, e.g. create two nodes and links them with 10 kbps bi-directional link.
2. Define a traffic pattern, e.g. when a certain FTP session starts and stops.

3. Collect statistics and output the results of the simulation. Results are usually written to files, including files for NAM (Network AniMator) [NS2], which is a free animation tool that comes with NS.

NS is a basic non-graphical tool. The output NS provides is very detailed. It gives per packet information at MAC, agent, and router levels.

4.1.1 Wireless in NS

Wireless implementation in NS was first developed by the CMU/Monarch (Mobile Networking Architecture) [MNRCH]. The wireless model consists basically of the mobile node at the core, with additional supporting features that allow simulations of multi-hop ad-hoc networks, wireless LANs etc. A mobile node is thus the fundamental node object with added functionalities of wireless and mobile node such as ability to move within a given topology, ability to receive and transmit signals to and from a wireless channel, etc. The difference between wired and wireless node is that a mobile node is not connected by means of links to other nodes or mobile nodes but instead it uses Radio Frequency (RF) for communication.

The Wireless model in NS has many features such as:

1. Complete implementation of the IEEE 802.11 MAC protocol.
2. Complete implementation of the Address Resolution Protocol (ARP).
3. Implementations of the following multi-hop ad hoc network routing protocols:
    - Dynamic Source Routing (DSR).
    - Destination Sequenced Distance Vector (DSDV).
    - Temporally Ordered Routing Algorithm (TORA).
    - Ad hoc On-demand Distance Vector (AODV).

4. Wireless network interface modeling the Lucent WaveLAN DSSS radio.

5. Modeling of signal attenuation, collision, and capture.

6. Two Ray Ground Reflection radio propagation model.

The Radio Propagation Model is considered to be one of the main factors that evaluates the accuracy of the simulator. CMU's NS radio models are implemented better than other wireless simulators but still fail to represent many aspects of realistic radio networks, including hills, obstacles, link asymmetries, and unpredictable fading [Kot03].

4.1.2 WSN in NS

Up till now, the current implementation of NS does not support the simulation of wireless sensor networks. It requires an extension to enable the special features of WSNs. One of the most used extensions is MannaSim [ManSim]. MannaSim's goal is to develop a detailed simulation framework, which can accurately model different sensor nodes and applications while providing a versatile testbed for algorithms and protocols.

NS with current extensions does not scale well for wireless sensor networks. A recent study [Xue07] showed that simulating a WSN with around 80 nodes using AODV routing protocol can result in huge number of packet drop. Some possible improvements are needed to work well in wireless sensor network environments [Xue07].

4.1.3 RTP in NS

RTP implementations in NS are poor. Officially, the RTP model is barely updated since early releases. The implementation of RTP in NS consists of three classes, Agent/RTP, Agent/RTCP and Session/RTP. Agent/RTP is responsible for sending RTP data among participants' nodes. The data sending rate by default is 3.75ms. Agent/RTCP is used to deliver the RTCP feedback and SDES packets every interval of time (default interval value is 1000ms). The Session/RTP class is used to control the session, compute most of the statistical data about the ongoing session, and insert this data in the header of RTCP packets.

There are hardly any current contributions for RTP under NS. So far, we could find only two improvement codes. The first was developed by El-Marakby [ElM01] using ns-2.1. Since then, the standard header information of packet.h has changed drastically. The current NS version is ns-2.33. El-Marakby's enhanced RTP code has provided functionalities like RTCP report packet structure, RTCP SRs and RRs. Moreover, El-Marakby corrected major errors with original NS implementations, i.e. creation of independent RTCP header instead of using RTP header for RTCP communications, correction of severe memory leakage, correction related to RTCP bandwidth fraction and many more. The other RTP contributed improvement was developed by Research Unit 6 (RU6) which is part of the R&D Department of the Computer Technology Institute [ResU6]. They have extended the functionality of the RTP and RTCP code in NS2 to include:

1. Most of its feedback functions described in RFC 3550.
2. TCP friendly behavior, i.e. the transmitted flow will consume no more bandwidth than a TCP connection.

One more limitation in the current RTP implementation in NS is that it does not support multiple RTP streams running in one network node. Another point, for RTP in NS to be fully functional, it has to be a member of a multicast group. However, AdamRTP requires a unicast connection between one sender and one receiver only.

4.1.4 Summary

NS was developed to assist researchers, students, and industry professionals to simulate and test an existing or newly developed network protocol. Currently, NS is considered to be the most popular network simulator, for many reasons, such as:

1. It supports wide range of topologies and protocols.
2. It is open source.
3. Compared to other simulators, it has the largest and most active community.
4. It is easily extensible by users who are familiar with the software.

On the other hand, NS has some disadvantages such as:

1. It is very hard and time consuming to learn.
2. Hard to troubleshoot.
3. Not enough detailed documentation.
4. Hard to get support.
5. Versions compatibility problem.
6. Some parts are left without update for a long time such as RTP implementations.

In conclusion, it is fair to mention other simulators that have the ability to simulate WSN and/or RTP, such as OPNET[3], OMNET++[4], and SensorSim[5].

4.2 Simulation Environment and Configuration

We used the Network Simulator (NS-2) version ns-2.31. NS was installed on two types of system. First system is Ubuntu Linux version 7.10 installed on Sun dual AMD Optron 64-bits workstation. The other system is a Sun Fire V880 with Solaris version 5.10. This system is equipped with 8 processors and 16 GB of memory size.

AdamRTP requires a multipath routing protocol. NS-2 does not have any built-in wireless multipath routing protocol. Therefore, we found one external extension called "Ad hoc On-demand Multipath Distance Vector routing" (AOMDV) [Yua05]. The NS implementation of AOMDV was

[3] http://www.opnet.com/
[4] http://www.omnetpp.org/
[5] http://nesl.ee.ucla.edu/projects/sensorsim/

developed at the Computer Science Department at Stony Brook University. Unfortunately, this code is fairly old and does not compile well with recent NS releases. So, we had to create a number of predefined paths between source and destination instead of developing a whole new multipath routing protocol that works with the new releases of NS. Instead, we used a protocol called DumpAgent. This protocol uses a broadcast technique to get an answer from a desired destination. For example, if many nodes are close to each other and we want two nodes to communicate, the source will send a broadcast Address Resolution Protocol (APR) message to all of its neighbors. Only the desired destination will answer back with another APR message before packet transmission begins. If the destination is not within the source's range of communication, the source node will get no answer. So, actually this is not a real routing protocol because we have to instruct each node individually where to send the next packet.

This limitation in NS has affected our ability to conduct a real simulation of multipath technique and therefore ruled out obtaining highly accurate results. But, we believe the percentage of error is very small because we used DumpAgent to simulate all different scenarios and therefore every situation has been treated the same way.

NS-2 supports many wireless Medium Access Control (MAC) protocols such as IEEE 802.11, SimpleMAC and sMAC. We used IEEE 802.11 as it is the dominant MAC protocol for wireless networks. By default, the feature of CTS/RTS[6] is disabled, as research has shown that exchanging RTS/CTS packets will degrade the performance when the hidden terminal[7] is not an issue because most data traffic goes into one direction, as well, it will consume more energy.

[6] RTS/CTS (Request to Send / Clear to Send) is a mechanism used by the 802.11 wireless networking protocol prior of exchanging any message, mainly developed to solve the hidden terminal problem.

[7] The hidden node problem occurs when two nodes (a and b) want to communicate with another node (c) but both nodes (a and b) are not visible to each other.

Another building block for the wireless model in NS is the Radio Propagation Model that is used to predict the received signal power of the packet. That is to say, each wireless node has a receiving threshold at its physical layer. If the signal power of the received packet is below the receiving threshold, it will be marked as error and dropped by the MAC layer. NS-2 implements three different propagation models: free space model, two-ray ground model, and the shadowing model [YeW]. In our simulation, we used the TwoRayGround model which considers both the direct path (clear line-of-sight path) and a ground reflection path. This model gives more accurate prediction at a long distance than the free space model.

4.3 Implementations

Creating an AdamRTP implementation using NS-2 consists mainly of two parts: creating the C++ core functionality of the protocol and designing the network topology using oTCL. We would like to take a top-down approach in explaining how AdamRTP works, starting from creating the network topology, how many sensors in our network, which node initiates the stream (source), and which node is the destination. Also, the oTCL script defines the different paths that data and AdamRTCP packets use.

We created a 10 by 10 grid of sensor nodes (total 100 nodes) as shown in figure 14. Each node is about 50 meters apart. We changed the values of RXThresh_ (Receiving Threshold) and Pt_ (Power Transmit) which control the transmitting power. Because, by default, these values were set as if the nodes are 250 meters apart. While creating the nodes, we assigned both AdamRTP and AdamRTCP agents to every created node in the topology. By doing this, each sensor node can act as a stream generator and a packet forwarder when necessary.

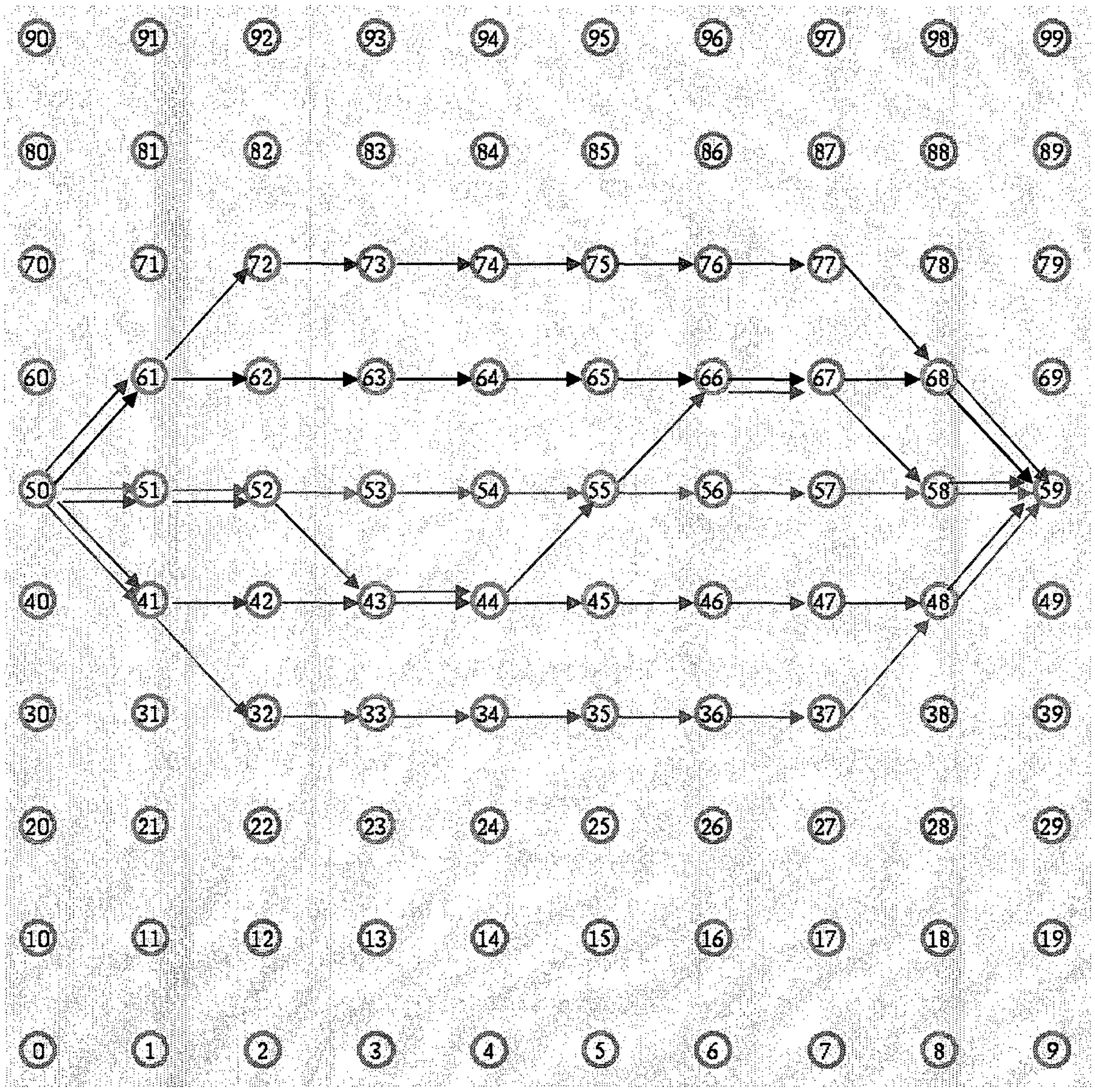


Figure 14: Network of 10 x 10 sensor node. The arrows show 6 different paths

Node 50 is the source of traffic, while node 59 is the destination (sink). We created 6 equal (in terms of number of hops) predefined paths between these two nodes, as follows:

- Path 1: 50-61-62-63-64-65-66-67-68-59
- Path 2: 50-51-52-53-54-55-56-57-58-59
- Path 3: 50-41-42-43-44-45-46-47-48-59
- Path 4: 50-61-72-83-84-85-86-77-68-59

- Path 5: 50-41-32-23-24-25-26-37-48-59
- Path 6: 50-51-52-43-44-55-66-67-58-59

When we instruct our oTCL script to start sending packets, the script is responsible for defining which path to use (as we are trying to emulate a routing protocol), it will use path 1 for single flow and use path 1 and path 2 for 2 flows and so on.

The session id, flow id, and sequence number are generated by the oTCL script depending on the number of flows used and send all these values to the AdamRTP C++ program by bind linkage method that NS-2 is equipped with. For example, if we are using 3 flows, the sequence number will be incremented by one every 3 generated packets, as shown in figure 15.

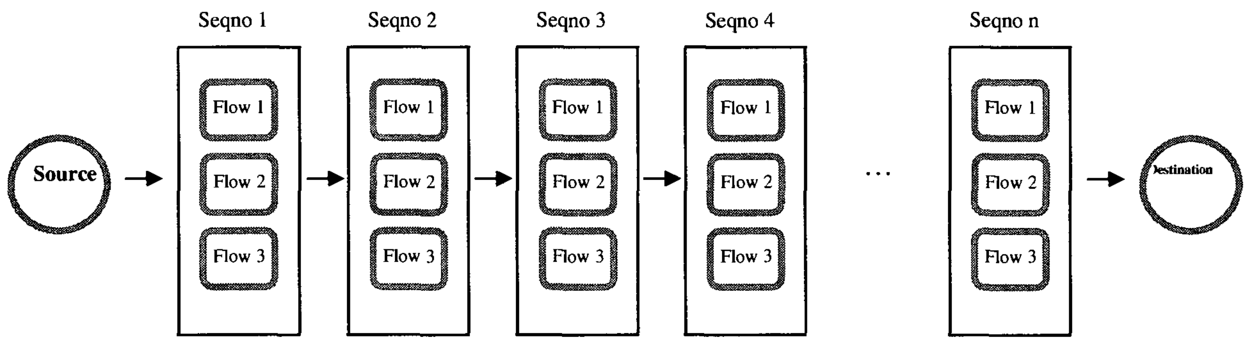


Figure 15: Flow ID and Sequence Number assignment

Throughout the path to the destination, each intermediate node adds its own energy level value and computes the average power value. This value will be calculated for each flow individually and when the receiver sends its own RR packet, it adds the average power value in the RR and sends it back to the source.

The C++ implementation part of AdamRTP consists of three major files: adamrtp.h, adamrtp.cc, and adamrtcp.cc. The header file adamrtp.h contains the C++ declarations, classes, and variables used for AdamRTP implementations. We defined 4 major parts here: 1) AdamRTP and AdamRTCP header structure, 2) AdamRTPStats which holds most of the statistical information about the session, 3)

AdamRTPAgent which is responsible for all AdamRTP functionalities, and last, 4) AdamRTCPAgent which is responsible for AdamRTCP tasks. Both agents are implemented on the sender node and the sink (destination).

The file adamrtp.cc manages all packet transmission details and computes statistical information about data packet transmission for each individual flow. Once the oTCL script instructs adamrtp.cc to send a packet using a specific path, the AdamRTP agent at the sender, stores the sessionid, flowid, sequence number, and the timestamp in the adamrtp_header and then sends it to the destination. Once received, the AdamRTP agent, at the destination, extracts sessionid, flowid, sequence number, and the timestamp from the header and starts computing statistical information such as:

1. Number of packets received: a counter that is incremented by one for every received packet. Note that a different counter is required for every different flow.
2. Extended Highest Sequence Number Received (ehsnr): stores the sequence number received for each separate flow.
3. Transit time: is the time required by one packet transmitted from the source to the destination. At the destination, it is calculated as follows:

$$transit\ time = recv.time - sent.time$$

sent.time is extracted from timestamp field found in the received AdamRTP packet.

recv.time indicates the current time the packet is received.

4. Jitter: is the variation of packet delay between current transit time and previous transit time for each individual flow. It is calculated at the destination as follows:

$$instantaneous\ jitter\ (d) = |\ transit\ time_{current} - \ transit\ time_{previous}|$$

$$jitter_{current} = jitter_{prev} + (\frac{d - jitter_{prev}}{16})$$

These values (in previous bullets 1, 2, 3, and 4) are stored in the AdamRTPStats class and are updated every time a packet is received. AdamRTPStats stores information such as: flow id (flowid), extended heighted sequence number (ehsr), packet sent (pktsend), packet received (pktrecv), transit time (transit), and jitter.

After an interval of time (by default 1000ms), the AdamRTCP agent, implemented in the sink, starts building the RR packet and sends it to the sender. Building the RR packet requires AdamRTCP to access the information that is stored in the AdamRTPStats class to compute the following:

---

1) received_interval = AdamRTPstats.pktrecv – previous_pkt_recv *\/* Total number of packets received during the current interval is the difference between current number of packets received and previous value of number of packet received stored from previous RR */*

2) expected = AdamRTPstats.ehsr - base_seqno +1 *\/* The number of packets expected is computed by the receiver as the difference between the highest sequence number received and the first sequence number received */*

3) expected_interval = expected – previous_expected *\/* Difference between current and previous expected value. If this value equals to 0, this means all packets expected have arrived */*

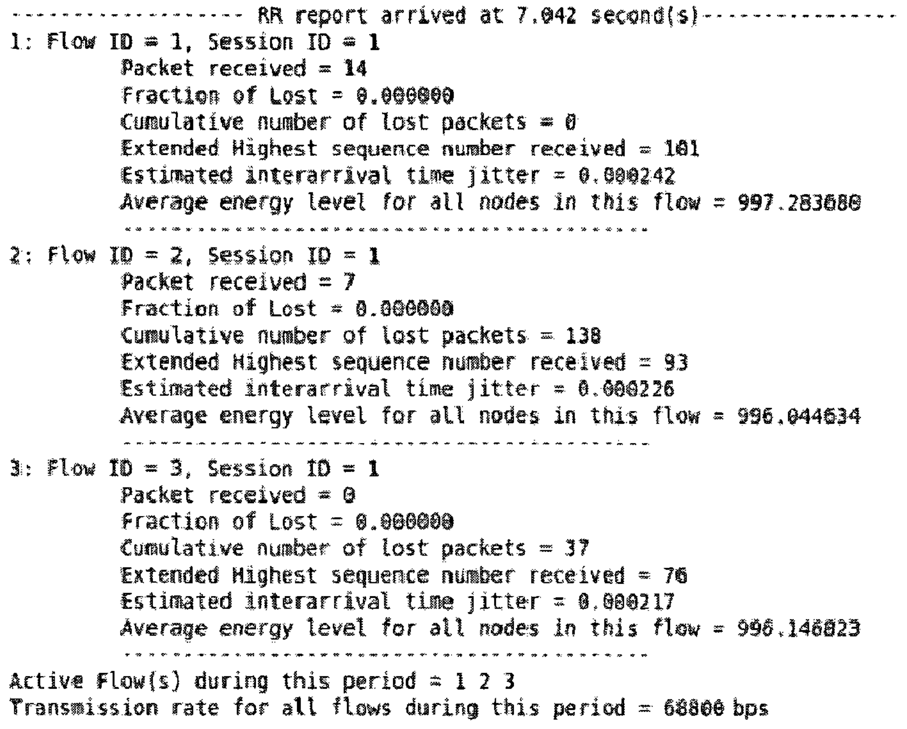
4) cumulative_pkt_lost = previous_cum_pkt_lost + (expected – AdamRTPstats.pktrecv) *\/* the cumulative number of packets lost calculated by adding the previous lost value and the difference between what is expected and the actually received packets */*

5) lost_interval = expected_interval – received_interval *\/* the lost value in the current interval equals to the difference between the number of packets expected in the current interval and the number of packets received during the same interval. In a healthy network, this value is negative, where expected_interval equals to 0 and received interval equals to a value > 0 */*

6) <u>Then compute fraction of packets lost, as follows:</u>

```
/* if the fraction of packet loss during the interval is 0 or negative (meaning no packet loss) or number of
packets expected during the interval equals to 0 (meaning no expected packets had arrived in this
interval), then, fraction of packet loss equals to Zero */

if (lost_interval <= 0 || expected_interval == 0 ) {

            fraction = 0

/* but if lost interval is greater than 0 (packet loss occurred) and expected interval does not equal to zero
(not all expected packets had arrived), in this case the value of fraction of packets lost equals to the
division of lost interval by expected interval    */

 } else { fraction = lost_interval  / expected_interval }
```

After getting these results, AdamRTCP agent stores them in the adamrtcp_header of the RR packet then, sends the RR packet to the sender. Figure 16 shows a result of RR packet summarizing a session of 3 flows.



```
------------------- RR report arrived at 7.042 second(s)---------------
1: Flow ID = 1, Session ID = 1
        Packet received = 14
        Fraction of Lost = 0.000000
        Cumulative number of lost packets = 0
        Extended Highest sequence number received = 101
        Estimated interarrival time jitter = 0.000242
        Average energy level for all nodes in this flow = 997.283080
        -----------------------------------------
2: Flow ID = 2, Session ID = 1
        Packet received = 7
        Fraction of Lost = 0.000000
        Cumulative number of lost packets = 138
        Extended Highest sequence number received = 93
        Estimated interarrival time jitter = 0.000226
        Average energy level for all nodes in this flow = 996.044634
        -----------------------------------------
3: Flow ID = 3, Session ID = 1
        Packet received = 0
        Fraction of Lost = 0.000000
        Cumulative number of lost packets = 37
        Extended Highest sequence number received = 76
        Estimated interarrival time jitter = 0.000217
        Average energy level for all nodes in this flow = 996.146023
        -----------------------------------------
Active Flow(s) during this period = 1 2 3
Transmission rate for all flows during this period = 68800 bps
```

Figure 16: AdamRTCP's RR packet

The sender reviews the result of each flow separately and judges whether to add or to remove a certain flow. The decision is made based on the value of Fraction of Lost. The value of Fraction of Lost for each flow is monitored individually. If this value, for all current active flows, is less than a MIN_TH value, i.e. network state is good, and the current number of flows is less than MAX_FLOWS, the sender adds a new flow after at least two RRs confirming this. By doing so, the sender makes sure that the WSN is free of congestions for some time before throttling the network with another new flow. If the Fraction of Lost value, for any given active flow, is greater than MIN_TH but less than MAX_TH, the sender marks the network as ACCEPTABLE and no action is taken so far. But, if the value of Fraction of Lost, for any given active flow, is greater than MAX_TH, even if the value of Fraction of Lost for all other flows is less than MIN_TH, then the affected flow is deleted unless the current number of flows is equal to MIN_FLOWS. If number of current flows is equal to MIN_FLOWS, then the sender adds another flow before deleting the bad one. Because if we delete the current flow first, then the current number of active flows will go below the value of MIN_FLOWS. That is why we need to add a new flow first.

4.4 Experiments

The AdamRTP script can simulate packet streaming based on a period of time or specific number of packets. For example, we can instruct the AdamRTP script to send a stream of packets for 10 minutes or we can tell the script to send 1000 packets only. As a result, we'll have a variety of simulations' options to estimate the results accurately. Each node is initialized with 1000 (Joules) as a source of power. Sending a packet consumes usually double the receiving power. A WSN could suffer from node failure and some external traffic that belongs to other streams or users. We randomized the node failure and the sources of external traffic. We have 3 different sources of external traffic, as shown in figure 17:

1. External traffic number 1 = from node(20) to node(29)
2. External traffic number 2 = from node(4) to node(94)
3. External traffic number 3 = from node(80) to node(89)

Four different types of simulation experiments were conducted. The first compares regular RTP behavior with AdamRTP. The second experiment compares single-flow with multi-flows performance. The third experiment compares AdamRTP with and without adaptations. Finally, the performance of AdamRTP in terms of energy consumption is evaluated.

Before experimentation, some terms need to be clarified. A congested network means a network that has an external source of traffic. As shown above, there are three external sources of traffic, one source of external traffic will be active at any time. Network with node failure is a network that suffers from some node failure. Node failures takes a uniform random probability of nodes to die for a random short duration, e.g. 2 seconds, starting from node 11 up to node 88 (excluding edge nodes). AdamRTP default packet size is 200 bytes. Transmission rate that we mostly start our script with is around 65 kbps. This means the sender sends a packet every 24.5 ms[8].

---

[8] ms stands for millisecond, 1 second = 1000 ms

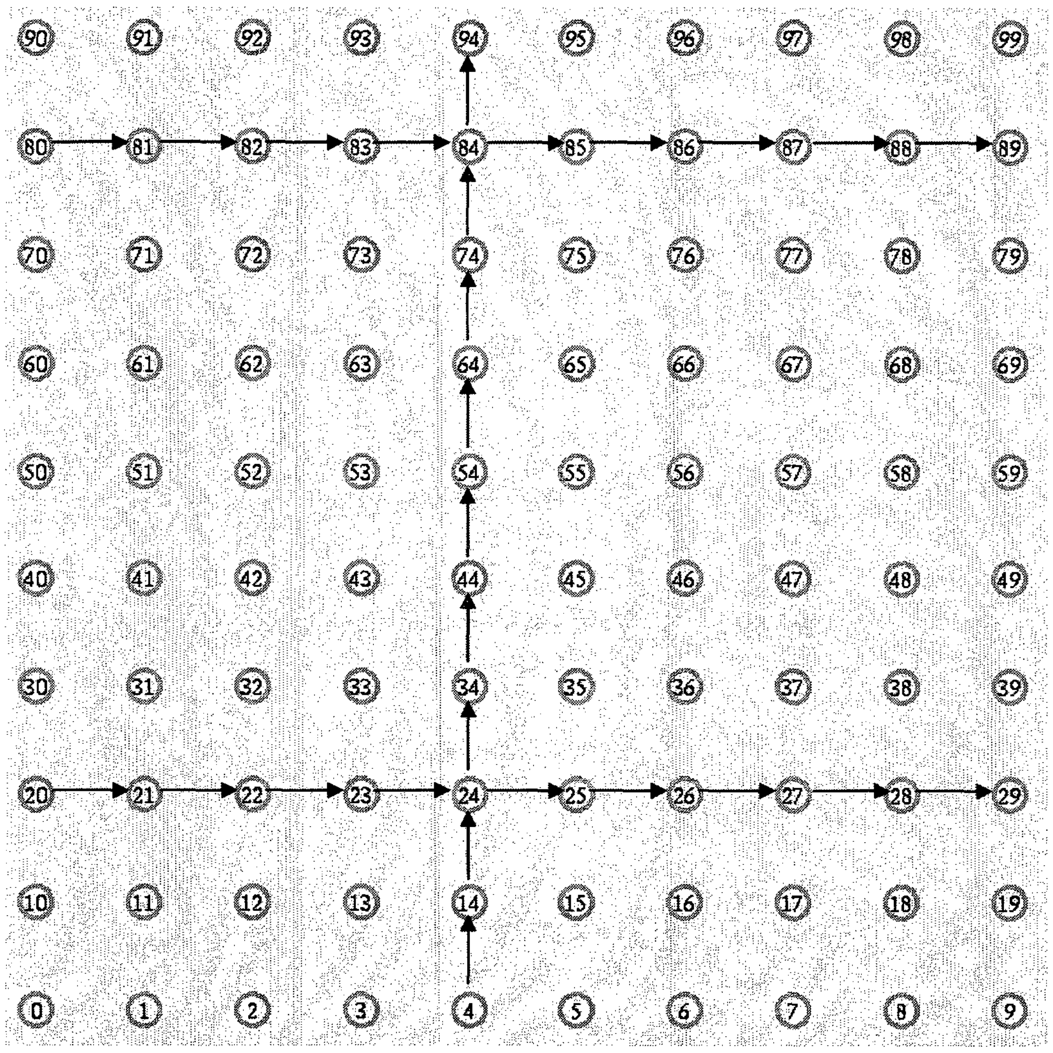


Figure 17: A 100 node topology with some sources of external traffic

4.4.1 RTP vs. AdamRTP

A new oTCL script was created for RTP with exactly the same topology and environment. We sent 1000 packets from node 50 to node 59 as shown in figure 14. The path was determined by the Ad hoc On-Demand Distance Vector (AODV) [Per03] routing protocol which is a single path routing protocol that is designed for Ad hoc networks. We simulated two scenarios, a healthy WSN network, where there is no node failure at all and no external traffic and a WSN with some node failure but no external traffic.

When running both scripts, AdamRTP and RTP, over a healthy network, we received all the packets sent in both scenarios. The average remaining power for the whole WSN resulted by both scripts is summarized in table 3:

Table 3: Power consumption comparison between RTP vs. AdamRTP

	RTP	AdamRTP with 5 flows
Minimum Node Power (J)	972.02	985.06
Average Node Power (J)	981.82	993.23

We calculated the average remaining power of the whole WSN, not only the sensor nodes that were involved in the transmission process because in wireless networks when one sensor sends any packet it will affect all surrounding nodes, even if this packet is destined for one node only. For example, in Figure 17, if node number 57 wants to communicate with node number 58, all nodes surrounding node 57 (which are node numbers 46, 47, 48, 56, 58, 66, 67, and 68) will consume some power receiving the ARP message that is generated by node 57 (the source) but only node 58 (the destination) will answer back with another ARP message telling the source that I'm here. This is part of the IEEE 802.11 MAC protocol specifications. Therefore, we felt the need to find out what is the effect of using many flows on the power usage of the whole WSN not only the nodes that were actively forwarding the packets between the source and the destination.

From Table 3, it is clear that AdamRTP consumed less power than RTP to send the same number of packets. However, when we instruct one node to fail for some time at a specific point of time on both scripts, then calculate the bandwidth[9] and the average power, we obtained the following results (shown in figure 18 and 19).

[9] Bandwidth calculated by multiplying number of packets received per second x packet size x 8

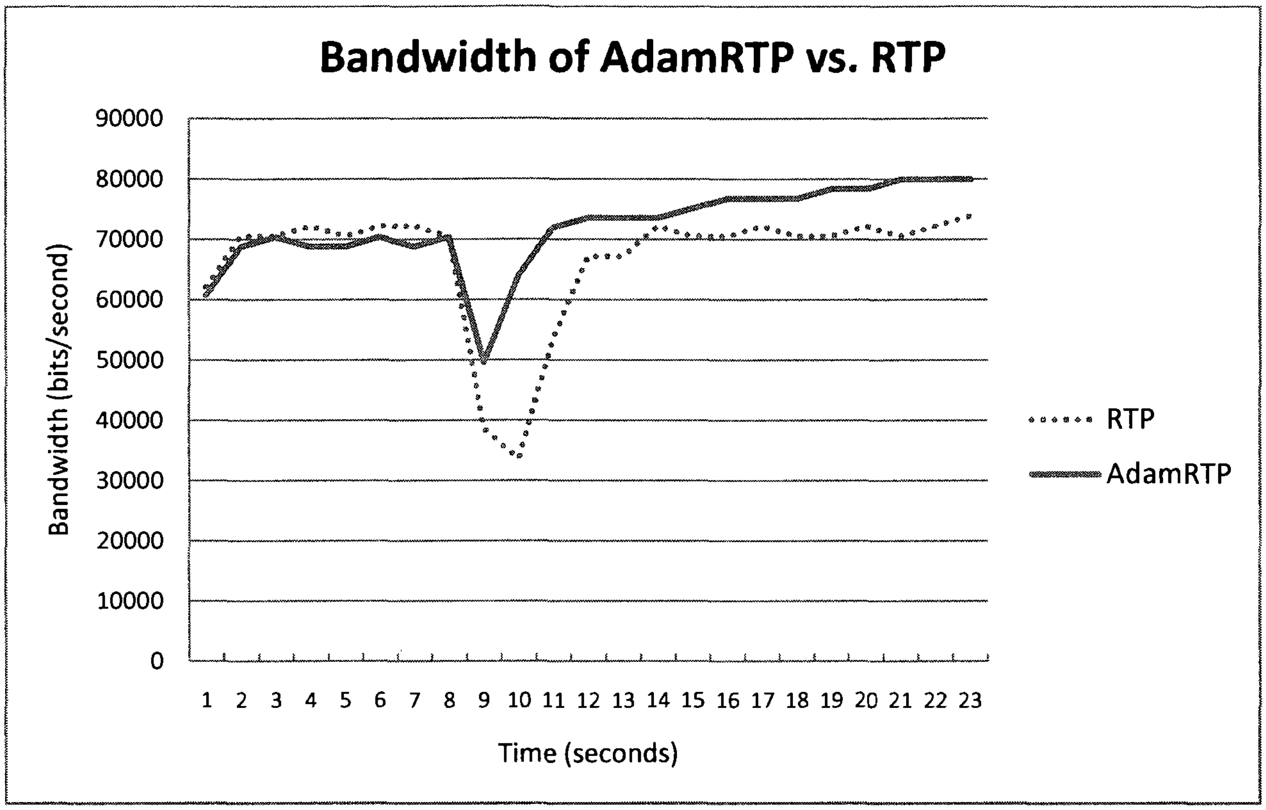


Figure 18: Bandwidth AdamRTP vs. RTP with node failure at the 8th second for 1 second duration

In Figure 18, when the node failed at the 8th second for about 1 second, AdamRTP was able to send packets using other flows. Only one flow was affected by the node failure, while in RTP, it took sometime to discover the failure before the routing protocol (AODV) could act upon this and rediscover another path to avoid the path that it is currently using which contains the dead node. Moreover, the sending rate in AdamRTP increases by time (up to a threshold) as long there is no packet loss as part of its adaptation technique. More rate adaptations are explained in Section 3.4.

The average power that remains in the whole WSN after sending 1000 packets over a network that has node failure is illustrated in Figure 19. Table 3 shows that in a healthy network the power consumed for AdamRTP with 5 flows is 1000 – 993.23 = 6.77 J, while Figure 19 shows that the power consumed in a network with node failure is 1000 – 993.15 = 6.85 J. AdamRTP with node failure consumed less power than when it ran in a healthy network because when the node failed for one second, the remaining nodes on the path to destination did not send any packets, thus saving power.

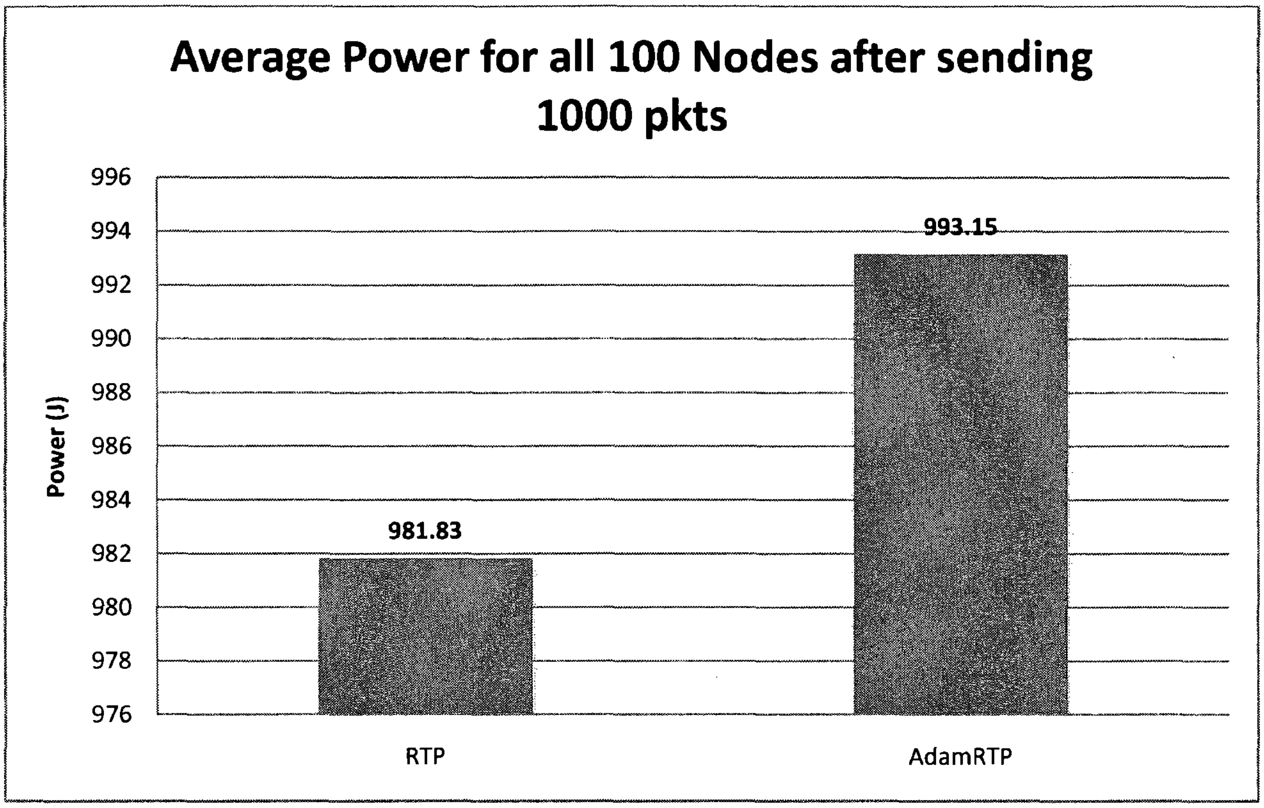


Figure 19: Average power RTP vs. AdamRTP over a network with node failure

4.4.2 AdamRTP single-flow vs. AdamRTP multi-flows

In this experiment, we compare the performance of AdamRTP in terms of the number of flows. We noticed that in a healthy network, where there is no node failure and no external traffic, the performance in terms of the number of packets received (received bandwidth) does not improve much because the transmission rate of the sender is the same in all cases whether it is using single or multi-flows. However, the average power consumption improves because the load of packet transmission is shared among more sensors.

The sender sends a stream of packets over a congested network, where we have randomly failing nodes and random external source of traffic (one source of external traffic at a time) as illustrated in Figure 17. The sender sends many packets for a duration of 60 seconds. The sender's transmission rate average is 70 Kbps, using one to five flows, each case we add one flow and run the script for 100 times each (the total running time equals 60 seconds x 5 (experiments) x 100 (times) = 3000 seconds (500 hours)). We computed the average packets received and the average power consumed for each case. Table 4 shows statistical results regarding the number of packets received when running the AdamRTP script, implemented in the sender, using one-flow and three-concurrent-flows respectively for 100 times each. When using 3-flows, we were able to perform better in terms of number of packets received. Note that the minimum value when sending 3-flows is higher than the minimum in 1-flow. This clearly indicates that multi-flows is capable of delivering more packets even in worst case scenario e.g. congestion.

Table 4: Packets received statistics of AdamRTP using one-flow vs. three-flows

AdamRTP using 1 flow		*AdamRTP using 3 flows*	
Mean	2292	Mean	2380
Standard Error	13.26	Standard Error	11.59
Median	2279	Median	2390
Mode	2409	Mode	2472
Standard Deviation	133	Standard Deviation	116
Minimum	1980	Minimum	2057
Maximum	2516	Maximum	2631
95% Confidence Interval	2292±26	95% Confidence Interval	2380±23

In order to test whether the data indicates differences between the two scenarios in terms of number of packet received, we carried out an independent two-sample t-test. The mean and standard deviation of the total number of packet received in the multi-flows scenario were 2380 and 116, respectively, whereas for the single-flow scenario, these quantities were 2292 and 133, respectively. Since the standard deviations were comparable, we used two-sample t-test with the assumption of equal variances. The statistic value obtained was 4.954754, which, when compared to a t-distribution with 198 degrees of freedom, gives a one-sided p-value of 7.75E-07. This is a strong indication that the average number of packet received by the multi-flow is statistically significantly higher than that of the single-flow.

Figure 20 shows a comparison regarding the average of number of packets received when running AdamRTP for 60 seconds over a congested and node failure network for 100 times.

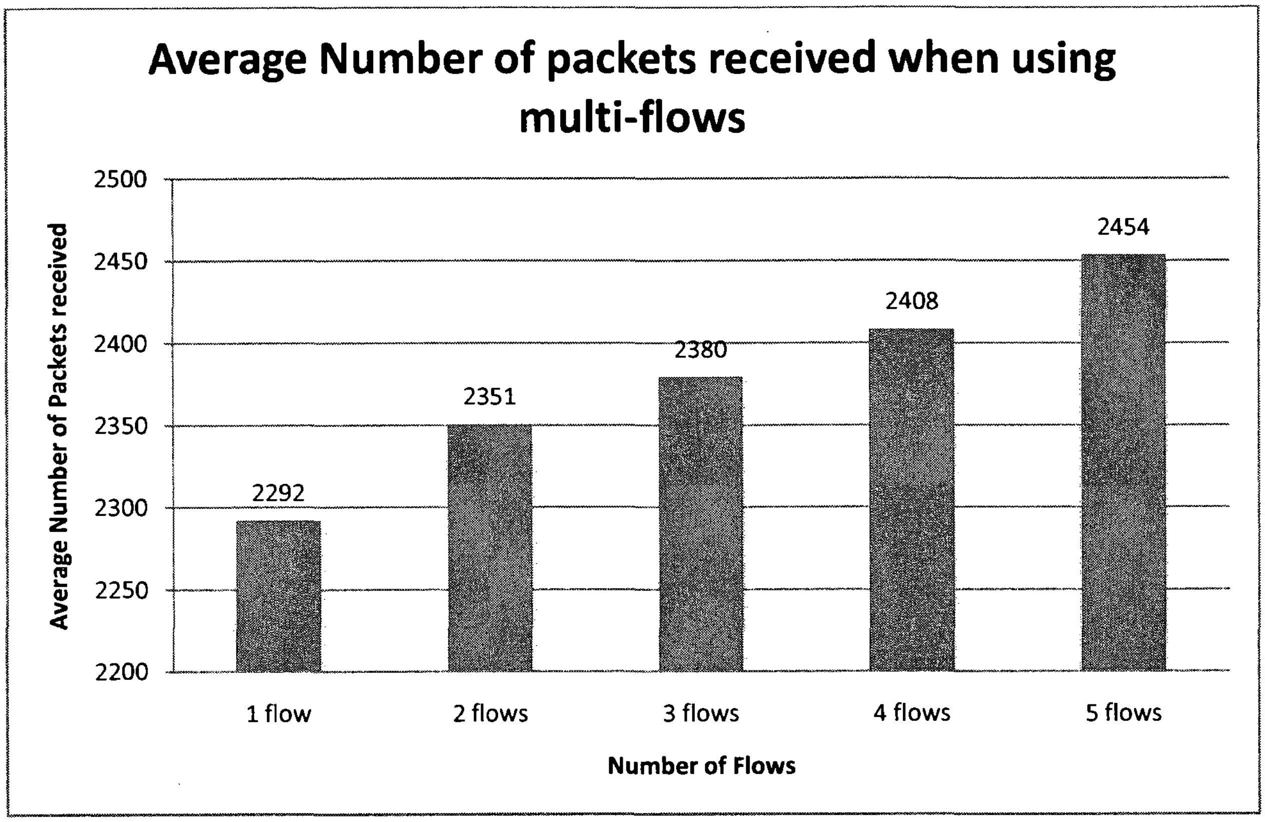


Figure 20: Comparison of AdamRTP using a single-flow and multi-flows over a congested network with node failure

There were 5 major experiments that were conducted to produce Figure 20. They are AdamRTP with 1-flow, 2-flows, ... , up to 5-flows. In all experiments, we instruct AdamRTP script to send packets from node 50 to node 59 (see Figure 14). The script ran for 60 seconds over a congested network, where one random source of external traffic was active at a time and with some node failure. One random node failure on the path between source and destination was active at a time. In our simulation, node failure selection takes uniform random probability from node 11 to node 88 (excluding edge nodes), although, we believe in real WSNs, number of node failures will not be very large.

AdamRTP starts always with one flow. If network state is good, AdamRTP will increment number of flows until reaching the MAX_FLOWS threshold. The MAX_FLOWS threshold value is fed by the script as an argument. Number of flow adaptation technique allows AdamRTP to increment active number of flows until maximum threshold is reached or network state goes to BAD where it will delete the affected flow and try to find a new alternative path right away. Rate adjustment adaptation technique increments the sending rate of the sender up to a threshold (MAX_RATE) until network reports that some congestion is occurring, where the sender will decrement the sending rate back to normal but not below another minimum threshold (MIN_RATE).

When a node fails on the path between the sender and the receiver, the receiver detects that there is no packet arriving on that flow (path). Therefore, in the next RR interval, the receiver informs the sender about this incident. In case the sender has MIN_FLOWS (default value is one), it instructs the routing protocol to change the affected path and look for another one. Once the routing protocol locates a new path, then the sender starts using this path to send the remaining packets over the new path. If the sender is sending more than one flow (multi-flows) and one of the nodes failed on one of the paths, the affected flow will be deleted (number of current flows will be decremented by one) and the other current flows will continue transmitting packets without interruption.

From figure 20, it is clearly illustrated that the performance improves when the number of flows increases, i.e. when used with adaptations.

Figure 21 shows a comparison between 1-flow and 5-flows in terms of bandwidth recorded every 1 second at the sink.

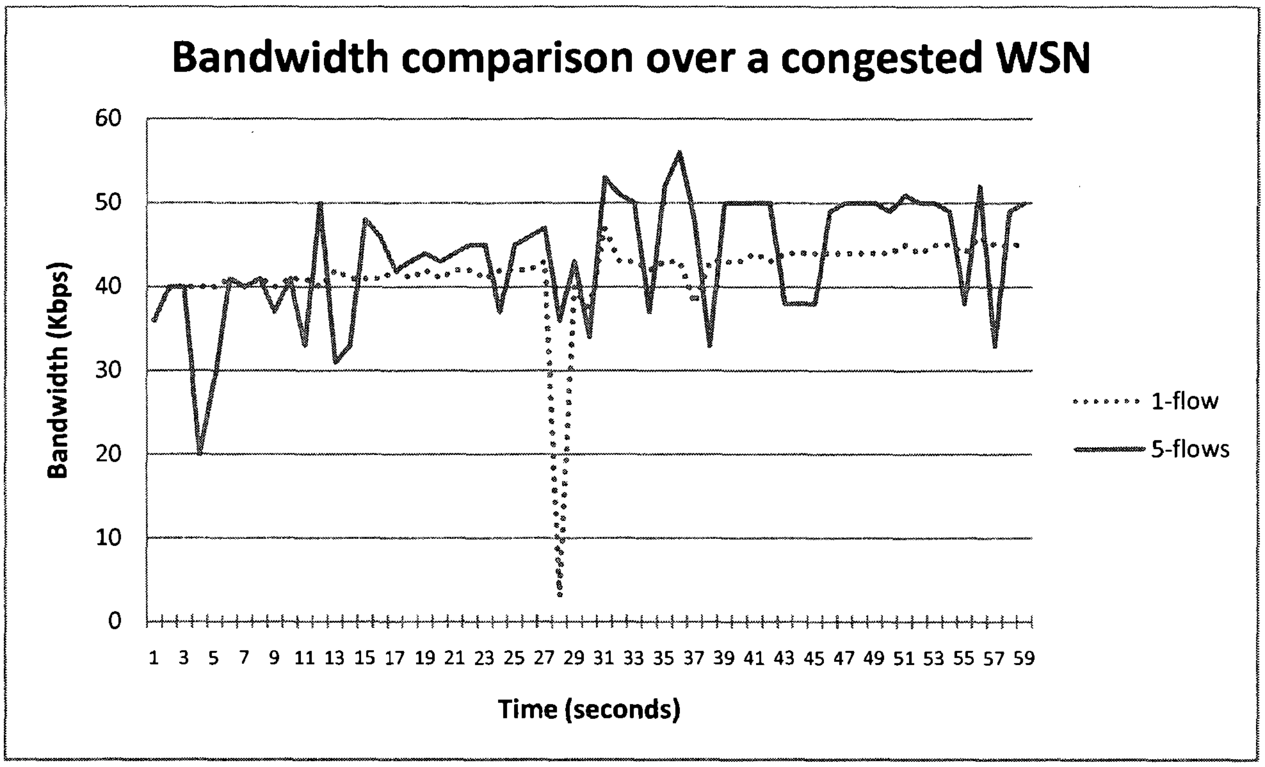


Figure 21: Bandwidth used in single-flow and bandwidth used in multi-flows over congested with node failure network

Figure 21 shows two different simulations. One simulation uses a single-flow AdamRTP, while the other simulation uses multi-flows AdamRTP. In case of multi-flows AdamRTP, it starts with single-flow at the beginning, then increases the number of flows gradually. In congested with node failure network, AdamRTP with multi-flows (5-flows) always maintains bandwidth over 30 kbps except at the beginning when the number of flows is minimum. Hence, we can always have some packets delivered even in the severe network conditions. However, when running the script for a single-flow, the bandwidth dropped to 3 kbps at the 28th seconds. Note that this is considered to be very low compared to the behavior of AdamRTP with multi-flows.

Figure 21 shows that the performance of multi-flows AdamRTP fluctuates over time. Because we have a network that experiences severe congestion at random times and where the nodes fail randomly, then AdamRTP adapts to alleviate these severe network conditions. Note that the chance to use a path that suffers from node failure increases when there are multi-flows, where one of the flows can be using a path with node failure.

4.4.3 AdamRTP without Adaptation vs. AdamRTP with Adaptation

The adaptation techniques in AdamRTP provide flexible reactions to frequent changes in the WSN. Once again, the script ran 100 times and the average number of packet received in the four scenarios was recorded as follows: when using one flow with and without adaptations, and when using 3 flows with and without adaptations. Figure 22 shows the results.

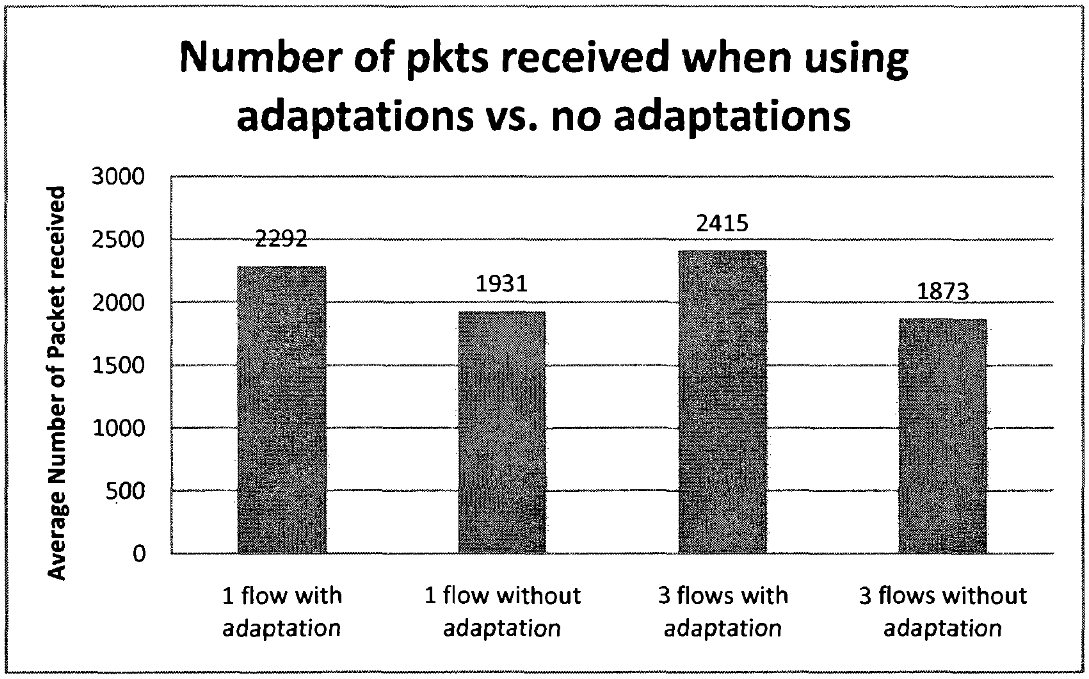


Figure 22: AdamRTP adaptation vs. no adaptation

AdamRTP starts usually with one flow, like slow start phase in TCP. Then, if RR reports a congestion-free and no node failure network, AdamRTP will increase number of flows by one until MAX_FLOWS threshold is reached. The MAX_FLOWS threshold number is fed to the script as a parameter, e.g *adamrtp.tcl 5 1000*. That is to say, this script sends 1000 packets using up to a maximum of 5 flows.

Two adaptation techniques are applied here: number of flows and rate adjustment. When AdamRTP protocol uses single flow only, the sender does not increase the number of flows during the transmission. It is always one flow. When using adaptation, AdamRTP may instruct the routing protocol to change the path if packets arrive late, which are considered lost. However, in case of one flow without the adaptation scenario, late packets are dropped without the ability to change that affected flow (path).

In Figure 22, it is clear that AdamRTP acts poorly when using 3 flows with no adaptations compared to the behavior of AdamRTP with single-flow and no adaptation. That is because the probability of having more node failures on the three paths carrying the three flows is higher compared to node failure on a single path carrying a single flow.

Next figure shows the performance of number of flows adaptation over time when instructing AdamRTP to send 3000 packets over congested with node failure network. The transmission time was 69 seconds. Number of flows was recorded every 1 second. Figure 23 shows the result.

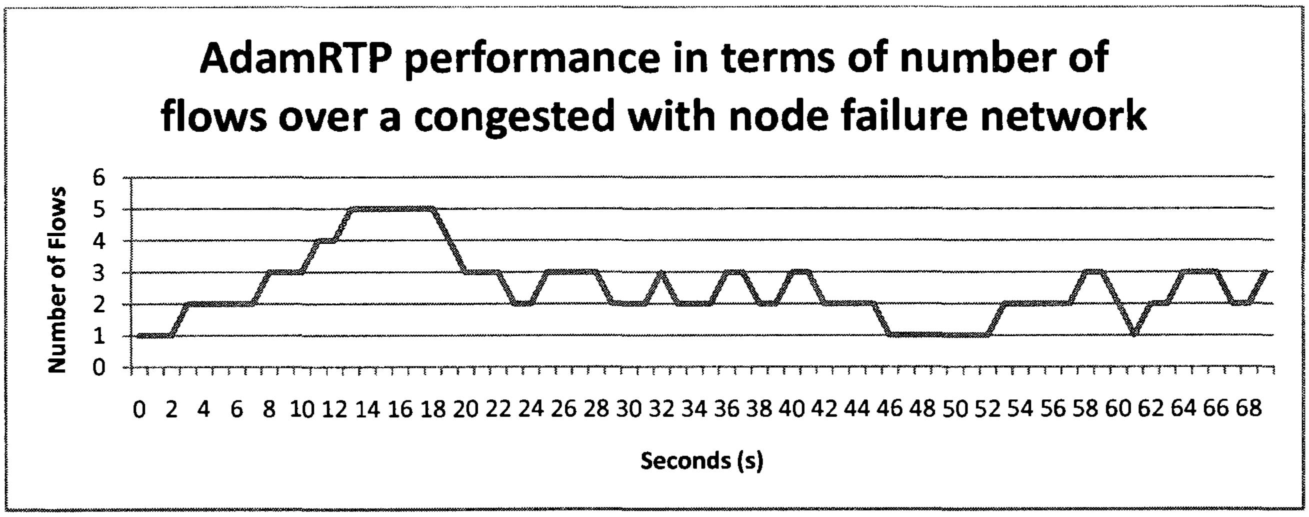


Figure 23: Change of number of flows over congested with node failure network

Figure 23 illustrates the number of flow adaptation technique that AdamRTP is using over time. AdamRTP starts with one flow always and after at least 2 RRs confirming that the network is good, it increments number of flows by 1 until reaching the MAX_FLOWS threshold, which is 5 in this case. It stays with 5 flows until the network reports congestion or node failure, as shown in the 19th second, where the number of flows dropped to 3. Next graph is a comparison with Figure 23 where AdamRTP is transmitting over a congested network with no node failure.

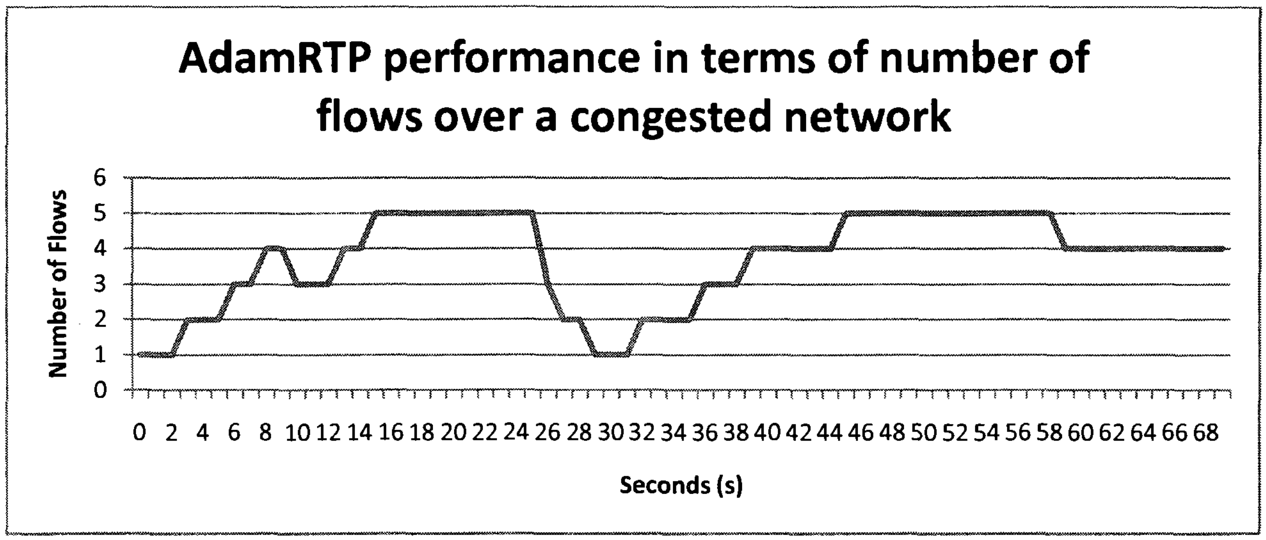


Figure 24: Change of number of flows over a congested network with no node failure

Figure 24 shows a better performance than Figure 23. In this case, we instruct AdamRTP to send 3000 packets over a congested network with no node failure. It is clear that the performance is better compared to Figure 23 where the adaptation technique was able to increase the number of flows to reach the maximum threshold value most of the time during the simulation period.

4.4.4 Energy consumption

Initially, each node in the network is configured with 1000 Joules as a source of power. When a node wants to transmit a packet, it consumes twice the amount of power compared to receiving a packet. As a result, when we have nodes with limited source of power, a WSN is able to send a little more when using multi-flows compared to single flow. That is to say, each intermediate node in the multi-flows scenario sends fewer packets, thus saving more power per node.

We compare two situations here. The first situation is, sending 1000 packets over a healthy WSN, where each node has 1000 Joules as source of power, while the second situation when each node has a limited source of power (e.g. 50 Joules). We choose a healthy network because in a network with node failure, if packet loss occurs on the path to receiver due to node failure, the remaining sensor nodes on the path will not send any packets. Therefore, they consume no power, which is not a fair comparison with a situation where we have more node failure.

Table 5 shows a comparison of the minimum power remaining among all sensors in the WSN and the average remaining power for the whole WSN when sending 1000 packets over a healthy network in single and multi flows scenarios. See page 63 for an explanation of why all sensor nodes are used for power calculation and not just the sensors on the paths carrying the data flows.

Table 5: Power comparison between single and multi-flows

	Single-Flow	Multi-Flows (5-flows)
Node's Minimum Power (J)	983.67	985.75
Average Power (J)	992.68	993.31

When comparing Table 3 and Table 5, we noticed that the performance of RTP and AdamRTP with single flow in terms of power consumption is different. We think the reason why RTP was consuming more power is due to the nature of the AODV routing protocol that RTP was functioning over. The AODV routing protocol requires initialization and may refresh the routing table several times during the transmission period especially if there is a node failure in the route to destination. This results in power consumption greater than AdamRTP which is using a static routing table so there is no need to refresh the routing table in the intermediate nodes. Besides that, RTP uses the complete RTCP protocol with its complete suite of packet collections such as RR, SR, SDES, BYE and APP. But, AdamRTP is using only RR packet which makes AdamRTP simpler and lighter to operate over WSN, therefore, saves more power.

We can derive the following from table 5: Transmitting 1000 packets, using multi-flows, will consume (1000 – 993.3053) = 6.6947 (J). Hypothetically, total number of packet that can be sent until the WSN dies (1000 x 1000 / 6.6947) = 149,372 packets. While in a single-flow, it is (1000 x 1000 / (1000 - 992.6814)) = 136,638 packets. This means a WSN which has 1000 J as source of power can deliver (149,372 – 136,638) = 12,734 packets more by using multi-flows. This is equal to (12,734 x 200 *(packet* *size in byte))* = 2,546,837 Bytes, which is equivalent to about 2.225 minutes of extra video feed when using a camera with resolution of 320x240 (QVGA) and MPEG4 compression [10].

When limiting the source of power to a smaller value (less than 1000 J), AdamRTP with multi-flows was able to receive more packets than with a single-flow. Because AdamRTP is able to distribute the traffic load over more sensors, so each intermediate sensor in case of multi-flows will send fewer packets compared to a single-flow scenario. Table 6 shows a comparison between a single-flow and multi-flows performance when each node has low source of power. Note that fewer packets are received compared to our hypothetical estimate because of the low battery source. According to NS-2 documentation, a sensor will check for the remaining power level in its own battery. If the value of the remaining power is less than the Pt_ (power transmit value), which is the amount of power used to transmit the packet, then the sensor will not be able to send anymore packets although it has some small amount of power remaining resulting in a node failure.

Table 6: WSN performance in terms of number of packet received when nodes have limited source of power

	Number of packets received using Single-Flow	Number of packets received using Multi-Flows	Enhanced delivery rate
Power = 10 J	645	807	25%
Power = 30 J	1936	2335	21%
Power = 50 J	3226	4048	25%
Power = 75 J	4838	6074	26%
Power = 100 J	6451	8098	26%
Power = 200 J	12903	16196	26%
Power = 1000 J	64768	83194	28%

From Table 6, to calculate the percentage change between the two columns (single-flow and multi-flows) we need to divide the absolute value of the difference of the second and the first numbers by the first number. For example ABS(4838-6074) / (4838) = 26%. This indicates that when using multi-flows, we can enhance the delivery rate by 26% more than a single-flow method of delivery.

[10] We use "IP Camera Bandwidth & Disk Space Calculator v 2.0" for calculations. Downloaded from http://www.video-home-surveillance.com/ipcameracalc

Figure 23 shows that we were able to extend the life-time of the whole WSN by using a multi-flow technique and we were able to receive more packets at their expected times, thus enhancing QoS.

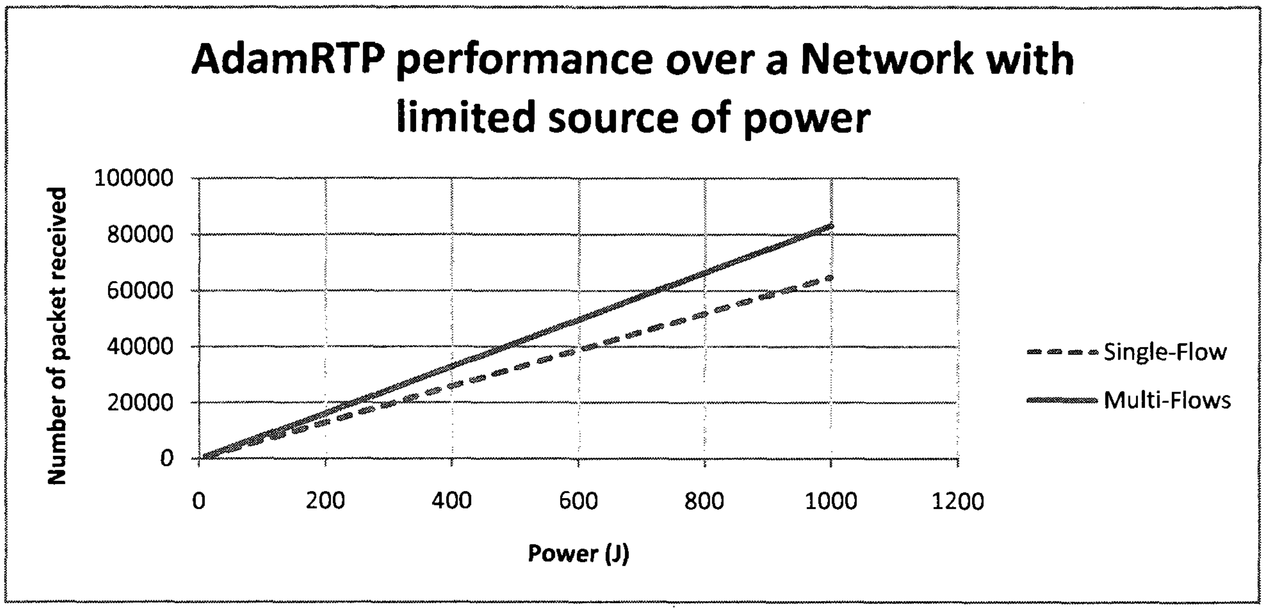


Figure 25: AdamRTP performance over a network with limited source of power

4.5 Summary

We used the Network Simulator NS-2 for conducting our simulation experiments. In this chapter, we explained the simulation environment and configuration followed by a detailed description of implementation scenarios.

We conducted four types of experiments. The first was a simulation comparing our proposed protocol, AdamRTP, with original RTP. Second, we showed the advantages of using more than one flow. Third, we considered how adaptations, by changing number of flow, could enhance the delivery rate. Finally, we compared the energy consumption between AdamRTP and regular protocols that uses single flow.

The experiments provided evidence that AdamRTP performed better than regular RTP when sending multimedia streams over a WSN. Also AdamRTP with its multi-flows and adaptation techniques has improved performance in terms of QoS.

CHAPTER V

CONCLUSIONS AND RECOMMENDATIONS

5.0 Conclusions

The need for real-time multimedia applications is increasing. Real-time multimedia applications are known to be resource hungry. This affects the capabilities of such applications to operate without some limitations over Wireless Sensor Networks which normally suffer from computational and memory constrains. Real-time multimedia applications normally require more computational and storage capabilities than regular network applications e.g. HTML WebPages and E-mail, beside the need for more bandwidth for transmission. The main obstacle to any proposed WSN protocol is power consumption. Achieving a good Quality of Service (QoS) is always at the expense of network resources such as power. Therefore, we need to achieve a balance between QoS and power consumption. WSNs are usually densely deployed. This creates many paths between any two sensors. Therefore, we can utilize this feature and divide the large multimedia files into smaller ones and then send each small portion over a separate path. This can ease the burden of delivering the multimedia stream from fewer nodes and share it among many others. This inspired us to propose Adaptive Multi-flow Real-time Multimedia Transport Protocol for Wireless Sensor Networks (AdamRTP), which divides one multimedia stream into many flows using an MDC coder, then sends each flow over a separate path (if possible) back to the destination. In addition to the multi-flow technique, AdamRTP employs some adaptation techniques that act like congestion control. AdamRTP uses three different adaptation techniques: number of flow, rate adaptation, and energy.

Our simulation results show that AdamRTP has enhanced the delivery, specially when the WSN is suffering from congestion due to node failure or external source of traffic, It was also able to reduce the power consumption and deliver more packets if the nodes have limited power source. We simulated four different scenarios. First, we compared AdamRTP with Real-time Transport Protocol (RTP) and the results showed that AdamRTP was able to perform better than RTP in terms of delivery rate and energy consumptions. Second, simulation studies confirmed that using more than one flow of transmission enhances the packet receiving rate compared to using only one flow in the presence of network congestion. Third, simulation experiments are about adaptation techniques. The results showed that using adaptation techniques such as number of flow and rate adjustment enhanced the packet delivery rate. Last scenario is about energy consumption, AdamRTP was able to transmit more packets when nodes suffer from limited source of power compared to single-flow technique. Therefore, AdamRTP successfully extends the life time of the WSN.

5.1 Recommendations

Some enhancements can be added to AdamRTP for further improvements. In our implementations, the coder divides the stream creating multiple flows of approximately equal importance (symmetric coding). A different, non-symmetric coding, could divide the media into two different frames: Intra-frame (I-frame) that contains all of the information to construct a whole media and Intra-frame (B-frame) that contains less data than I-frames, and is generated by looking at the difference between the present frame and the previous one. B-Frame can be used to enhance the quality of the received I-frame and correct errors caused by transmissions. By employing this feature, we could instruct AdamRTP to use the most efficient flow to send the packets that belong to the I-frame and another flow to send the packets that hold the B-frame. In figure 16, flow number 1 is an example that shows some flows are better than others in term of power consumption and packet loss.

Another possible modification is related to the interval between issuing two feedback received reports (RRs). Usually the RR helps the sender to have advanced knowledge about the state of the network especially when there is congestion. However, in reality, congestion and node failures are not very common, so we can modify the interval of sending the RR and ask the receiver to send its own reports only when it detect congestion. By this, we could free the network's bandwidth for sending actual data, therefore, saving more power.

REFERENCES

[Ahm74] Ahmed, N., Natarajan, T., & Rao, K. R. (1974). *Discrete Cosine Transform.* IEEE Transactions on Computers, Volume 23, pp. 90-93.

[Akk05] Akkaya, K., & Younis, M. (2005). *A survey on routing protocols for wireless sensor network.* Ad Hoc Networks, 3 (3), pp. 325–349.

[Akk03] Akkaya, K., & Younis, M. (2003). *An Energy-Aware QoS Routing Protocol for Wireless Sensor Networks.* The 23rd International Conference on Distributed Computing Systems, pp. 710- 715. Providence, Rhode Island.

[Aky02] Akyildiz, I., Su, W., Sankarasubramaniam, Y., & Cayirci, E. (2002, August). A *survey on sensor networks.* Communications Magazine, 40 (8), pp. 102 - 114.

[Aky05] Akyildiz, I. F., Vuran, M. C., Akan, Ö. B., & Su, W. (2005). *Wireless Sensor Networks: A Survey Revisited.* Elsevier Computer Networks Journal.

[Aky06] Akyildiz, I. F., Vuran, M. C., & Akan, O. B. (2006). *A Cross-Layer Protocol for Wireless Sensor Networks.* 40th Annual Conference on Information Sciences and Systems (CISS '06). pp. 1102 - 1107. Princeton, NJ.

[Ati01] Atiquzzaman, M., & Hassan, M. (2001). *Adaptive Real-Time Multimedia Transmission over Packet Switching Networks.* Real-Time Imaging, 7 (1), pp. 219-220.

[Bol94] Bolot, J.-C., & Turletti, T. (1994). *A rate control mechanism for packet video in the Internet.* IEEE Infocom, 3, pp. 1216-1223. Toronto.

[Bra94] Braden, R., Clark, D., & Shenker, S. (1994, June). *Integrated Services in the Internet Architecture: an Overview.* RFC1633. Network Working Group.

[Bus02] Bushmitch, D., Panwar, S., & Pal, A. (2002). *Thinning, striping and shuffling: traffic shaping and transport techniques for variable bit rate video.* Global Telecommunications Conference, GLOBECOM '02. 2, pp. 1485- 1491. IEEE.

[Cha03] Chakareski, J., Han, S., & Girod, B. (2003). *Layered coding vs. multiple descriptions for video streaming over multiple paths.* Proceedings of the eleventh ACM international conference on Multimedia. pp. 422 - 431. Berkeley, CA, USA: ACM.

[Che04] Chen, D., & Varshney, P. K. (2004). *QoS support in wireless sensor networks: a survey.* International Conference on Wireless Networks (ICWN '04), 1, pp. 227-233. Las Vegas, NV, USA.

[Dur08] Durresi, A., Paruchuri, V., Durresi, M., & Barolli, L. (2008). *Adaptive Layered Multimedia Transmissions over Wireless Networks.* International Conference on Distributed Computing Systems Workshops, ICDCS '08. pp. 48-53. IEEE.

[ElM01] El-Marakby, R. (2001). *NS Implementation of RTP/RTCP.* The 1st IEEE International Symposium on Signal Processing and Information Technology. Cairo, Egypt: IEEE.

[Est01] Estrin, D., Girod, L., Pottie, G., & Srivastava, M. (2001). *Instrumenting the world with wireless sensor networks.* IEEE International Conference on Acoustics, Speech, and Signal Processing. 4, pp. 2033-2036. Salt Lake City, Utah: IEEE.

[Fel06] Felemban, E., Lee, C.-G., & Ekici, E. (2006, June). *MMSPEED: Multipath Multi-SPEED Protocol for QoS Guarantee of Reliability and Timeliness in Wireless Sensor Networks.* IEEE Transaction on Mobile Computing , 5 (6), pp. 738-754.

[Gan01] Ganesan, D., Govindan, R., Shenker, S., & Estrin, D. (2001, Oct). *Highly-resilient, energy-efficient multipath routing in wireless sensor networks.* ACM SIGMOBILE Mobile Computing and Communications Review , 5 (4), pp. 11 - 25.

[Gar03] Garcia-Macias, J. A., Rousseau, F., Berger-Sabbatel, G., Toumi, L., & Duda, A. (2003, July). *Quality of service and mobility for the wireless internet.* Wireless Networks, 9 (4), pp. 341 - 352.

[Goy01] Goyal, V. K. (2001). *Multiple Description Coding: Compression Meets the Network.* IEEE Signal Processing Magazine, 18 (5), pp. 74-94.

[HeT03] He, T., Stankovic, J., Lu, C., & Abdelzaher, T. (2003). *SPEED: a stateless protocol for real-time communication in sensor networks.* 23rd International Conference on Distributed Computing Systems, pp. 46- 55.

[Hua03] Huang, C.-F., & Tseng, Y.-C. (2003). *The coverage problem in a wireless sensor network.* The 2nd ACM international conference on Wireless sensor networks and applications pp. 115--121. San Diego, CA, USA: ACM.

[ISIUSC] Information Sciences Institute at USC. (n.d.). *The Network Simulator - NS-2.* Retrieved from http://www.isi.edu/nsnam/ns/

[Kim05] Kimura, N., & Latifi, S. (2005). *A Survey on Data Compression in Wireless Sensor Networks.* International Conference on Information Technology: Coding and Computing (ITCC'05). 2, pp. 8-13. Las Vegas, NV: IEEE.

[Koi00] Koistinen, T. (2000). *Protocol overview: RTP and RTCP.* Nokia Telecommunications.

[Kot03] Kotz, D., Newport, C., & Elliott, C. (2003). *The mistaken axioms of wireless-network research.* In Technical Report TR2003-467, Hanover, NH, USA: Dartmouth College Computer Science.

[Kur03] Kurose, J., & Ross, K. (2003). *Computer Networking A top-down approach featuring the Internet* (2nd ed.). Addison Wesley.

[Kyu05] Kyu-Han, K., Zhu, Y., Sivakumar, R., & Hung-Yun, H. (2005). *A receiver-centric transport* *protocol for mobile hosts with heterogeneous wireless interfaces.* Wireless Networks, 11 (4), pp. 363-382.

[Lak98] Lakshman, T., Ortega, A., & Reibman, A. R. (1998). *VBR Video: Tradeoffs and Potentials.* Proceedings of the IEEE, 86, pp. 952-973.

[Lot03] Lotfallah, O. A., & Panchanuthan, S. (2003). *Adaptive Multiple Description Coding for Internet Video.* IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '03). 5, pp. 732-735. Hong Kong: IEEE.

[Mad02] Madden, S., Franklin, M. J., Hellerstein, J. M., & Hong, W. (2002). *TAG: a Tiny AGgregation service for ad-hoc sensor networks.* ACM SIGOPS Operating Systems Review, 36, pp. 131--146.

[Mah99] Mahadevan, I., & Sivalingam, K. M. (1999). *Quality of Service Architectures for Wireless Networks: IntServ and DiffServ Models.* International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN '99) p. 420 - 425. Washington, DC, USA: IEEE Computer Society.

[Man07] Mangharam, R., Rowe, A., & Rajkumar, R. (2007). *FireFly: a cross-layer platform for real-time embedded wireless networks.* Real-Time Systems, 37 (3), pp. 183 - 231.

[ManSim] Manna Research Group & Sensornet Project. (n.d.). *Mannasim simulation environment.* Retrieved from http://www.mannasim.dcc.ufmg.br/index.htm

[Mao06] Mao, S., Bushmitch, D., Narayanan, S., & Panwar, S. (2006). *MRTP: a multiflow real-time transport protocol for ad hoc networks.* Transactions on Multimedia, 8 (2), pp. 356 - 369.

[Meg01] Meguerdichian, S., Koushanfar, F., Potkonjak, M., & Srivastava, M. (2001). *Coverage problems in wireless ad-hoc sensor networks.* INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. 3, pp. 1380-1387.

Anchorage, AK, USA: IEEE.

[MKQ01] Meguerdichian, S., Koushanfar, F., Qu, G., & Potkonjak, M. (2001). *Exposure in wireless Ad-Hoc sensor networks.* MobiCom '01: The 7th annual international conference on Mobile computing and networking pp. 139-150. New York, NY, USA: ACM.

[Nas07] Nasser, N., & Chen, Y. (2007). *Secure Multipath Routing Protocol for Wireless Sensor Networks.* The 27th International Conference on Distributed Computing Systems Workshops, ICDCSW '07 pp. 12-19. Washington, DC, USA: IEEE Computer Society.

[Nic98] Nichols, K., Blake, S., Baker, F., & Black, D. (1998, December). *Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers.* RFC 2474. Network Working Group.

[NS2] Information Sciences Institute. University of Southern California. *Network Simulator NS-2.* http://www.isi.edu/nsnam/ns.

[Per03] Perkins, C., Belding-Royer, E., & Das, S. (2003, July). *Ad hoc On-Demand Distance Vector (AODV) Routing.* RFC 3561 . Network Working Group.

[ResU6] Research Unit 6 at Computer technology Institute. (n.d.). *Extensions to NS-2 RTP code.* Retrieved from http://ru6.cti.gr/ru6/ns_rtp_home.php

[San03] Sankarasubramaniam, Y., Akan, Ö. B., & Akyildiz, I. F. (2003). *ESRT: event-to-sink reliable transport in wireless sensor networks.* International symposium on Mobile ad hoc networking & computing. pp. 177 - 188. Annapolis, Maryland, USA: ACM.

[Scf03] Schulzrinne, H., Casner, S., Frederick, R., & Jacobson, V. (2003, July). *RTP: A Transport Protocol for Real-Time Applications.* RFC 3550. Network Working Group.

[Sch03] Schulzrinne, H., & Casner, S. (2003, July). *RTP Profile for Audio and Video Conferences* *with Minimal Control*. RCF 3551. Network Working Group.

[Sch96] Schulzrinne, H., Casner, S., Frederick, R., & Jacobson, V. (1996, January). *RTP: A Transport Protocol for Real-Time Applications*. RFC 1889. Network Working Group.

[Soh00] Sohrabi, K., Gao, J., Ailawadhi, V., & Pottie, G. J. (2000). *Protocols for Self-Organization of a Wireless Sensor Network*. Personal Communication Magazine, 7 (5), pp. 16–27.

[Sta03] Stann, F., & Heidemann, J. (2003). *RMST: Reliable Data Transport in Sensor Networks*. IEEE International Workshop on Sensor Network Protocols and Applications, pp. 102- 112. Anchorage, Alaska, USA.

[MNRCH] The Rice University Monarch Project. (n.d.). *Wireless and Mobility Extensions to NS-2*. Retrieved from http://www.monarch.cs.rice.edu/cmu-ns.html

[Til02] Tilak, S., Abu-Ghazaleh, N. B., & Heinzelman, W. (2002, April). *A taxonomy of wireless micro-sensor network models*. ACM SIGMOBILE Mobile Computing and Communications Review, 6 (2), pp. 28 - 36.

[URI03] University of Rhode Island. (2003). *Compression 101*. Retrieved from http://www.uri.edu/artsci/langlab/documentation/compress101.html

[Wan02] Wan, C.-Y., Campbell, A. T., & Krishnamurthy, L. (2002). *PSFQ: a reliable transport protocol for wireless sensor networks*. International workshop on Wireless sensor networks and applications pp. 1 - 11. Atlanta, Georgia, USA: ACM.

[Wan05] Wang, Y., Reibman, A. R., & Lin, S. (2005). *Multiple Description Coding for Video Delivery*. Proceedings of the IEEE, 93, pp. 57-70.

[Won06] Wong, A., & Bishop, W. (2006). *Practical Content-Adaptive Subsampling for Image and Video Compression*. Eighth IEEE International Symposium on Multimedia, ISM'06. pp. 667 -

673. San Diego, California: IEEE.

[Xue07] Xue, Y., Lee, H. S., Yang, M., Kumarawadu, P., Ghenniwa, H., & Shen, W. (2007). *Performance Evaluation of NS-2 Simulator for Wireless Sensor Networks.* Canadian Conference on Electrical and Computer Engineering, CCECE 2007. pp. 1372-1375. IEEE.

[YeW] Ye, W. (n.d.). *Radio Propagation Models chapter from the NS Manual.* Retrieved from http://www.isi.edu/~weiye/pub/propagation_ns.pdf

[Yew02] Ye, W., Heidemann, J., & Estrin, D. (2002). *An Energy-Efficient MAC protocol for Wireless Sensor Networks.* IEEE Infocom. pp. 1567-1576. New York, NY, USA: IEEE.

[You04] Younis, M., Akkaya, K., Eltoweissy, M., & Wadaa, A. (2004). *On Handling QoS Traffic in Wireless Sensor Networks.* The 37th Annual Hawaii International Conference on System Sciences. pp. 10-19. Big Island, Hawaii: IEEE Computer Society.

[Yua05] Yuan, Y., Chen, H., & Jia, M. (2005). *An Optimized Ad-hoc On-demand Multipath Distance Vector (AOMDV) Routing Protocol.* Asia-Pacific Conference on Communications. pp. 569-573. Perth, Australia: IEEE.

APPENDICES

Appendix A

Terminology

Name	**Definition**
Adaptation	Autonomic changes in network state designed to maintain best possible communication performance.
Bandwidth	The amount of data that can be transferred over a network connection in a given period of time. Bandwidth is usually measure in bits per second.
Codec	Mathematical algorithms used to compress and reduce the number of bytes that represent any given multimedia file.
Disjoint path	Two or more paths those are not joint by any intermediate node.
Encoding	A process of transforming text, picture, audio and video to a digital format using mathematical algorithms for the purpose streaming or downloading.
Flow	Data packets transferred from the sender to the receiver using certain path.
Jitter	The unwanted variation in the time between arriving packets, caused by network congestion or route changes.
Joint path	Two or more paths which are sharing one or more intermediate node
Multimedia	Presentation of information using audio, video, pictures, and animations instead of regular text
Multiple Description Coding	A technique to fragment a single multimedia stream into 2 or more sub-streams.
Quality of Service (QoS)	The network ability to provide the end user with maximum availability and minimum delay.

Real-time multimedia	Digital Audio/Video that is considered obsolete if not delivered on certain time (deadline).
Session	One or more flows which carry a single real-time multimedia stream between sender and receiver and necessary control packets.
Sink	A more powerful node in terms of power, memory, and computational capabilities that other sensors in the WSN usually send data to.
Thinning and Striping	Traffic shaping technique used to create a new media stream by picking some of the elements of the original stream in specific order.
Throughput	A rate which a computer or node are using to send or receive data through a channel of communication link

VITA AUCTORIS

NAME:	ANISS M. ZAKARIA
D.O.B:	June 29, 1975
PLACE OF BITH:	Lebanon
EDUCATION:	M.Sc. Computer Science University of Windsor Windsor, ON. Canada 2008
	B.Sc. Computer Science Ajman University of Science and Technology Ajman, United Arab Emirates 1997