NTNU
Norwegian University of
Science and Technology

Energy Efficiency of Streaming over Mobile Ad-hoc Networks

Prashanth Pattabiraman

Master in Security and Mobile Computing
Submission date: June 2010
Supervisor: Øivind Kure, ITEM

Norwegian University of Science and Technology
Department of Telematics

Problem Description

Mobile Ad hoc Networks (MANET) are infra-structureless networks where mobile nodes cooperatively form a network. Enabling media streaming services over such networks could be very useful, for instance, in emergency and rescue scenarios where the communication infrastructure might have been (partially) destroyed or might not exist at all. There is relatively little previous research work that has focused on enabling media streaming services in such scenarios.

The objective of this thesis is to investigate the energy consumption and performance of routing protocols by performing measurements on a real world MANET testbed and by using simulations. The final outcome of the work will provide quantitative and qualitative results on the energy efficiency of routing protocols when used for the purpose of streaming in a mobile ad-hoc network.

Assignment given: 26. January 2010
Supervisor: Øivind Kure, ITEM

Aalto University
School of Science and Technology
Faculty of Information and Natural Sciences
Degree programme of Nordic Security and Mobile Computing

Prashanth Pattabiraman

Energy Efficiency of Streaming over Mobile Ad-Hoc Networks

Master's Thesis
Espoo, June, 2010

Supervisor: Professor Antti Yla-Jaaski, Aalto University
Supervisor: Professor Oivind Kure, Norwegian University of Science & Technology
Instructor: Matti Siekkinen M.Sc. (Tech.), Aalto University

Abstract

Hand held mobile devices are widely used today primarily due to their rich functionality and the ease of portability. However, the battery life of these devices is very limited and deploying resource hungry applications such as streaming on these mobile devices is a challenging task. It is extremely important to maximize the efficient use of the contained resources on these devices especially when they participate in a mobile ad hoc network. The optimization can occur in any layer of the OSI stack, however, this thesis work focuses only on the routing protocols used in the network layer.

In this thesis work we have been able to evaluate the Energy Efficiency of the four most widely used MANET routing protocols (AODV, OLSR, DSDV and DSR) in terms of their energy consumption and performance. The initial phase of the work was carried out using the Network Simulator 2(NS2) tool and later the observations were done on a real world MANET testbed. The influence of several external factors on the performance and energy consumption are also taken into consideration while performing the simulations and experiments. The results obtained from our observations provide both qualitative and quantitative analysis of the routing protocols. Furthermore, it also highlights how the behavior of the protocols are sometimes highly unpredictable, yielding results that we may not expect.

Keywords: MANET, Network Simulator 2, Routing Protocols, Energy Efficiency, Perforamce Analysis, Hardware Performance Counters, Power modelling

Language: English

Acknowledgements

I would like to express my sincere thanks to Prof. Antti Yla-Jaaski from Aalto University, School of Science and Technology, for offering me this challenging research topic and for his valuable guidance during this thesis work. I'm also grateful to him for believing in my potential and for offering me financial support during this period.

I would also like to express my gratitude to Professor Oivind Kure from Norwegian University of Science and Technology (NTNU), whose support and suggestions were of great help while obtaining results using Network Simulation tools.

I'm deeply indebted to my instructor Matti Siekkinen for his encouragement, support and guidance. The interesting discussions that I had with him helped me to understand the thesis problem well. This deep understanding influenced my work and aided in obtaining meaningful results.

I would like to express my gratitude towards my family for their moral support and to GOD for giving me this opportunity to study and the strength to overcome obstacles that were in my way.

Espoo June 26th 2010

Prashanth Pattabiraman

Abbreviations and Acronyms

MANET	Mobile Ad-hoc Network
NS2	Network Simulator 2
AODV	Ad-hoc On-Demand Distance Vector
DSR	Dynamic Source Routing
DSDV	Destination Sequenced Distance Vector
OLSR	Optimized Link State Routing
TORA	Temporally-Ordered Routing Algorithm
DYMO	Dynamic Manet On-Demand
RREQ	Route Request
RREP	Route Reply
RERR	Route Error
MPR	Multipoint Relays
OTcl	Object Tcl
RWP	Random Waypoint
GOD	General Operations Director
CBR	Constant Bitrate
MAC	Medium Access Control
DCF	Distributed Coordination Function
RTS	Request to Send
CTS	Clear to Send
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
IFq	Interface priority queue
PDF	Packet Delivery Fraction
NRL	Normalized Routing Load
MTU	Maximum Transmission Unit
WNI	Wireless Network Interface
PSM	Power Saving Mode
CAM	Continuously Active Mode
SSH	Secure Shell
UPnP	Universal Plug and Play
HPC	Hardware Performance Counters

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

Over the last decade the use of mobile devices have increased humongously. It is now the case that a significant proportion of the human population in developing and developed countries have some kind of a mobile device with them. These devices that we frequently refer to as "nodes" in the context of MANETs, can be mobile phones, laptops, personal digital assistance (PDA) or any other personal device that has a wireless interface card and is capable of participating in a wireless network. A mobile device supports several wireless communication technologies that exist today, such as 802.11 WLAN, 4G, 3G, Bluetooth, etc and a device may choose the appropriate interface to communicate in a particular scenario. Kwon et al. referred to this as "wireless technologies convergence" [25]. With this increase in the number of mobile devices, the need to use conventional fixed infrastructure networks for communication has been challenged. The shift has been in the favor of using dynamic networks called MANETs in establishing connectivity between nodes.

MANET is the acronym for Mobile Ad-Hoc Networks. MANETs are an autonomous collection of mobile nodes that dynamically create a wireless network among themselves. They do not require any dedicated infrastructure or administrative support. They are self-creating, self-organizing and self-administering. The wireless communication devices are transmitters, receivers and smart antennas. Each node within a MANET is free to move in any arbitrary direction with any arbitrary speed. These nodes may be present in vehicles or may be carried in hand by an individual. Either ways, these nodes are capable of discovering other nodes in their vicinity and forming arbitrary topologies by connecting with these nodes.

A node in a MANET not just sends or receives its own data, but also participates in forwarding data intended for other nodes. The versatility of MANETs makes them best suited for certain scenarios such as battlefields, disaster hit areas, meetings, etc. Since the nodes are highly capable of configuring themselves, the network connections can be established without any hassle and in very short period of time. In addition, MANETs are highly dynamic and spontaneous networks. Nodes can join and leave the network at any point in time without any notification. Hence they are suitable for cases where a set of nodes may be a part of the network only for a particular communication session and may leave the network once this session expires.

One of the major drawbacks of nodes within in a MANET is their constrained battery life. Hence the protocols designed for use in MANETs must consider energy efficiency as one of its primary design criteria. This is especially vital in the case of deploying a resource hungry application such as streaming in MANETs. Applications such as streaming and many others put heavy burden on the limited resource availability of the mobile nodes. The Internet Engineering Task Force's (IETF) has a MANET working group (WG) that focuses on designing and developing IP routing protocol technologies which improve mobile routing and interface definition standards within the IP protocol suite. There are several routing protocols that have been designed specially for MANETs. Extensive research work is carried out to study some of the most commonly used protocols such as Ad hoc On-demand Distance Vector (AODV), Dynamic Source Routing (DSR), Temporarily Ordered Routing Algorithm (TORA) and Optimized Link State Routing(OLSR). The energy optimization of routing protocols designed for MANETs can be performed at any layer of the OSI stack. However, recent research works have focused on cross-layer designs. Using this approach, information can be shared between the various protocol layers in order to achieve higher power conservation.

It might be hard at times to quantitatively classify one routing protocol as better than another since the performance and energy consumption metrics vary with the scenario and application that one considers during the evaluation process. However, irrespective of which routing protocol is employed in a MANET, it may still be important to understand the behavior, the performance, the energy consumption and the parameters that influence these three metrics, in order to get the maximum out of our nodes and network.

1.2 Problem Description

Streaming over MANETs is one of those topics that haven't received much focus in the research world. Majority of the research work in the MANET area has been around developing routing protocols and observing the performance of these protocols. However the evaluation of energy efficiency has always been given a low priority during the performance evaluation process. Furthermore, most evaluation processes involve the use of scenarios that were modeled using network simulation tools such as NS2 and OMNet++. The use of real world MANET testbeds in studying routing protocols has been very minimal due to several factors such as cost and complexity of setup, difficulty in monitoring critical parameters, efficiency in carrying out the experiment, difficulty in reproducing the results, challenges in introducing obstacles and randomness in the experiments, etc.

One of the important applications of streaming over a MANET is in a disaster recovery operation. In a disaster hit area, the communication infrastructure may be damaged or absent and it may be vital to establish a temporary network that assists the rescue workers during their rescue operation. Such a network would aid in facilitating communication and cooperation between the various emergency teams involved in the rescue operation. Mobile devices that are carried by the rescue personnel may be used to stream live video captured through a cam, to a central server. This live stream can be used to timely dispatch medical assistance and supplies to the right areas and people who need them the most. It may also help is assessing the damage to property, infrastructure and machinery due to the disaster. The application of streaming over a MANET is not confined to a rescue operation and may span many other application areas such as battlefields, entertainment purposes, location-based services, etc.

Mobile Ad-Hoc networks offer a number of benefits and are suitable for a wide range of scenarios. Their ease of setup and low cost of operation makes them an attractive option on several occasions. However, establishing these networks also have a lot of challenges. One of the major challenges with using MANETs is that the nodes usually have limited memory, processing capabilities and battery power. This makes deploying applications such as streaming even harder on these networks. A node that dies due to low battery power can cause an existing route or link to break. In the worst case, it might also lead to network partitioning and may render some nodes unreachable. These are situations that are not desirable in a network that is primarily used for streaming and especially in a critical rescue operation. Hence the focus of this thesis work is on understanding the energy efficiency of some of the commonly used routing protocols in a mobile ad-hoc network. Energy efficiency has been considered by taking two different aspects of a routing protocol, namely the performance of the protocol and the energy consumption of the protocol. The effect of certain factors such as network load, packet size, mobility rate, varying traffic patterns, etc. that influence the performance and energy consumption of the routing protocol have also been studied. The energy efficiency of streaming was evaluated using both a simulated environment as well as a simple real world testbed. The results obtained in this work will help us in choosing the right routing protocol for our scenario and application.

1.3 Scope of Research Work

In this thesis work we use both simulation tools as well as a simple real world testbed to analyze the performance and energy consumption of MANET routing protocols. We have considered two reactive protocols (AODV and DSR) and two proactive protocols (OLSR and DSDV) for our evaluation. The NS2 (Network Simulator) tool was used to model the MANET. The simulation trace files that were generated by the NS2 were parsed to extract the necessary data. Our choice of simulation tool for the evaluation process was influenced by the fact that 95% of all simulation experiments in the field of wireless communication were carried out using the NS2. Our choice of protocols for the evaluation process were greatly constrained by the protocol support available for the network simulator tool at the time of this thesis work. Performance of the protocols has been evaluated using metrics such as throughput, normalized routing load, packet delivery fraction, etc. The total energy consumption in the network for each protocol and the energy expensed per byte of data are some of the energy metrics that are presented to understand the behavior and characteristics of these routing protocols. Our work also explores certain factors such as packet size, mobility rate, transmission rate and network load, and their effect on the performance as well as on the energy consumption of the protocols. The simple real world testbed comprises of a laptop that is used as the streaming server and a Nokia N810 tablet device that is used as the mobile client. Both devices are configured to be a part of the same ad-hoc network and has OLSR protocol driving it. The energy consumption of the mobile device for a time period has been evaluated under four cases, namely, when the node is idle, when the node has OLSR daemon running and maintaining the routing table, when the node is receiving an audio stream and when the node is receiving a video stream.

Although achieving a high energy efficiency requires understanding the protocols used in every layer of the OSI stack, our thesis work focuses only on the protocols used in the network layer. Furthermore, in our simulations we have eliminated the option of choosing between TCP (Transmission Control Protocol) and UDP (User datagram protocol) in the transport layer and we have simply stuck to using UDP throughout all our experiments. A previous study has shown UDP to be more energy efficient compared to TCP since they do not employ acknowledgement messages that increase the overhead and the energy consumption. All nodes in the simulation topography are assumed to be operating in Continuously active mode (CAM) since it is impossible for a node to predict when its participation is required in the network. This results in the node being only in three of the four possible states, namely Idle, Transmit and Receive. The nodes do not have the luxury of entering the Sleep state which can greatly aid in energy conservation.

1.4 Document Structure

Introduction: This chapter gives a brief description on Mobile Ad-hoc Netwoeks (MANETs) and the advantages and disadvanteges of using these networks. It introduces the problem that this thesis work tries to address, which is the energy efficiency of streaming over mobile ad hoc networks. It also clearly defines the scope of the research work.

Background: This chapter provides some background informantion on MANET routing protocols. It also presents results of some previous work that has been done to evaluate the energy efficiency of protocols that operate in the various layers of the OSI stack.

Simulation Setup: This chapter gives an overview of the Netwrok Simulator (NS2) tool that we use in our thesis work. It also describes the traffic patterns, the mobility patterns and the process of generating the files that introduce these patterns in our simulations.

Performance Evaluation of Routing Protocols: This chapter presents the results of our observation of the performance of four commonly used MANET routing protocols, namely OLSR, DSR, DSDV and AODV. The performance is represented in terms of several performance metrics such as throughput, normalized routing load, packet delivery fraction, etc.

Energy Consumption of Routing Protocols: This chapter presents the results of our evaluation of the energy consumption of the routing protocols considered. It also presents the power models that were considered for evaluating the energy consumption during the simulation period.

Power Consumption on a real world MANET testbed: In this chapter we describe the testbed that we setup and present our results on the energy consumption of a hand held mobile device (Nokia N810 tablet) while streaming audio or video data. We compare these energy consumption values with the energy consumed while the device is idle in order to understand the impact of streaming on the battery life.

Conclusion and Future Work: This chapter summarises the thesis work and suggests the future direction in which the work could be carried forward.

Appendix A provides the configurations and setups that need to be done on the mobile device in order to establish our real world MANET testbed. Appendix B provides some of the scripts that were used on the NS2 simulator for evaluating the performance and the energy consumption of our protocols.

Chapter 2

Background

2.1 MANET Routing Protocols

The routing protocols used in MANET can be classified as either table-driven or on-demand driven. In a table-driven routing protocol, each node maintains a consistent table containing up-to-date routing information to every other node in the network [38]. This is achieved by each node updating its own routing table and forwarding the update to its neighboring nodes. Whereas in a on-demand driven routing protocol, the path to a destination is not known before hand[38]. When data is to be sent to a destination, the source performs route discovery by sending a route-request and waits for a route-reply packet back. Once a route is established, it is maintained in a route-cache present in each node along the path. An energy aware routing protocol uses energy related metrics such as energy consumed/packet, cost/packet, maximum node cost, time to network partition and variance in power node to choose the most efficient route that will maximize the lifetime of the network. The cost per packet metric is similar to the energy per packet metric but it also accounts for the residual energy left in a node in addition to the transmission energy required to transmit the packet to the next hop. An energy aware routing protocol will prefer a link that requires low transmission energy, but at the same time avoids nodes that have low residual power in them since the node cost of such nodes is considered high. A path chosen using the above principle is called a min-max path, referring to minimum transmission energy and maximum residual energy. NS2 currently does not have support for any of the power aware routing protocols and hence only a certain set of commonly used MANET routing protocols are considered for our evaluation.

2.1.1 Ad-Hoc On-Demand Distance Vector (AODV)

Ad-Hoc On-Demand Distance Vector (AODV)[33] Routing Protocol is a multihop routing protocol employed in an ad-hoc environment where adapting to dynamic topological changes, occurring as a result of frequent mobility, is essential. When a route to a destination node from a certain source node is not known, the source node sends out a Route Request (RREQs)message towards the destination node. Every node that receives the route request message caches a route back to the originator of the request. A route can be formed if this message reaches either the destination node itself or any other intermediate node which has a fresh enough route to the destination node. The freshness of the route information is determined using the destination sequence number associated with it. The route information contained in the intermediate node must have a destination sequence number that is atleast as great as that contained in the RREQ message inorder to be accepted as a valid route entry. Once a route is determined, the route information is sent as a Route Reply (RREPs) message to the requesting node. Since every intermediate node that received the route request has cached a route back to the requesting node, the RREPs can be unicasted from the destination node towards the node that requested the route. Route Error messages (RERRs) are used by nodes to inform neighboring nodes about the unreachability of certain destinations as a result of broken links along the active path to that destination. When links break and active routes become invalid, the corresponding route entries contained in the routing table of the nodes forming the route are deemed stale and are invalidated.

2.1.2 Dynamic Source Routing protocol (DSR)

Dynamic Source Routing protocol (DSR) [22] is a simple and a completely on-demand routing protocol. It does not flood the network with periodic routing advertisements, link status sensing or neighbor detection packets. Hence the routing overhead is scaled to only those needed to react to a change in a route that is currently active. DSR is best suited for a mobile ad-hoc environment where there are at most 200 mobile nodes. It also performs well with nodes having very high mobility rates. DSR supports multiple routes to a destination. This offers the source the flexibility of choosing a suitable route for the purpose of load balancing and to obtain increased robustness. There are two important mechanisms that DSR is composed of: Route Discovery and Route Maintenance. Route Discovery is the mechanism through which a source node wishing to send data to a destination node obtains a source route to that destination. Route Discovery happens only when the source node does not have knowledge of any pre-existing route to the destination. Route Maintenance is the mechanism through which a source node detects the breakage of a link along the route to a destination due to a topology change. If an alternate source route is available then the source node would use that. In the absence of an alternative route to the destination the source node performs Route Discovery again.

2.1.3 Destination Sequenced Distance Vector protocol (DSDV)

Destination Sequenced Distance Vector protocol (DSDV) [34] is based on the Bellman-Ford algorithm [9] with certain improvements that address the poor looping properties of the algorithm. Every mobile node has a routing table that contains the number of hops to all available destinations within the ad-hoc network. Each routing table entry has a sequence number associated with. This sequence number helps to identify stale routes and to avoid formation of loops. The mobile nodes periodically transmit updates to their immediate neighbors. These updates carry information on which destinations are reachable from each mobile node and the number of hops required to reach them. An update is also sent when major topological changes occur. Hence the update packets are both time-driven and event-driven. The updates from a mobile node can contain either the entire routing table or just those entries for which there has been a metric change since the last update that was advertised. The former is known as a 'full dump' while the latter is known as an 'incremental update'. Full dumps are best suited for an ad-hoc network where the topology changes very frequently. Using incremental updates in such cases would lead to a large growth in the number of incremental packets that hog the network bandwidth. In the case of an ad-hoc network that has very few topology changes, the use of full dumps is very frequent. When a mobile node receives a routing information which was advertised by its neighboring node, it compares the information received with the routing information that it has locally in its routing table. If it finds a route entry which has a more recent sequence number then that route is used. Entries with older sequence numbers are invalidates. If a route in the advertisement has the same sequence number as an existing route, then the route with a better metric is chosen. The newly recorded routes are later scheduled to be advertised periodically.

2.1.4 Optimized Link State Routing protocol (OLSR)

Optimized Link State Routing protocol (OLSR) [14] is a proactive and table driven routing protocol that is best suited for a large and dense mobile ad-hoc network where a large subset of the nodes are communicating with another large subset of nodes. It performs a hop-by-hop routing where each node uses the routing table available locally to route the packets. Every node selects a set of its neighboring nodes as multipoint relays (MPR). Only nodes that are selected as MPRs can forward broadcast messages during the flooding process. This helps in scaling the control traffic by reducing the number of transmissions required. With these control packets a node informs the network that it has reachability to the nodes that have chosen it as their MPR. The optimization offered by OLSR can be observed clearly in a large and dense network as compared to a sparse network. Since OLSR is a proactive routing protocol, it has a route readily available when data is to be sent to a destination.

2.1.5 Temporally-Ordered Routing Protocol (TORA)

Temporally-Ordered Routing Protocol (TORA) [32] is a distributed routing protocol, hence each node in the mobile ad-hoc network has a logically separate copy of TORA running for each destination within the network. TORA is based on a 'link reversal' algorithm and route discovery is performed on-demand when data is to be sent to a destination. TORA allows multiple routes to a destination and it reduces the communication overhead arising due to a topology change by localizing the algorithmic reaction to these changes. When a node needs to discover a route to a destination, it simply broadcasts a QUERY packet containing the address of the destination to all the neighboring nodes. This packet propagates through the network until it reaches the final destination node or until it reaches another intermediate node that knows the route to the destination. The recipient of the QUERY packet then broadcasts an UPDATE packets that specifies its height with respect to the destination node. Every node that receives the UPDATE packet will now set its height to the destination node to a value greater than that specified by the UPDATE packet received from the neighboring node. Through this mechanism it is possible to create directional links from the node that broadcast the QUERY packet towards the node that first broadcast the UPDATE packet. In a mobile ad-hoc network, network partitioning happens quite often as a result of mobility. When such a network partitioning is detected by a node, it sends out a CLEAR packet that clears out any invalid route entries in the routing table and resets the routing state. In case a route to a destination becomes invalid at some point in time then the node sets its height to the destination to a local maximum with respect to its neighboring nodes. It then sends out an UPDATE packet. If a node does not have neighbors with a finite height to the destination node, it performs the route discovery as discussed above. By using TORA a node would rather use an existing route to a destination which may not be optimal than use the shortest path that may be discovered with some additional overhead that comes with discovering newer routes.

2.1.6 Dynamic MANET On-Demand Routing protocol (DYMO)

Dynamic MANET On-Demand (DYMO) [12] Routing protocol is a multihop unicast routing protocol that is comprised of two basic operations, namely, route discovery and route maintenance. These operations are similar to those that happen in AODV routing protocol. DYMO is suitable for a network that has a large number of routers with only sparse traffic present in the network. It can also be used in a network where the mobile devices have a memory constraint because DYMO stores very minimal routing information in each of the DYMO routers. The routing information that is stored is very minimal compared to other proactive routing protocols because DYMO stores routing information only for routes that are currently active. Whereas proactive routing protocols store routing information to all routers that are within the routing region.

Note: Network Simulator 2 (NS2) [31] has built in support for DSR, AODV, DSDV and TORA. However, it does not support OLSR or DYMO out of the box. UM-OLSR and DYMOUM are implementations of OLSR and DYMO protocol respectively for the NS2 network simulator and are available under the GNU General Public License (GPL). UM-OLSR [36] supports all the functionalities of OLSR as stated in RFC 3626 and DYMOUM [35] complies with the DYMO drafts -04 and -05. Both implementation have been successfully tested on NS2 prior to making them publicly available. TORA and DYMO have not been considered for our evaluation. TORA has been ignored due to the lack of sufficient documentation on its deployment and use in NS2, while DYMO has been ignored due to the inability to patch NS2 with DYMOUM patch after a prior patching with UM-OLSR. Please refer to the section on NS2 installation and Patching for more information on the steps taken to get NS2 up and running.

2.2 Energy Efficiency at various layers of the OSI stack

To achieve the maximum energy efficiency while streaming over mobile adhoc networks, the right choice of protocol must be made at each layer of the OSI stack. This combination of protocols operating at each layer will greatly influence the total energy conserved during the streaming process. Very little research has been done in evaluating the efficiency of such protocol combinations as a whole and much of the effort has been on evaluating the efficiency of protocols in each layer individually.

One of the main challenges in streaming over MANETs is the choice of video codec used. The video codec employed must be capable of adapting the bit rate at run time, taking into consideration the energy level of the mobile nodes and the network's current condition. The trade off between quality and bit rate should also ensure that a minimum acceptable quality is always provided to the end user. A video codec such as H.264/AVC [46, 5], also referred to as MPEG-4, has proved to increase coding efficiency by almost 50% compared to its predecessor the MPEG-2 and H.26X standards. They also offer more flexibility in coding by letting groups of macroblocks to be coded independently. However, this flexibility comes at the cost of needing complex decoders and information feedback from protocols operating in the other layers of the stack.

In the transport layer, the study of energy efficiency of three variants of TCP: Reno, Newreno and SACK (Selective Acknowledgements) indicate that SACK outperforms both Reno and Newreno when it comes to total energy (E) consumed [40]. The total energy E refers to the energy consumed by a system from the beginning of the data transmission to its end and this total energy is inversely proportional to the throughput of the protocol used. However, when the idealized energy (E_I) consumption [Idealized Energy=Total Energy-Idle Energy] is considered, SACK performs poorly. This is because of the fact that SACK has additional data structures and complex computations involved at the sender's end.

The idle energy (E_{Idle}) consumed by a system is the energy expended by the sender when it waits for ACKs before continuing to transmit more data packets. Currently SACK consumes much lower idle power than Reno and NewReno. If we can design the wireless devices to either doze off or enter deep power saving modes when there is no activity, we can succeed in reducing the idle power consumption greatly. This would make Reno and Newreno more energy efficient than SACK on an overall scale [40]. However, if the idle power consumption remains high then SACK would definitely be the better alternative to Reno and Newreno by providing higher energy savings. Furthermore, the observations made during the study indicate that packets with smaller MTUs consume less total energy in scenarios where packet losses are high. Large MTUs are suitable for low packet loss scenarios. In most cases, SACK completes data transmission earlier compared to Reno and Newreno and thus consumes lower overall energy and provides higher throughput.

In [20] the authors evaluated the power behavior of TCP and UDP under multimedia-like streaming conditions. The test bed consisted of 49 wireless stations in a 7x7 grid topology. Each node's transmission range only covered the nodes that were horizontally and vertically adjacent to it. The bandwidth of the wireless medium was 2Mbps, two different data traffic flows (5 and 10 data flow) used, the packet size was 512 bytes and the transmission rate was set to 300 kbps for each flow. Studying the distribution of the power consumption for both TCP and UDP under the given conditions shows that TCP dissipates higher amounts of energy from its nodes compared to when UDP is employed.

In MANETs, a node that suffers power failure may hamper the reachability of certain other nodes and can also cause those nodes to be left completely disconnected from the network. Hence a lot of effort is put into developing energy aware routing protocols. These energy efficient routing protocols optimize either the active communication energy required to transmit and receive data packets or the inactive energy consumed when a node remains idle but listens to the wireless medium for possible incoming communication requests from other mobile nodes. An energy aware routing protocol uses energy related metrics[42] such as energy consumed/packet, cost/packet, maximum node cost, time to network partition and variance in power node to choose the most efficient route that will maximize the lifetime of the network. The cost per packet metric is similar to the energy per packet metric but it also accounts for the residual energy left in a node in addition to the transmission energy required to transmit the packet to the next hop. An energy aware routing protocol will prefer a link that requires low transmission energy, but at the same time avoids nodes that have low residual power in them since the node cost of such nodes is considered high. A path chosen using the above principle is called a min-max path, referring to minimum transmission energy and maximum residual energy.

The "Transmission power control" approach is one way of minimizing the active communication energy required. The idea behind this approach is to chose an optimal routing path that minimizes the total transmission energy required to deliver a data packet from the source to its destination. Some protocols developed using this approach are Flow Augmentation Routing (FAR) Protocol [21], Online Max-Min Routing Protocol [27], Power-aware Localized Routing (PLR) Protocol [43] and Minimum Energy Routing (MER) Protocol [15].

The "Load distribution" approach is another way of reducing the active communication energy. Its main focus is not to reduce the energy consumption of individual nodes but to balance the energy usage among all nodes in order to increase the longevity of the network lifetime by avoiding over-utilized nodes when selecting a routing path. This will ensure that certain nodes that have low energy left may not be necessarily chosen even if they may lead to the formation of the shortest path to the destination, taking into consideration the high path cost incurred by selecting a path with such a node. Localized Energy Aware Routing Protocol [47] and Conditional Max-Min Battery Capacity (CMMBCR) Protocol [45] are protocols built on this principle.

The "Sleep/power-down mode" approach helps to minimize the inactive energy consumption by either switching off the node or switching to deep power save mode when there is no activity. This approach must however guarantee that the data will be delivered to the desired node even when most of the nodes sleep and do not forward packets to other nodes. SPAN Protocol [13], Geographic Adaptive Fidelity (GAF) Protocol [50] and Protocol Embedded Network (PEN) Protocol [18] help conserve a node's critical battery power by pushing them into these deep power saving modes as frequently as possible.

Each routing protocol has its own advantages and disadvantages. No single protocol can cater to all our needs. The solution is to combine and integrate the available routing protocols to create a more energy efficient routing protocol. A cross-layer approach is also required for creating energy efficient routing protocols by exploiting the information available at other layers.

In [11] the energy consumption of four protocols, namely On Demand Distance Vector (AODV), the Direct Source Routing (DSR), the Temporally-Ordered Routing Algorithm (TORA) and the Destination-Sequenced Distance-Vector Routing (DSDV) are compared under extremely low network load where each nodes send only four 512 bytes packets per second. For the basic scenario, the energy consumption due to reception was observed to account for a larger percentile of the total energy consumption as compared to the energy required for the transmission. The receiving process includes both reception of actual data packets and reception of neighbor's data (overhearing). DSR consumes the least energy for routing protocol packets among the four protocols while TORA has the highest. DSDV consumed the least energy in terms of transmitting the CBR data. In the case of varying motion pattern, TORA has the highest routing energy consumption while DSR has the best performance. On-demand protocols show a decrease in the routing energy consumption as the motion rate of the nodes drop and the table driven protocol (DSDV) has a constant routing energy consumption irrespective of the change in the node motion rates. As speed of the nodes grow, DSDV performs better than AODV. Furthermore, when traffic sources vary from 10 to 20, the routing energy increases 7.31% in DSR, 88.97% in AODV and 4.71% with TORA. While moving from 20 to 30 sources the routing energy increases 41.73% in DSR, 15.88% in AODV and 13.37% with TORA.When the number of nodes increases from 25 to 50; TORA has a massive increase in its routing energy consumption while the increase in the remaining protocols is comparatively less.

In [51] a simulation environment consisting of 49 wireless nodes randomly distributed over a 150m x 150m area was used to evaluate the energy efficiency of ECSRP (Energy-Aware Candidate Set Routing Protocol) in comparison with CMMBCR (Conditional Min-Max Battery Cost Routing)[45] under the Gilbert error model which generates loss events in each receiving channel. ECSRP chooses a route from the candidate subset in the route cache in which all the intermediate nodes along the route have sufficient residual battery power. This ensures that certain routes are not over utilized. ECSRP also takes channel condition into consideration by employing the packet loss probability in the computation of energy consumption. This helps to reduce retransmissions arising from packet loss associated with poor channel conditions and thus helps to save energy. The results obtained from the simulation of 50 randomly generated instances of the topology are used to study the residual battery power of nodes and the node death speed. The node death speed traces how many nodes are still alive after certain duration of time. Lower node death speed is an indication of greater longevity of the network due to load balancing. It was observed that the average residual battery power of ECSRP is much higher than that of CMMBCR. In addition, ECSRP has lower node death speed compared to CMMBCR which indicates that ECSRP has better load balancing that prolongs the lifetime of individual wireless nodes.

In [44] two novel protocols are proposed: MTRP (multi threshold routing protocol) and EMTRP (enhanced multi-threshold routing protocol). MTRP divides the total energy of a wireless node into multiple ranges. The routes are classified according to their bottleneck energy i.e. the minimum residual energy among the nodes traversed in the route. The lower bound of each range is called a threshold and the protocol iterates from the highest threshold to the lowest one in order to choose a route which has bottleneck energy higher than the current range's threshold limit. This ensures that there is load balancing and that certain routes are not over utilized. If no route satisfies the current threshold, then the current threshold is lowered and the search for a route is carried out again. In the case where there are multiple routes that have bottleneck energy greater than the threshold, the protocol choses the route that has the smallest hop count. This further improves the energy efficiency of the protocol. EMTRP is based on MTRP and it takes channel conditions also into consideration while selecting routes. Choosing a route with better channel conditions will contribute to lower packet loss and thereby reduce the number of retransmission required. This helps in conserving energy. The results obtained from fifty simulations, conducted using 49 wireless nodes distributed randomly over an area of 130m x 130m area, indicate that the two proposed algorithms have lower overhead compared to CMMBCR. This is due to the fact that the algorithms only consider the residual energy information while choosing a route and the distance-related information is ignored. The residual energy of EMTRP and CMMBCR in both models are comparable since even CMMBCR achieves energy efficiency by choosing the route that requires the lowest transmission energy. However, in the case of Gilbert error model the residual energy of EMTRP is slightly higher than CMMBCR. The node death speed is much lower when EMTRP is used as compared to CMMBCR. When the simulation ended there were 25 nodes that were still alive in EMTRP and only 17 nodes were alive in CMMBCR.

In [29] a new MAC protocol is proposed that combines the mechanism of power control, RTS/CTS dialogue and busy tones. The objective of the protocol is to constrain the power level with which a mobile node transmits data packets. This is achieved by determining the distance between the two communicating nodes by measuring the signal strength on which the RTS and CTS packets are received. The benefits of using lower power include increased channel reuse, increased channel utilization, reduced co-channel interference and saving of precious battery life of the mobile nodes. A simulation was carried out in an area of size 500 x 500 with 200-1800 randomly generated sender-receiver pairs. The number of communication pairs that were granted in the area by using the two models (power control model proposed and the non-power control based model) were studied. This can also be interpreted as the amount of channel reuse that can be offered with or without the power model. The results indicate that the power control model can offer 1.5 times that of the communication pairs offered without the power control. Furthermore, the proposed power control MAC protocol has better channel utilization compared to DBTMA (dual busy tone multiple access) under various loads.

[28] proposes two different MAC protocols namely, Multi-Rate Medium Access Control (MR-MAC) operating in the 802.11g environment and Cooperative Multi-Rate MAC protocol (CMR-MAC). MR-MAC economizes on the energy for low traffic scenarios and offers high throughput in the case of high traffic scenarios. The MR-MAC consists of three parts, namely, channel utilization estimation, transmission rate selection and transmit power calculation. CMR-MAC adds an energy helping module to existing three modules of MR-MAC. This module is used by a node to periodically exchange energy situations with its neighbors. It also adjusts its transmission rate correspondingly to accomplish the energy consumption balance in a local area. The results from simulations indicate that MR-MAC outperforms Receiver-Based Auto-Tune (RBAR) by 40% in terms of energy efficiency, yet maintains a comparable throughput with the latter. RBAR uses RTS/CTS to probe channel conditions and to select rates at a receiver end based on the signal to noise ratio (SNR) and a series of reception thresholds. MR-MAC's energy efficiency was also higher than CMR-MAC's energy efficiency by 20%. In the case of low traffic the energy efficiencies of MR-MAC and EMR (En-

ergy Efficient Multi-Rate) are comparable. However, MR-MAC outperforms EMR at high traffic loads. CMR-MAC reduces energy consumption rate by 50% compared with RBAR via its cooperation mechanism. CMR-MAC is 20% and 30% better than MR-MAC in terms of network lifetime and number of delivered packets respectively.

[39] proposes the Energy-Efficient MAC protocol (EE-MAC). In EE-MAC, master nodes are elected from all nodes present in the network. These node form the virtual backbone of the network and are responsible for routing packets on behalf of their slave nodes. These nodes must stay awake at all times and a fair rotation mechanism is used to ensure that all nodes share the job of providing global connectivity equally. Simulations were carried out using NS2 to evaluate the performance of EE-MAC and compare the results with that of 802.11 MAC protocol without power saving and the 802.11 PSM. The results indicate that as the network load increases all three protocols perform poorly due to higher collision rate. Under heavy traffic load, the performance degradation in EE-MAC is due to the frequent running of the master election algorithm. In terms of energy efficiency, EE-MAC performs best among all protocols. This is because EE-MAC can tell slaves to enter sleep mode once they have received all data that is addressed to them. However, if there are many source nodes that intend to send data in the network then most nodes that comprise the network may have to stay awake. This destroys the energy saving mechanism employed in EE-MAC and its performance might be lower than 802.11 in these cases. At higher node densities, EE-MAC outperforms the other protocols.

Chapter 3

Simulation Setup

3.1 NS-2 Overview

NS is a discrete event simulator which began as a variant of the REAL network simulator in 1989 [31] . It has evolved ever since in its functionality and features and is widely used in networking research. It is an open source software and is distributed for free under the General Public License (GPL). It supports the simulation of routing protocols in a wired as well as a wireless network. It is currently supported by DARPA with SAMAN and NSF with CONSER. Several researchers have contributed towards the development of NS, including wireless code from the UCB Daedelus and CMU Monarch project and Sun Microsystems. NS2 is written in C++ and for setting up the simulation environment a script language called OTcl (Object Tcl) is used.

There are a few vital components that make up any network simulation. We employ OTcl [31] script to provide the network configuration, a mobility pattern that describes the movement of nodes within the topography during the simulation and a traffic pattern that describes the data flows between nodes. Now when a simulation is run, the trace of the node movement and the data flow is output into a file that can be used for post-simulation analysis. The post-simulation analysis usually involves parsing the trace file using a script in order to obtain the necessary results.

Normally, two different kinds of trace files can be generated depending on the user's requirements. A NAM trace file which has the extension '.nam' can be used as an input by any NS2 compliant animator tool to visually depict the node movement and data flow activity during the simulation. A simple trace file which has the extension '.tr' can be parsed using a script to extract useful information. For our simulations we have used the awk script to extract the necessary information from the trace files.

3.2 Computer Environment

For our simulations, NS2 was installed on a Ubuntu 9.10 - Karmic Koala (Linux) Operating system. NS2 can also be installed in a Windows environment, but it requires the installation of a Unix emulator such as Cygwin before the installation of NS2. The Linux environment is usually the preferred environment because of instability issues with the software in a Windows environment. In addition, most of the patches and codes contributed for extending the software's functionality are either available only for linux or are not supported well by the Windows environment.

The trace files generated while running the simulations are large and hence a lot of hard disk space is required to store them until they are parsed and relevant information is extracted from them. A secondary partition of about 80 GB was used for this purpose initially. At a later stage during the thesis this allocated space wasn't sufficient and hence an external hard drive of size 640 GB was used to save all the trace files. A simulation that consists of 30 mobile nodes that move at a maximum speed of 10m/s, and runs for about 200 seconds can produce a trace file that is roughly 55MB in size. The file size has been observed to grow beyond 1GB in certain cases where heavy traffic loads are induced in the network and the simulation runs for about 800 seconds. The files can be made smaller by reducing the simulation period, however, care should be taken while reducing the simulation time since a small simulation period may give rise to inaccurate results as the time is insufficient for the convergence in network routing.

The system on which the simulations were carried out has an Intel Centrino processor that has a clock speed of 2.0 GHz and a 2 GB RAM. The version of the simulation software that we have used for our simulations is NS-2.34. The UM-OLSR and DYMOUM patches are available for this version.

3.3 Installation

The installation of NS2 requires the following packages to be present before the installation process begins.

1. **libx11-dev**
2. **libxmu-dev**
3. **libxmu-headers**
4. **libxt-dev**
5. **libtool**

If all or some of these packages are missing then install them using the following commands:

```
$ sudo apt-get install build-essential libx11-dev
$ sudo apt-get install build-essential libxmu-dev
$ sudo apt-get install build-essential libxmu-headers
$ sudo apt-get install build-essential libxt-dev
```

During the NS2 installation process an error was generated and the installation terminated. A snippet of the error report is provided below. This error is probably due to missing gcc-4.4.1 / g++-4.4.1 compilers. Even earlier versions of the compilers might be sufficient to fix the problem.

```
ld -shared -o libotcl.so otcl.o
otcl.o: In function `OTclDispatch':
/home/bogdan/ns/ns-allinone-2.34/otcl-1.13/otcl.c:495:
undefined reference to `__stack_chk_fail_local'
otcl.o: In function `Otcl_Init':

/home/bogdan/ns/ns-allinone-2.34/otcl-1.13/otcl.c:2284:
undefined reference to `__stack_chk_fail_local'
ld: libotcl.so:hidden symbol `_stack_chk_fail_local' isn't defined
ld: final link failed: Nonrepresentable section on output

make: *** [libotcl.so] Error 1
otcl-1.13 make failed! Exiting ...
See http://www.isi.edu/nsnam/ns/ns-problems.html for problems
```

Install the required compiler and then run the 'install' script as shown below. This time the installation shouldn't throw any errors.

```
$ sudo apt-get install g++-4.3
$ CC=gcc-4.3 CXX=g++-4.3 ./install
```

Once the NS2 simulator is installed, it is necessary to introduce NS2 to the system by adding the environment variables to the bash file. The path assigned to the variables have to be modified since the paths shown in the snippet below are only applicable in the authors case and may change according to the directory where the reader decides to install NS2.

```
$ gedit ~/.bashrc

# LD_LIBRARY_PATH
OTCL_LIB=/home/prashanth/Desktop/ns2/otcl-1.13
NS2_LIB=/home/prashanth/Desktop/ns2/lib
X11_LIB=/usr/X11R6/lib
USR_LOCAL_LIB=/usr/local/lib
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$OTCL_LIB:$NS2_LIB
:$X11_LIB:$USR_LOCAL_LIB

# TCL_LIBRARY
TCL_LIB=/home/prashanth/Desktop/ns2/tcl8.4.18/library
USR_LIB=/usr/lib
export TCL_LIBRARY=$TCL_LIB:$USR_LIB

# PATH
XGRAPH=/home/prashanth/Desktop/ns2/bin
:/home/prashanth/Desktop/ns2/tcl8.4.18/unix
:/home/prashanth/Desktop/ns2/tk8.4.18/unix
NS=/home/prashanth/Desktop/ns2/ns-2.34/
NAM=/home/prashanth/Desktop/ns2/nam-1.14/
PATH=$PATH:$XGRAPH:$NS:$NAM
```

After these steps, you can now run the ns validation suite to validate the installation process. The NS2 installation process is now done and the reader can execute a sample Tcl script and observe the output.

```
cd ns-2.34; ./validate
```

3.4 Mobility Model

The mobility model describes the movement of nodes within the simulation area. In our simulations we have used the Random Waypoint mobility model (RWP) [23] which was first proposed by Johnson and Maltz. They used their model in the evaluation of DSR routing protocol. In RWP mobility model the nodes move from one waypoint to the next. A specific speed and duration is chosen for every transition. After the stipulated transition duration ends the node may pause for a specific duration of time before starting its transition towards the next waypoint. This pause time is optional and may be equivalent to zero in the case of a network comprising of nodes that are under continuous mobility. A pause time that is equivalent to the duration of the simulation indicates a completely stable network that does not undergo any topological changes due to mobility. RWP, however, has one demerit when it is used for protocol evaluation. In RWP there is high probability that a node might traverse through the middle of the topography when it moves from one point to another, thus affecting the accuracy of our protocol evaluation results [52]. NS2 comes with a built in CMU tool called 'setdest' [31] that can be used to generate a large number of nodes and their movements. It is available under the ns2.34/indep-utils/cmu-scen-gen/setdest directory. Setdest generates the position of nodes during the start of the simulation, their movement speed and direction throughout the simulation.

Since in our simulation we are trying to mimic a mobile ad-hoc environment comprising of mobile nodes carried by rescue personnel in a disaster struck region, we assume that the speed of the nodes is at most equivalent to the average speed of walking which is 5.5 km/hr.

Syntax: (Syntax applicable for version 1. Refer setdest.cc for syntax for version 2)

```
./setdest [-v version] [-n num_of_nodes] [-p pause_time]
    [-M max_speed] [-t simulation_time] [-x maxx] [-y maxy]
```

Example:

```
./setdest -v 1 -n 30 -p 250 -M 1.5 -t 800 -x 1000 -y 600
```

The position of the nodes are defined at the beginning in the scenario file. The position is defined in terms of the x, y and z coordinates of the nodes. A simple extract of this from the scenario file is given below. The position of nodes 1 and 2 are assigned.

```
$node_(0) set X_ 940.331586540903
$node_(0) set Y_ 215.160392307062
$node_(0) set Z_ 0.000000000000
$node_(1) set X_ 79.469738884295
$node_(1) set Y_ 43.955083317742
$node_(1) set Z_ 0.000000000000
```

The below line from the scenario file describes that Node 0 begins to move at time 200 sec towards (723.40, 153.06) at a speed of 0.858 m/s.

```
$ns_ at 200.000000000000 "$node_(0) setdest 723.406549419407
                          153.062155525416 0.858036902610"
```

The General Operations Director (GOD) object stores information on the state of the environment, network or the nodes that an omniscent observer may have. But this information is not accessible or known to other participants in the simulation. The GOD object at present stores information on the shortest number of hops required to travel from one node to another in a multi hop route. This piece of information is not calculated by the GOD object during run time rather loaded into the god object from the scenario file. Lines that look similar to the one below provide the god object with the necessary knowledge.

```
$ns_ at 200.296933514152 "$god_ set-dist 12 27 2"
```

The god object now knows that the nodes 12 and 27 are just two hops away at time 220.296 seconds.

3.5 Traffic Model

A random traffic pattern comprising of either TCP or CBR (constant bit rate) connections can be generated using a script called 'cbrgen.tcl' [31] which can be found in the ns2.34/indep-utils/cmu-scen-gen directory. When creating a traffic pattern file, we need to specify whether the traffic contains TCP connections or CBR connections, the number of nodes present in the simulation environment, the number of connections that are to be established, the maximum number of connections that are allowed, a random seed value and the packet transmission rate in the case of a CBR based traffic pattern. The inverse value of the packet transmission rate would give us the time interval between two packets.

The default packet size is set to 512 Bytes in the cbrgen.tcl file and hence the traffic pattern generated will also carry the same packet size. This value must be modified in the cript if a different packet size is desired. The various connections generated by the script have start times that are randomly generated. However, the maximum start time is set to 180 seconds in the script. Even this default start time value can be modified in the script to suite our requirement.

Syntax:

```
ns cbrgen.tcl [-type cbr|tcp] [-nn nodes] [-seed seed]
    [-mc connections][-rate rate]
```

Example:

```
ns cbrgen.tcl -type cbr -nn 30 -seed 1.0 -mc 5 -rate 4.0>cbr5test
```

The file thus created will contain information on several connections that are started at random times. A snippet of the code describing the setup of a single connection is given below. The meaning of the code is self explanatory and hence the task of interpreting it is left to the will of the reader.

```
#
# 4 connecting to 5 at time 89.745845245079067
#
set udp_(0) [new Agent/UDP]
$ns_ attach-agent $node_(1) $udp_(0)
set null_(0) [new Agent/Null]
$ns_ attach-agent $node_(5) $null_(0)
set cbr_(0) [new Application/Traffic/CBR]
$cbr_(0) set packetSize_ 512
$cbr_(0) set interval_ 0.02
$cbr_(0) set random_ 1
$cbr_(0) set maxpkts_ 20000
$cbr_(0) attach-agent $udp_(0)
$ns_ connect $udp_(0) $null_(0)
$ns_ at 89.745845245079067 "$cbr_(0) start"
#
```

Chapter 4

Performance Evaluation of Routing Protocols

4.1 Simulation Environment

The simulations were carried out in NS2 using the methods that have been discussed previously. This includes the generation of the mobility model using the 'setdest' tool available as a part of NS2 and the generation of traffic pattern using the 'cbrgen.tcl' script. The testing is done in a simulation environment which is 1000m x 600m in area and has 30 mobile nodes. The topography can be defined using either a flat grid as in our case or using a digital elevation map. The simulation time is 200 seconds and each simulation is performed under varying pause time and varying number of traffic sources. The pause time indicates the amount of time that a node will pause inbetween two transitions. The pause times considered for this particular simulation are 10, 20, 40, 60 and 120 seconds. A pause time of 10 seconds would denote a rapidly changing network topology and a pause time of 120 seconds would denote a relatively stable network. The number of traffic sources considered are 5, 10, 15 and 20. The speed of the nodes are randomly assigned during the creation of the mobility pattern. The speed varies between 0 and 10 m/s. CBR traffic is sent where the packet size is set to 512 Bytes and the packet interval time is 0.02 seconds. This means that we are trying to stream data at the rate of 200kbps which is assumed to be a good data rate for receiving a video of good quality. The bandwidth of the wireless links is 11Mbps, similar to those of a 802.11b based network. Under the above conditions we have studied some performance metrics of four ad-hoc routing protocols, namely, DSDV, AODV, DSR and OLSR.

4.1.1 Medium Access Control

IEEE's 802.11 Medium Access Control (MAC) Distributed Coordination Function (DCF) [3] is implemented in the link layer of the simulations. Two nodes that decide to communicate must first exchange Request-to-Send (RTS) and Clear-to-Send (CTS) messages to reserve the wireless channel for their communication. These messages help to avoid the problem of hidden nodes and thereby reduce data loss occurring as a result of collision at a node. One of the major limitations of using DCF is that once a node reserves access to a wireless channel, it can hold on to it for as long as it choses to. This may cause a wireless channel starvation for other nodes that may be waiting to reserve the same channel to transmit data.

4.1.2 Packet Buffering

The mobile nodes are configured to buffer upto 50 packets that are to be transmitter by the network interface. Any packet that arrives after the queue is filled up will force the node to drop a packet in the drop tail fashion. 'Queue/DropTail/PriQueue' and 'CMUPriQueue' are the two commonly used values for the interface priority queue(IFq) parameter while configuring the simulation environment. PriQueue is a priority queue where the routing packets are given priority over normal data packets by inserting them at the head of the queue. It is possible to filter out packets placed in the queue based on a specified destination address by running a filter over all the packets.

4.1.3 Network Traffic

There are two kinds of traffic that can be used in the simulations. One is the TCP based traffic and the other is the UDP based Constant Bit Rate (CBR) traffic. In all our simulations we have used the CBR based traffic that runs on top of UDP since UDP has proven to consume much lower energy than TCP in a previous study. Since UDP traffic is employed, there are no retransmission of lost packets and lost packets are simply ignored. The only traffic that is present in the network is the legitimate traffic between the mobile nodes. For the sake of simplicity we have assumed that there is no background traffic. Majority of the work that was presented in the 'Related Work' section were performed under similar traffic assumptions.

4.2 Packet Delivery Fraction (PDF)

The packet delivery fraction is the ratio of total number of successfully received packets to the total number of sent packets. We have visualized the results in two different ways: a) the effect of increasing the network load in each of the routing protocol and b) the performance of each protocol under the same network load. This is done so that the reader gets a better understanding of the results that were obtained. It is very obvious from the first set of graphs that as the network load is increased by increasing the number of traffic sources the packet delivery fraction drops. In the case of all four routing protocols the PDF drops by almost 50% when the number of traffic sources in the network goes up from 5 sources to 10 sources. However, the further drop in PDF for higher number of traffic sources seems to be marginal compared to initial drop and the PDF achieved under such high network loads may not be acceptable in most cases. We also observe another expected behavior which is the increase in the PDF as the network becomes more and more stable. The PDF achieved when the pause time is 120 seconds is relatively higher than those achieved when the pause time is 20, 40 or 60 seconds. This result is clearly visible under low network loads in each of the protocol as compared to higher network loads.

With increasing load the PDF offered by OLSR and AODV seems to be improving. This may suggest that these two protocols work slightly better under these heavy traffic conditions. However, under low traffic loads DSDV and OLSR seem to be better. In all the observed cases we notice that DSR has the worst packet delivery fraction. This may be due to the aggressive caching of routes by DSR. As a result of this there are very few routing control packets when this protocol is employed. Hence DSR is inclined to choose a wrong route to a destination as the route chosen may be stale due to the rapid change in network topology over time.

The good packet delivery fraction offered by AODV under heavy loads can be addressed by the fact that AODV being an on-demand routing protocol finds a route to a destination only when needed. Hence the probability of choosing a stale route is low. This probability is further reduced in AODV because of two reasons, namely, sequence numbers associated with route entries are used as indicators of route freshness and AODV maintains timer based states in each node for utilization of individual route entries.

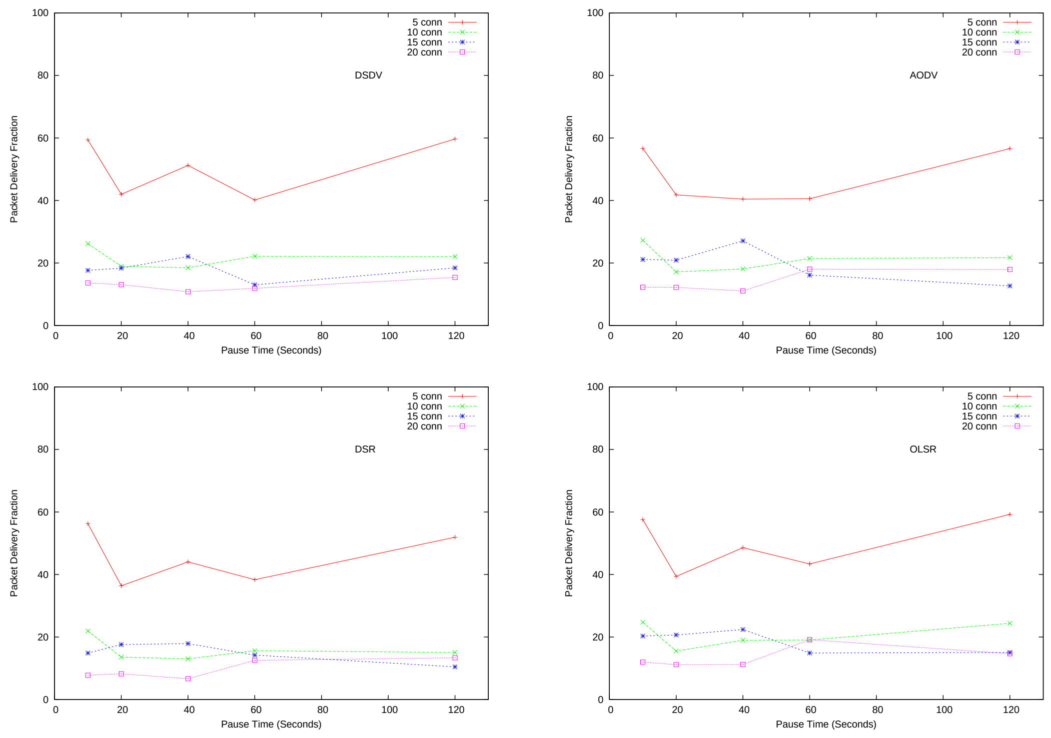
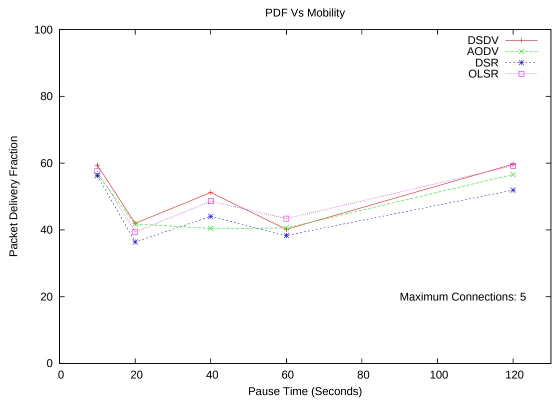


Figure 4.1: **a) PDF - Effect of increasing the network load in each of the routing protocol**

4.3 Total Dropped Packets

Counting the total number of dropped packets in a particular scenario by employing different routing protocol can give us an idea of the performance of these protocols. As discussed earlier, we can see that as the network becomes more stable the total number of dropped packets reduces in majority of the cases. We can also observe that as the network load increases, the total number of dropped packets also increases. The count of dropped packets during the simulation period is lowest in the case of OLSR under almost all traffic loads. Similar to results that we observed under packet delivery fraction we can see that DSR has the maximum number of dropped packets. Once again we can reason that this is due to the high probability of using stale routes by DSR while routing packets to a destination.



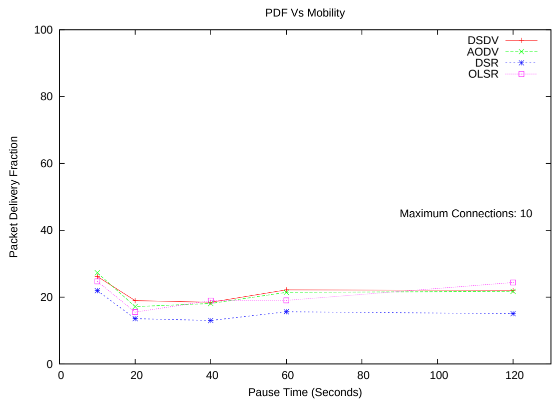
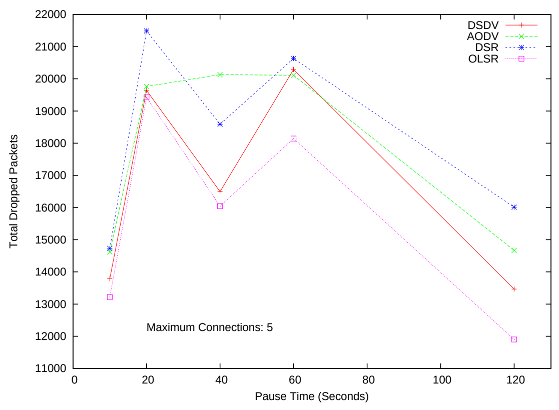


Figure 4.2: **b) PDF - Relative performance of each protocol under the same network load**

4.4 Normalized Routing Load

The normalized routing load is defined as the ratio of total routing control packets to the total number of successfully received packets. In terms of normalized routing load we observe that DSDV has the lowest value for all different traffic loads considered. This may appear contradictory to our expectations of seeing DSR as having the lowest normalized routing load due to their aggressive caching mechanism. However, the results obtained aren't misleading or wrong. In fact they show us the real picture of the DSR protocol. Although DSR does have the lowest routing overhead, when the ratio of routing overhead to successfully received packets is considered we see that DSDV outperforms DSR.



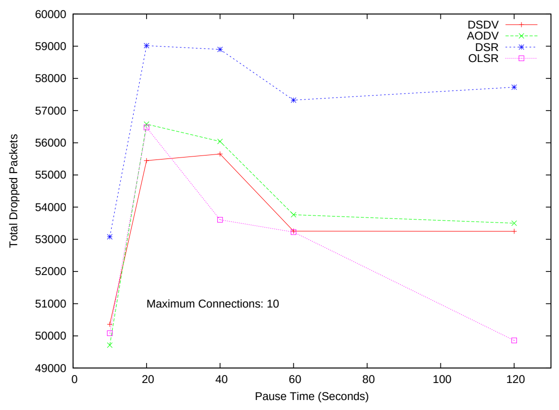


Figure 4.3: **Total number of dropped packets**

Both DSR and AODV are on-demand routing protocols, so as the network load is increased by increasing the number of traffic sources, we expect the number of routing control packets to go up since there are more destination nodes to which the network must establish and maintain routes. This expected behavior is reflected in our normalized routing load values as well with the steady increase in their values with the increasing number of traffic sources in the network. From the graph that depicts the case where we have 10 traffic sources, we can see for ourselves that the normalized routing load of DSR and AODV are significantly higher than that of OLSR and DSDV. DSR and AODV however have one desirable property, with the increase in the number of traffic sources the incremental cost associated with the additional traffic sources decreases as both protocols are capable of using information learned from a previous route discovery to complete any subsequent route discovery that is performed.

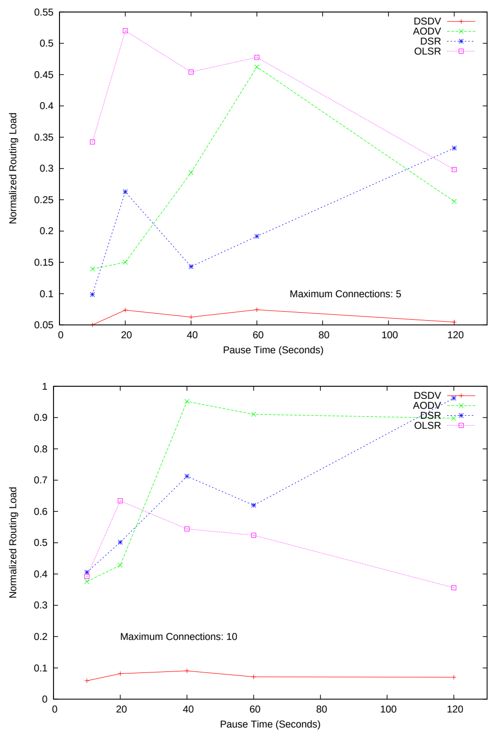


Figure 4.4: **Normalized Routing Load (# Routing Control Packets /# Successively Received Packets)**

The routing overhead of DSDV is almost constant and does not change much with increasing network load. In the case of DSDV, each mobile node broadcasts routing control packets periodically and topology changes occurring due to mobility does not trigger an update instantaneously. In DSDV, when a node receives a periodic update with a new sequence number for a node, it knows that the update is important since there has been a change in the topology. The node immediately distributes the update to other nodes around it. Hence every node that receives a periodic update with a new sequence number for a node generates a triggered update that is broadcasted. This may cause the network to get flooded with routing control packets as every node that receives a triggered update will further generate more triggered updates to be broadcasted. However, this exponential growth in routing control packets does not happen at the rate that one would expect it to happen. This is because every node limits the rate at which it sends out triggered updates to one per second.

While measuring the routing overhead, we have considered only the number of routing control packets and have ignored the size of the control packets and the size of the source route header placed by DSR in each of the data packets. If the size of the routing controls packets and the source route header were to be taken then DSR will clearly have a much higher routing overhead than what is currently considered [10]. Even though AODV and DSR have almost the same mechanism, AODV stores hop by hop routing state in each of the mobile nodes along a route, thus eliminating the need to place source routing information in the data packet header.

4.5 Average Throughput

Throughput of an established data flow is calculated as the total number of kilo bits (Kb) of data successfully received by the receiver per unit time (second). We have considered scenarios where we have multiple data flows/-connections during a simulation. Hence we take the average of throughputs of all these connections to obtain our average throughput. During our simulations we noticed that not all the connections were established successfully. At times, nodes that participated in a connection were unreachable due to network partitioning and as a result of that these connections could not be established. In such cases, we have ignored the zero throughput achieved for theses connections, while calculating our average throughput.

We can notice that DSDV offers higher average throughput than the other protocols when the network load is less. As the traffic load increases, there is a considerable decrease in the average throughput of all the protocols. DSR offers the lowest throughput as we had expected. This is due to the high probability of stale route selection in DSR as discussed previously. This causes a large number of packets to be dropped and thereby lowers the average throughput. We can also observe the steady increase in the average throughput as the network becomes more and more stable. The average throughput offered by OLSR was observed to be better than the rest when the number of connection in the network are roughly 20. This improvement in performance under heavy traffic load can be explained by the fact that OLSR is best suited for a large and dense mobile ad-hoc network where a large subset of the nodes are communicating with another large subset of nodes [14]. It

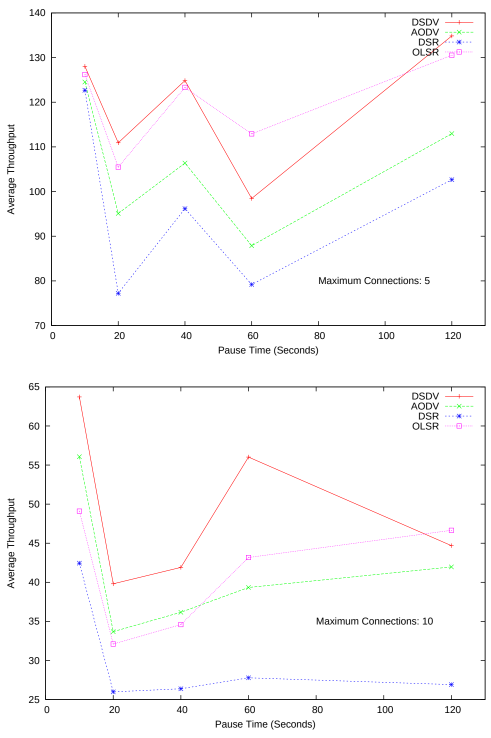


Figure 4.5: **Average Throughput (in Kbps)**

is important to remind the reader at this point that we are streaming at 200 Kbps to the receiver node. The maximum average throughput that we observe is only about 130-135 kbps and it is achieved when the number of connections in the network is less. The average throughput achieved at high traffic loads is not suitable for video streaming.

4.6 Other Observations

4.6.1 Effect of varying the mobility rate

The packet delivery fraction (PDF) was observed for DSDV protocol at periodic intervals during a 800 seconds simulation period. The simulations were conducted with varying levels of mobility, ranging from zero or no mobility as indicated by a Pause Time of 800 seconds to continuous mobility indicated by a Pause Time of zero seconds. The speed of the mobile nodes can be adjusted while generating the mobility pattern using the 'setdest' tool available as a part of the NS2 simulator. It is clear from the plots that the performance of the protocol in terms of PDF is hardly affected by the low walking speeds. For example, considering the scenario where there are 5 connections, it can be noted that in most cases the PDF remains around 40% irrespective of the level of mobility. As the load in the network increases the PDF decreases considerably. The PDF reduces by about 50% when the load shifts from 5 connections to 10 connections. However, in the case 15 and 20 connections the PDF remains almost the same for all scenarios. Network partitioning and the resulting unreachability due to mobility, appears to be one of the major causes of the low packet delivery fraction. It is also interesting to note that over time the PDF seems to stabilize and remain constant without deteriorating further.

4.6.2 Effect of varying packet size

A set of simulations were carried out using DSDV routing protocol with varying packet sizes in order to observe its effect on the performance of the protocol. The data packet sizes considered are 512, 768, 1024 and 2048 bytes.

Packets larger than these were not considered for two reasons: Firstly, the maximum transmission unit (MTU) supported by 802.11 standard is 2272 bytes. Secondly, past studies have shown that the use of large packets impose heavy costs on nodes and the probability of these packets getting corrupted is also high due to the high bit error-rates in wireless networks [8]. The network is subjected to traffic from five sources and the number of connections in the network has been kept constant throughout all the simulations. The packet interval time has been adjusted for each packet size to make sure that the nodes are streaming at 200 Kbps. This rate has been observed to be a good data rate for streaming videos.

It is interesting to see that the packet delivery fraction is higher in the case of packets of size 768 Bytes compared to the 512 bytes packets. A similar behavior is exhibited by the 2048 bytes packets and the 1024 bytes packets as well, with the former offering a slightly higher packet delivery fraction than the latter. After the rapid initial drop in the PDF, the network seems to have reached a state where the PDF remains almost the same until the end of the simulation.

The average throughput that was obtained by streaming packet of different

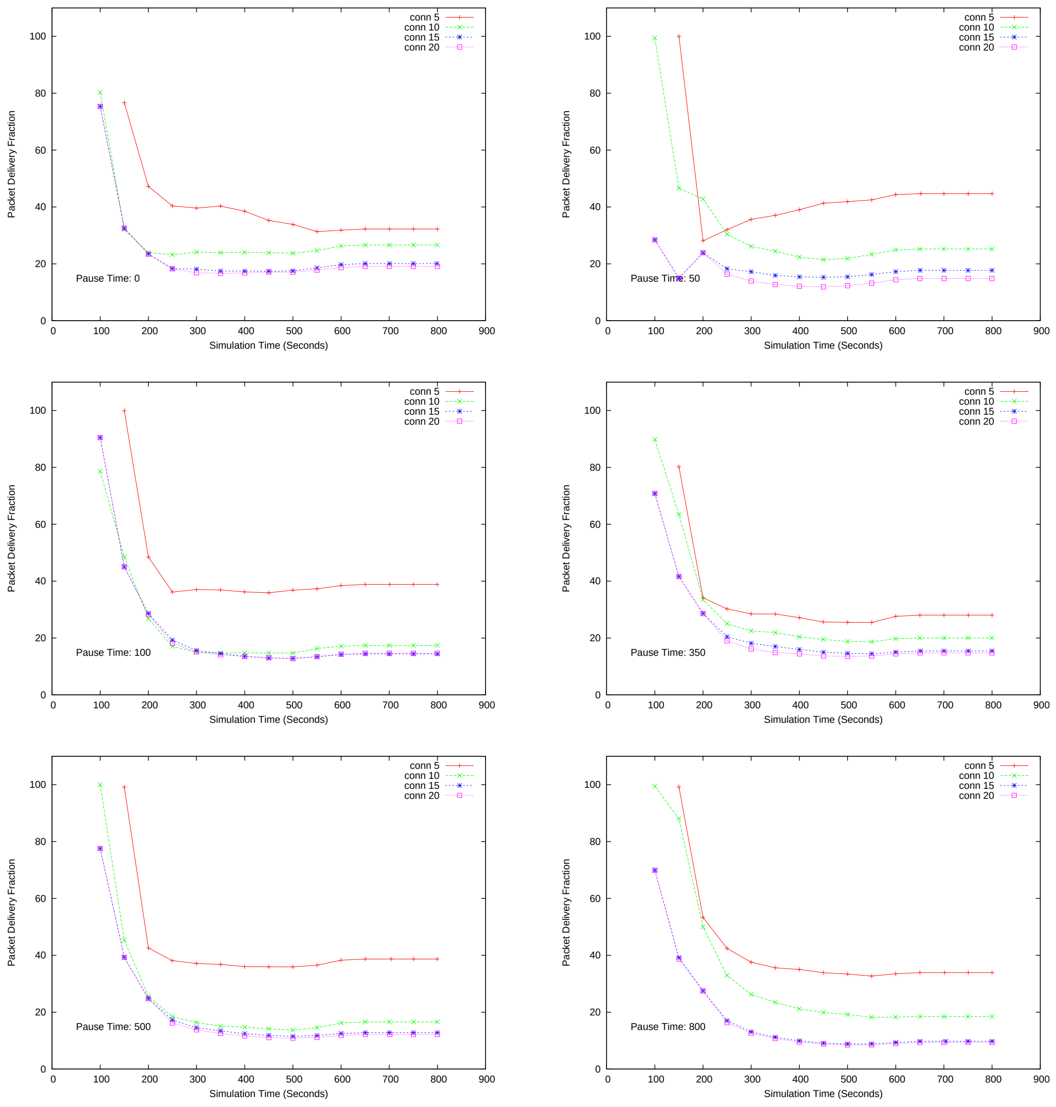


Figure 4.6: **Packet Delivery Fraction of DSDV under varying levels of mobility**

sizes does not provide any concrete results and is in fact contrary to our expectations. One would expect the average throughput to increase in a more stable network where the mobile nodes move less frequently, but the throughput that we observed in such a scenario was lesser than certain throughputs that were observed when the nodes were in motion. We can safely argue that

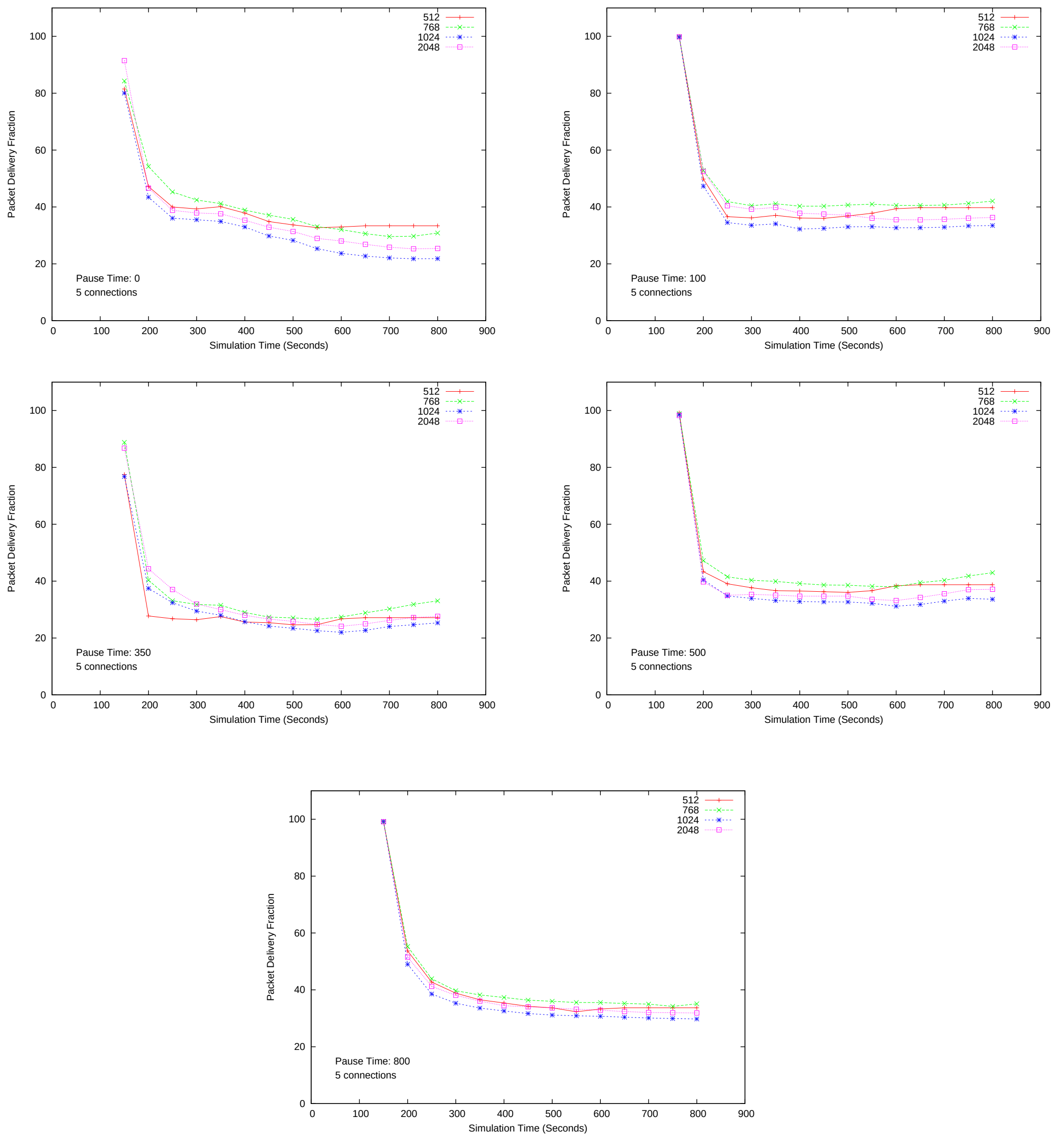


Figure 4.7: **Packet Delivery Fraction of DSDV under varying packet sizes**

the low mobility speeds (0 - 1.5 m/s, which is the average walking speed of a person) that we assign for the nodes in our simulation has very little impact on the performance of the protocol. Network partitioning that happens when some nodes become unreachable, may be blamed for the low throughput that

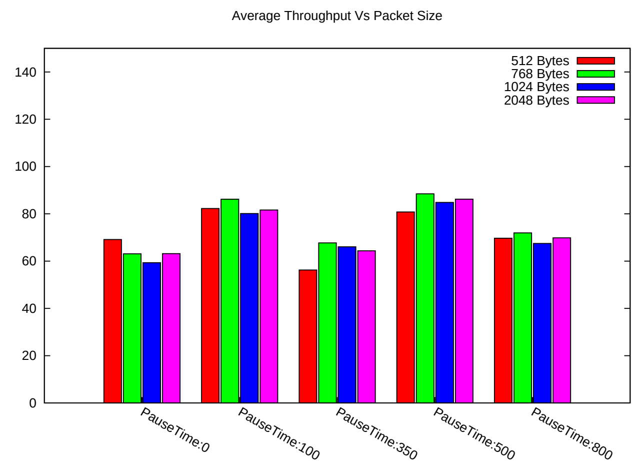


Figure 4.8: **Average Throughput (Kbps) Vs Packet Size (Bytes)**

we get in some cases.

4.7 Summary

Our observations indicate that network load has a direct effect on the packet delivery fraction. PDF decreases with the increase in network traffic. PDF drops by almost 50% when the number of traffic sources in the network goes up from 5 sources to 10 sources. We also observe another expected behavior which is the increase in the PDF as the network becomes more and more stable. With increasing load the PDF offered by OLSR and AODV seems to be improving. DSR has the worst packet delivery fraction among all protocols considered. It also drops the maximum number of packets due to the high probability of stale route selection resulting from its aggressive caching mechanism. DSDV has the lowest normalized routing overload (NRL). The NRL increases steadily, as the number of traffic sources in the network increases, in the case of on-demand routing protocols like DSR and AODV. DSDV offers the highest average throughput while DSR offers the lowest. The maximum throughput that was achieved was only between 130-135 Kbps. We also observed that the low walking speeds employed for the mobile nodes in our simulations, had very little impact on the PDF. The throughput achieved using 768 bytes packets is higher than that achieved using 512 bytes packets. This was also true in the case of 2048 bytes packets offering a higher throughput that 1024 bytes packets.

Chapter 5

Energy Consumption of Routing Protocols

5.1 Power Models for NS2 Simulations

The measuring of the energy consumption behavior of a node in a mobile ad-hoc network requires an energy model. Observing the energy consumption of a node while employing different routing protocols will provide us with deep insight on the performance of the protocol. This will also help us make the necessary improvements to the protocol to make it more energy efficient. There are a wide range of energy models that are available. Some models account only for the energy consumption at the network interface while some models also consider the energy consumed by other hardware units such as processors, display units, etc. Although energy consumption is a very important criteria in the evaluation of routing protocols, very little work has been done along these lines. Two energy models were considered for our purpose and one was finally chosen over the other since the model reflected the specific hardware that was being simulated. The evaluation of energy consumption is particularly important in a mobile ad-hoc environment because a mobile node supports not just the applications that are running locally on it, but also the unpredictable demand of the network infrastructure and applications that are running in remote nodes.

Usually in an infrastructure network the base station is responsible for moderating the communication between the nodes that make up the network. They schedule and buffer traffic so that the nodes can spend most of their time in the sleep state and conserve their energy. The base station wakes up the nodes that are in sleep state only when their participation is required to complete some communication. However, we do not have any base stations in an ad-hoc network and hence there is no central entity that controls the nodes and their states. Even the nodes are unaware of when they will receive any traffic and therefore the nodes by default stay in the idle mode waiting to either transmit or receive data.

5.1.1 Feeney's Energy Consumption Model

This model [17] is built based on the IEEE 802.11 protocol rather than on the electronic properties such as mode switching or signal response. The experimental results of energy consumption that were obtained through the simulation of IEEE 802.11 wireless interface are incorporated into the model, thus providing a quantitative measure for the energy consumption. A network interface is always in one of the four possible states: transmit, receive, idle and sleep. While in transmit state the node transmits data and it receives data packets while in the receive state. A node can either transmit or receive while in the idle state which is also the default state of a network interface. When the network interface does not have anything to transmit or receive, it switches to sleep mode. When in sleep mode, a node consumes extremely low power and thus helps in conserving energy. The node needs to be woken up again in order to perform a transmit or receive operation. A node also spends comparatively less power in the idle state than in the receive or transmit state.

This model assumes that there is no arbitrary transition to the sleep state and nodes periodically enter sleep state before waking up again to participate in a communication. The model also assumes that the cost associated with the link layer operations is constant and this assumption may not hold true in many cases, for example, signal strength affects the energy required to receive a data packet. In this model, the cost to send or receive some traffic is modeled to be linear. The cost comprises of a fixed cost (b) associated with the channel acquisition and an incremental cost (m) that is proportional to the size of the packet.

Cost = m x size + b

When a communication happens, the total cost of transmitting a packet from one node to another includes the cost incurred by the sender node, the destination node and the nodes that are within the range of the source node or the destination node and those that participate in the communication. In the case of point to point traffic the incremental payload cost m$_{send}$ and m$_{recv}$ are the same and the fixed cost comprises of both the channel access cost and the MAC negotiations. The IEEE's 802.11 MAC protocol comprises of the RTS (Request to Send) and CTS (Clear to Send) signals. Before a node begins to transmit data, it first sends out a RTS signal to the receiver. The receiver replies with a CTS message and this is followed by the actual data transmission. On receiving the data the destination sends an ACK message back to the source. This model assumes the cost associated with the control messages (b$_{sendctl}$, b$_{recvctl}$)to be the same in order to keep the model simple. The model also proposes the cost of discarding a packet when a non-destination node that is not intended to receive the packet happens to overhear due to its presence in the range of the sender or the receiver node.

[Sender] Cost $= b_{sendctl} + b_{recvctl} + m_{send}$ x size $+ b_{send} + b_{recvctl}$

[Receiver] Cost $= b_{recvctl} + b_{sendctl} + m_{recv}$ x size $+ b_{recv} + b_{sendctl}$

[Discard] Cost $= \sum_{n \varepsilon S} b_{discardctl} + \sum_{n \varepsilon D} b_{discardctl} + \sum_{n \varepsilon S} (m_{discard}$ x size $+ b_{discard}) + \sum_{n \varepsilon D} b_{discardctl}$

It is not possible to obtain direct values for the coefficient used in the above equations from the wireless network interface. Hence through conducting some experiments that measure the energy of 802.11 based network interface cards, constant values have been proposed for the above coefficients. Since these values have been obtained through indirect measurements, they might not be accurate and make be off by about 10-15%. However, they still would give us a good picture of the relative power consumptions of our routing protocols. This will help us evaluate the energy efficiency of the routing protocol and also help in identifying short comings in the protocol implementation that can improved.

Constants	Power Consumption
m_{send}	1.89 mWs/byte
b_{send}	246 mWs
m_{recv}	0.494 mWs/byte
b_{recv}	56.1 mWs
$m_{discard}$	-0.490 mWs/byte
$b_{discard}$	97.2 mWs
$b_{sendctl}$	120 mWs
$b_{recvctl}$	29 mWs

Constants used in the energy model

5.1.2 Power Modeling for Data Transmission over 802.11g networks for Wireless Applications

This power model [49] can be used to estimate the power consumption during data transmission over an 802.11g WLAN. The wireless network interface (WNI) is the component that consumes the maximum power during data transmission in a wireless application. This model is based on the internet flow characteristics and the Power Saving Modes employed in mobile devices to conserve energy. Although it is possible to estimate the power consumption of the WNI by measuring the amount of time the WNI spends in each operating mode, this kind of measurement is challenging as it is hard to obtain the total time spent in each mode directly from the mobile device. The energy consumption of data transmission is basically composed of the transmission cost and the computational cost. The former cost is based on the operation of the WNI, whereas the latter is based on the cost of copying data between the user space, kernel space and the network interface.

For the purpose of our simulation using NS2, we assume that the WNIs of the mobile nodes that make up the mobile ad-hoc network operate in the Continuously Active Mode (CAM) and not in the Power Saving Mode (PSM). We make this assumption because it is hard to guess when a node might receive data or forward data to other nodes in the ad-hoc setup. The WNIs of our mobile nodes will thus be only in one of the three modes: Transmit, Receive or Idle. In this paper, the authors also observe the effect of changing the transmit power of the antenna in the mobile device. The mobile device that was used is the Nokia N810 Internet tablet which is the same device that is used in our testbed. Even our simulations were carried out by adjusting the necessary parameters to mimic the same hardware profile of a N810 Internet tablet. The effect of changing the transmit power of the antenna had insignificant effect on the total power consumption. On a Nokia N810 device the difference in power consumption was less than 1% when the device's transmit power was changed from 10mW to 100mW. The average power consumption of a Nokia N810 under different WNI operating modes is given below.

WNI operating mode	Average Power (W)
Idle P$_I$	0.884
Sleep P$_S$	0.042
Transmit P$_T$	1.258
Receive P$_R$	1.181

Average power consumption of a Nokia N810

5.2 Energy Consumption of Point to Point Traffic

Understanding the energy consumption of routing protocols is an important issue for mobile computing devices in an ad-hoc network. Routing protocols that are deigned for the mobile ad-hoc environment consider energy consumption as one of the design criteria. However, during the evaluation of a routing protocol, its energy consumption takes a low priority. Observing the energy consumption while evaluating a routing protocol will throw light on costly protocol behavior that can be fine tuned to achieve higher energy efficiency.

For evaluating the energy consumption of the routing protocol, we use the energy model that is built into the NS2 network simulator [24]. This energy model is built around the IEEE 802.11 MAC protocol. The results that are shown below are values of total energy consumed by the network interfaces of all nodes that make up the ad-hoc network. A set of simulations were performed with point to point traffic. The network load was varied by increasing the number of connections that existed in the network. A constant packet size of 768 Byte was used throughout all the simulations since it had offered a much higher throughput than 512 Bytes data packets, in one of our previous simulations.

One of the observations that is immediately noticeable from the graphs below is the low energy consumption of on-demand routing protocols. Clearly, DSR and AODV being on-demand routing protocols, consume much lower energy than OLSR and DSDV. In terms of bandwidth utilization, DSR is usually considered as the most efficient protocol. However, in terms of energy consumption, it is less efficient than AODV due to the high cost of eavesdropping. Nevertheless, the nodes that use DSR, eavesdrop on the source routing headers of all data packets forwarded by nodes which are in their one hop neighborhood. As a result of this, these nodes do not have to perform route discovery very often since their route cache may already contain a large fraction of the information about the network topology. A considerable improvement in the performance was observed by using this method of promiscuous eavesdropping in routing algorithms [37]. On one hand, this helps in reducing the cost incurred with broadcast flooding to find or repair routes, but on the other, the cost of eavesdropping on traffic in promiscuous mode increases.

There is also a gradual increase in the total energy consumption with the increase in the number of connections in the network. The energy cost increases by only 500 joules on an average in the case of OLSR and DSDV when the number of connections increases from 5 to 15, whereas in DSR and AODV it is considerably higher. This could be associated with the increase in the number of routing packets required to maintain routes to more destination nodes in the case of on-demand routing protocols. Proactive routing protocols by default maintain routes to all possible destinations within the network irrespective of whether there is any data to be sent to that destination or not.

DSDV has a very high energy consumption because every time a node realizes that there has been a topology change, by observing a new sequence number in the periodic updates that it receives, it would in turn send out a triggered update to all other nodes. This flooding of the network with control packets is the primary reason for this high energy consumption that we observe in these plots.

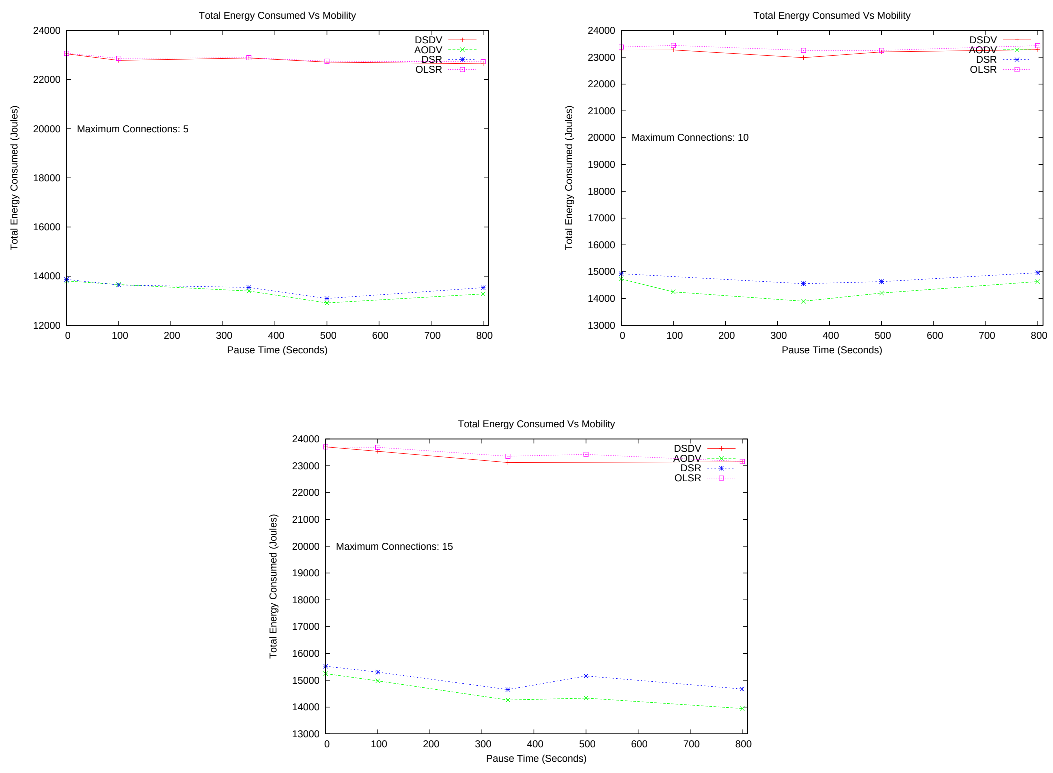


Figure 5.1: **Total Energy Consumption of Routing Protocol @ Network Interface for 30 nodes**

5.3 Energy Consumption in a Multi Source - Single Destination Setup

The plots presented in this section are obtained by performing a set of simulations that are slightly different from those performed up until now. These simulations were carried out under the assumption that the mobile nodes are streaming data to a central server. This assumption was made since such a setup would simulate an environment which can be found in the case of a disaster recovery operation, where hand held mobile devices are streaming video to a server. The total energy consumed per node through the entire 800 seconds simulation is presented in the plots. These values are averages of the total energy consumed per node, in three different scenarios which are generated using the previously mentioned 'setdest' tool in NS2. The process of evaluating the energy consumption of a node as an average over multiple scenarios, reduces any error that may occur as a result of considering just a single topology during the evaluation process. It is possible at times that a different topology might yield a different result. However, our methodology will erase any such doubt that might arise by considering multiple random topologies that have random node mobilities within them.

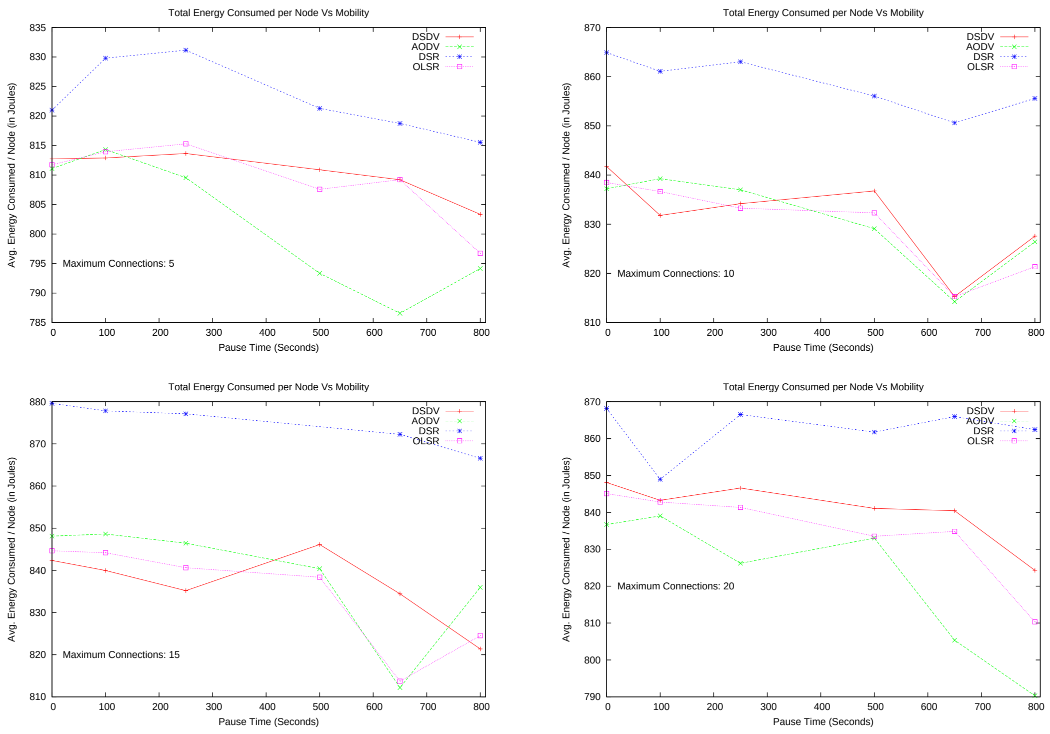


Figure 5.2: **Avg. Energy Consumption Per Node**

What we observe from these plots is very interesting since in our previous set of simulations we noticed that DSR actually consumed lesser energy than the other protocols, whereas here we notice that it consumes much higher energy than the rest. Although, we cannot provide a concrete reasoning as to why this happens, it would be reasonable to associate this energy behavior with the aggressive caching mechanism used by DSR. The aggressive caching could lead to the selection of stale routes while routing packets, and this does not happen with all on-demand routing protocols. In the case of on-demand routing protocols such as AODV, a source node will simply not inject any packet into the network if it fails to discover a route to the destination. Whereas with DSR's aggressive caching, there is a high probability that a sending node may use one of the previously available routes in its route cache to send packets to the destination node, and this route may already be stale at the time of route lookup. This results in packets being sent along a route that is no longer valid and ultimately these packets will be dropped somewhere along the route. This unnecessary routing of packets along stale routes may be one of the reasons for the high energy consumption of DSR. Another reason for the high energy consumption in DSR, as mentioned earlier, could be a node's eves dropping on the source routing header of packets forwarded by other nodes in its one hop neighborhood.

There is not much of a difference in the energy consumption of the remaining three protocols under low network traffic. However, as the load in the network increases we notice the lines in the plots begin to move away from each other and thus give us a clear picture of the average energy consumption per node for each of the routing protocol. Under heavy traffic, AODV seems to consume the the least energy, followed by OLSR and DSDV in the same order as they are mentioned here. In the case of OLSR, the Multipoint Relays (MPRs) forward broadcast messages on behalf of all nodes that have chosen it as a MPR and this helps in scaling the number of control packets by eliminating the broadcast flooding performed by all nodes within the network. This optimization offered by OLSR is the key reason behind its low energy consumption compared.

The energy consumption drops steadily as the network becomes more stable due to the low mobility of the mobile nodes. The network topology of a stable network does not change as frequently as it does with an unstable network and hence the number of control packets exchanged are lesser in the case of the former compared to the latter. As a result of this, considerable amount of energy is saved when the mobility level of nodes within a network is low. There is also an increase in the energy consumption as the network load increases. This is obviously due to the increase in the number of data packets handled by the nodes during the simulation.

5.4 Avg. Energy Consumption with Varying Packet Size

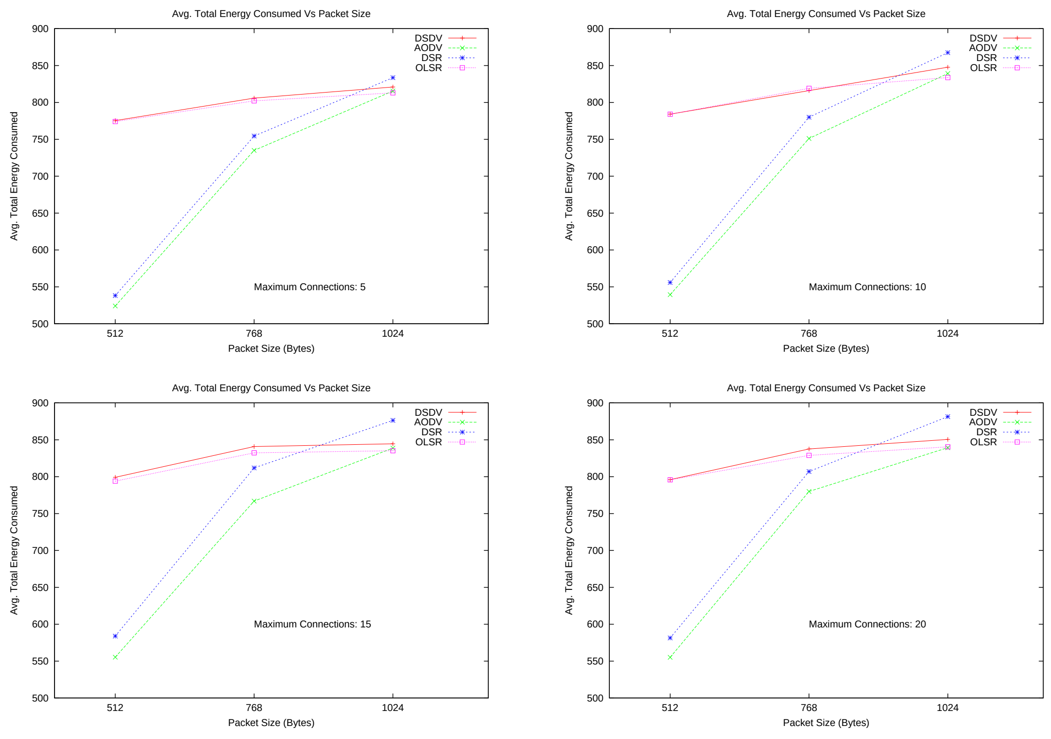


Figure 5.3: **Avg. Energy Consumption Per Node with varying Packet Size**

The effect of varying packet size on the total energy consumption of a node has been calculated by taking the average of energy consumption values of five different scenarios that have different node mobility levels. The node mobility levels are determined using the pause time of nodes: 0, 200, 400, 600, 800 seconds. The packet sizes considered for the simulation are 512, 768 and 1024 Bytes. The packet interval time is adjusted in order to get a data transmission rate of 200 Kbps. The total number of simulations that were performed are:

```
5 Scenarios x 3 Packet Sizes x 4 Protocols x 4 Network Loads = 240
```

We observe that the energy increases with the increase in the packet size. However, the increase in energy consumption is much greater when we move from 512 bytes to 768 bytes compared to moving from 768 bytes to 1024 bytes. DSR which consumes lesser energy than DSDV and OLSR for 512/768 bytes data packets, consumes more energy than the rest when 1024 byte packets are used. The reason why there is an increase in energy consumption when one uses large packets can be assumed to be related to the amount of time these large packets occupy in intermediate nodes along a route. A large packet indicates more payload bits that need to be handled by the network interface card. This means that the network interface card spends more time in the Transmit mode and ultimately less time in the Idle or Receive mode. It is known to us that the NIC consumes more power while in Transmit mode compared to other modes and this explains our observation.

As mentioned earlier, the data transmit rate has been maintained at 200 Kbps for all packet sizes by adjusting the transmission rate accordingly. Hence, the larger the packet size , the lower the transmission rate. We know that with increasing transmission rate the energy cost also increases since a NIC has to handle more transmissions. One would therefore expect the energy consumption to drop as we use larger packets, but the energy saving achieved by using larger packets is much lesser than the energy lost due to the longer durations these packets occupy in the intermediate nodes. This explains the steep raise in the energy costs with increasing packet sizes. It is interesting to note that the increase in the energy cost in the case of DSDV and OLSR, as one begins to use larger packets, is very minimal compared to the increase we see in DSR and AODV.

5.5 Avg. Energy Consumption per byte of successfully received Data

For obtaining this metric, we have taken the total energy consumption for each of the simulation and divided them by the total number of successfully received bytes for that particular simulation. We have taken into consideration not just the successfully received data bytes, but also the bytes of control data. On-demand routing protocols (DSR and AODV) once again show signs of consuming lower energy per byte compared proactive routing protocols. DSR routing protocol offers the lowest energy per byte value amongst these two on-demand routing protocols. The reason for this is that DSR conserves a lot of energy due to their low routing overhead compared to other routing protocols. The use of large packet size is also recommended since the value of the metric seems to drop with larger packet sizes. The drop in the energy per byte metric is more predominant in the case of OLSR and DSDV protocols. However, in the case of DSR and AODV this drop is almost negligible. The transmission rate that we require in the case of smaller packet sizes, in order to achieve a 200 Kbps stream rate, is greater than the transmission rate required with larger packets. We know for a fact that the energy consumption increases with the increase in the transmission rate. This is caused due to the prolonged durations that a network interface card must stay in the transmit mode with every increase in the transmission rate.

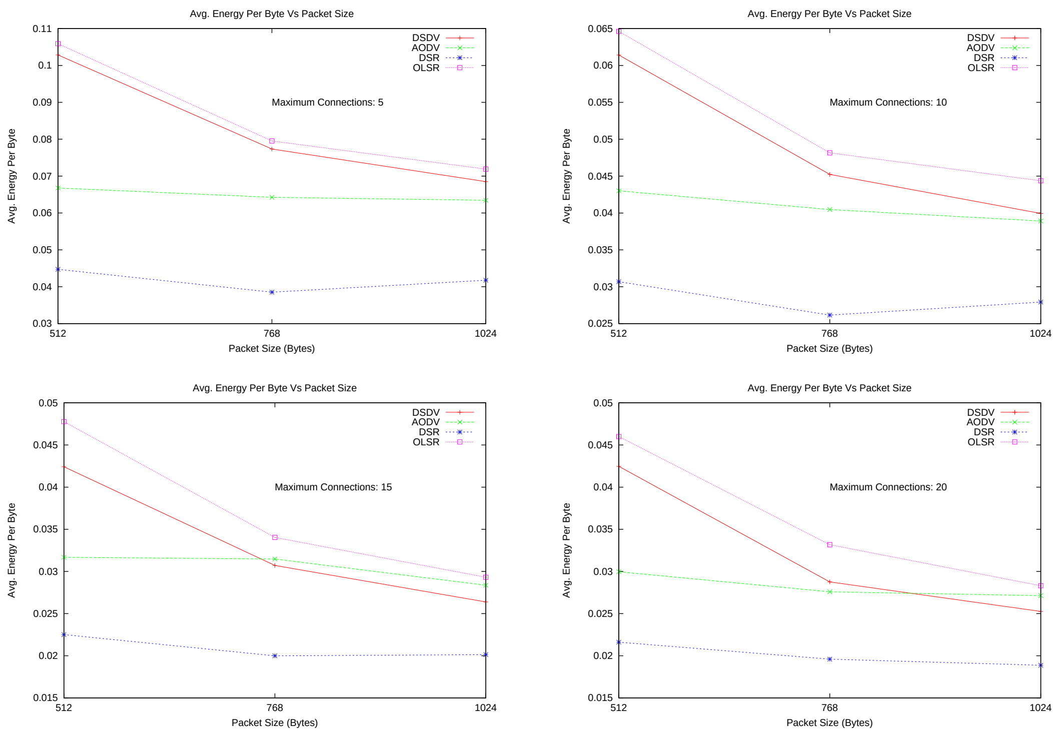


Figure 5.4: **Avg. Energy Consumption Per Byte of successfully received data**

5.6 Summary

In the evaluation of energy consumption in point to point traffic, we observe that DSR and AODV being on-demand routing protocols, consume much lower energy than OLSR and DSDV. DSR is usually considered as the most efficient protocol in terms of bandwidth utilization, but the energy conservation achieved due to this is over shadowed by the high cost of eavesdropping that DSR performs. The total energy consumption of a node increases as the traffic in the network increases. The energy cost increase that comes with the increase in network load is more predominant in the case of on-demand routing protocols than table driven protocols. This could be associated with the increase in the number of routing packets required to maintain routes to more destination nodes in the case of on-demand routing protocols. However, proactive routing protocols by default maintain routes to all possible destinations within the network irrespective of whether there is any data to be sent to that destination or not. For the multi source - single destination scenario, DSR is observed to consume the maximum energy. This is due to the unnecessary loss in valuable energy, resulting from transmission of packets along stale routes. Under heavy traffic, AODV seems to consume the the least energy, followed by OLSR and DSDV, in the same order as they are mentioned here. The control packet optimization that multipoint relays (MPRs) offer, is the reason for OLSR's low energy consumption. We observe that the energy increases with the increase in the packet size, just as would expect it to. However, the increase in energy consumption is much greater when we move from 512 bytes to 768 bytes, compared to moving from 768 bytes to 1024 bytes. In the measurement of average energy consumed per byte of successfully received data, on-demand routing protocols (DSR and AODV) once again showed signs of consuming lower energy per byte compared to proactive routing protocols.

Chapter 6

Power consumption in a real world MANET testbed

6.1 System-level Model for Runtime Power Estimation in Mobile Devices

In order to measure the total power consumption of a mobile device due to a streaming process, we need to choose a suitable power model that will account for the power consumption of every bit of major hardware in the mobile device. In [48] a system-level model is proposed for the runtime power estimation on mobile devices. The power estimated using this model, takes into consideration the processor utilization, the network interface card and the display. These three components consume the maximum power in the case of multimedia streaming.

The proposed regression model uses a set of hardware performance counters (HPCs), network transmission parameters and the display settings as regression variables. HPCs have been used widely in developing processor power models [41, 26]. However, the major bottleneck in using HPCs lies in the number of HPCs that can be monitored simultaneously during runtime. In the case of an ARM 1136 mobile processor only three HPCs can be monitored simultaneously during runtime, namely CPU_CYCLES, DCACHE_WB and TLB_MISS. CPU_CYCLES indicates the workload on the processor in terms of the number of clock cycles utilized for a particular process while DCACHE_WB and TLB_MISS indicate the memory access efficiency in a CPU cycle. Earlier energy profilers such as Nokia Energy Profiler use current and voltage to estimate power consumption, whereas this model uses variables that reflect the activity levels of hardware resources in the mobile device.

Power (W) = 0.7905 + 0.0068 x $g_1(x_1)$ + 0.0828 x $g_2(x_2)$ + 0.2270 x $g_3(x_3)$ + 0.0018 x $g_4(x_4)$ + 0.0015 x $g_5(x_5)$ + 0.4969 x $g_6(x_6)$ + 0.1361 x $g_7(x_7)$

$g_1(x_1)$= (DCACHE_WB/CPU_CYCLES - 0.009)/0.000458
$g_2(x_2)$= (TLB_MISS/CPU_CYCLES - 0.000597)/0.000418
$g_3(x_3)$= (CPU_CYCLES/monitoring_period - 1643.579)/1583.112
$g_4(x_4)$= upload data rate (KB/s)
$g_5(x_5)$= download data rate (KB/s)
$g_6(x_6)$= CAM switch
$g_7(x_7)$= brightness level

The HPCs are monitored from the user space using the oprofile tool. The HPCs are aggregated counters and the appropriate values for each variable is obtained by subtracting the counter values recorded just before and just after the monitoring period. The setup procedure for oprofile and the process of extracting periodic reports using the tool are discussed in detail in the appendix section.

6.2 Testbed Environment

The MANET testbed consists of a laptop which acts as the streaming server and a Nokia N810 tablet which servers as a streaming client. The mobile device's kernel has been flashed in order to install a kernel that supports oprofile. Oprofile lets us monitor the hardware performance counters in real time in order to estimate the power consumption using a liner regression model that uses these values as inputs. 'Media Stream' is the client end software for receiving a stream while the laptop uses a UPnP media server called 'MediaTomb' for streaming media data. The mobile device is configured to accept remote logins, using OpenSSH. This greatly eases our task of moving files between our system and the device and also lets up work on the mobile node from a remote terminal. The OLSRd is installed in both the laptop as well as the mobile device. Both nodes are made to join an ad-hoc network that can be easily created from the Nokia N810 tablet. Unfortunately, we were unable to introduce any form of mobility in the network. Both nodes were placed a few meters away from one another and the data was streamed between them. Furthermore, the streaming software that was employed did not have the option for adjusting the stream rates and hence the media stream was automatically uploaded and downloaded at rates that could be handled by the softwares, wireless medium and network interfaces. During the power consumption evaluation process the data traffic was monitored using packet analyzers like wireshark and tcpdump. These tools assisted in calculating the values for the upload and download data rates that are required in estimating the power consumption using the model that we have taken.

6.3 Introduction to Nokia N810 Internet Tablet

The Nokia N810 is an Internet tablet designed for providing Internet connectivity, web browsing, email, media playback facility and all the other functionalities expected from a pocket PDA. It is the successor to the N800 and comes with a 800 x 480 pixels touch screen that extends the user-interface capabilities. It comes installed with the OS 2008 operating system which is a modified version of the Debian/Linux, offering a rich graphical user interface. Since this operating system was built from Linux, many of the tools that are available for linux could be easily cross compiled for using on the tablet device. The Scratchbox environment is used for this purpose and it is configured as per the instructions provided in the appendix section for making the cross compiled softwares to work on the arm processor that is present in the N810 device. The default kernel of the tablet device does not support the installation of the oprofile software and hence it must be replaced with a more suitable kernel.

Hardware specification [2]

Size: 72 mm x 128 mm x 14 mm (226 g)
Display: High-resolution 4.13", WVGA (800x480 pixels, 65000 colors)
Processor: TI OMAP 2420, 400Mhz
Memory: DDR RAM 128MB, Flash 256MB
Storage: 2GB internal memory, Supports miniSD/microSD (max 8GB)
Operating Times:

- Battery: Nokia Battery BP-4L
- Continuous usage (display on, wireless LAN active): up to 4 hours

- Music playback: up to 10 hours
- Always online time: up to 5 days
- Standby time: up to 14 days

Connectivity: WLAN: IEEE 802.11b/g, Bluetooth specification v.2.0
Media:

- In-built media player for viewing and listening to downloaded, transfered or streamed media content and easy-on-device management of media library
- Direct access to shared media over Universal Plug and Play (UPnP)
- Supported video formats: 3GP, AVI, WMV, MP4, H263, H.264, MPEG-1, MPEG-4, RV (RealVideo)
- Supported audio formats: MP3, WMA, AAC, AMR, AWB, M4A, MP2, RA (RealAudio), WAV
- Supported playlist formats: M3U, PLS, ASX, WAX, WVX, WPL

The tablet device comes with a very limited set of standard Linux softwares that it supports. Majority of the utilities that are readily available are the ones in the BusyBox. BusyBox is an utility that combines many common UNIX utilities into a single executable. This forced us to install several softwares that were required for the purpose of the experiment. Some softwares and utilities that were used for the experimentation include Live Stream (Stream Client), OpenSSH server (remote login), tcpdump (packet analyzer), oprofile (HPC monitoring), olsrd (OLSR daemon), rootsh (for gaining root access), iwconfig (NIC configuration) and the Connection Manager (creating ad-hoc network).

6.4 Experiment Methodology

Initially, the power consumption of the mobile device when it is idle is measured. In this state the mobile device is not connected to any ad-hoc network and hence no routing control packets are exchanged. The power consumed by the mobile device is least in this state. Then the OLSR daemon is started and the device is connected to the ad-hoc network in which one other node (Stream server) is present. The protocol begins to populate the routing table and the node constantly exchanges routing information with other nodes that may be present in its range. Finally the power consumption is estimated individually for an audio and video stream. For these streaming processes, the hardware resources are utilized heavily and the power consumption results must indicate the same. The three hardware performance counters that are required for our linear regression power model are monitored in run time using oprofile.

6.5 Observations

The first set of observations were the power consumption and the hardware performance counter values during a sixty seconds test period, while the tablet device was idle. There was no routing information exchange and this reduces the power consumed by the network interface card. The mean power consumption of the device in this scenario was about 1.5858 Watts. The was a gradual increase in the number of CPU Cycles during the monitoring period. The next set of observations were carried out with the OLSR daemon running in the background. The mobile device was performing route discovery and route maintenance periodically. This control packet overhead adds to the slight higher power consumption at the network interface card. The mean power consumption in this scenario was about 1.6859 Watts. This is a meager 6.31% increase in the mean power consumption of the device. However, this estimate is bound to increase as the number of nodes in the ad-hoc network increases. In the testbed that we had setup, there was only one other node in the network apart from the Nokia N810 tablet and this was the laptop that was used as the streaming server.

The third scenario was to stream audio data to the mobile device and then evaluate the power consumption. The mobile device at this point in time has the routing protocol as well as the applications required for receiving and processing the stream adding to the existing workload on the CPU. The audio file that was streamed was of mp3 format and the stream upload and download rates were roughly 234 Kbps and 212 Kbps respectively. As mentioned previously, the applications used for our streaming purpose did not have the option for adjusting stream rates and the above mentioned data rates were determined by using Wireshark packet analyzer. The mean power consumption in this scenarios was 1.9121 Watts. This is a 13.41% increase in the power consumption compared to the power consumed when the mobile

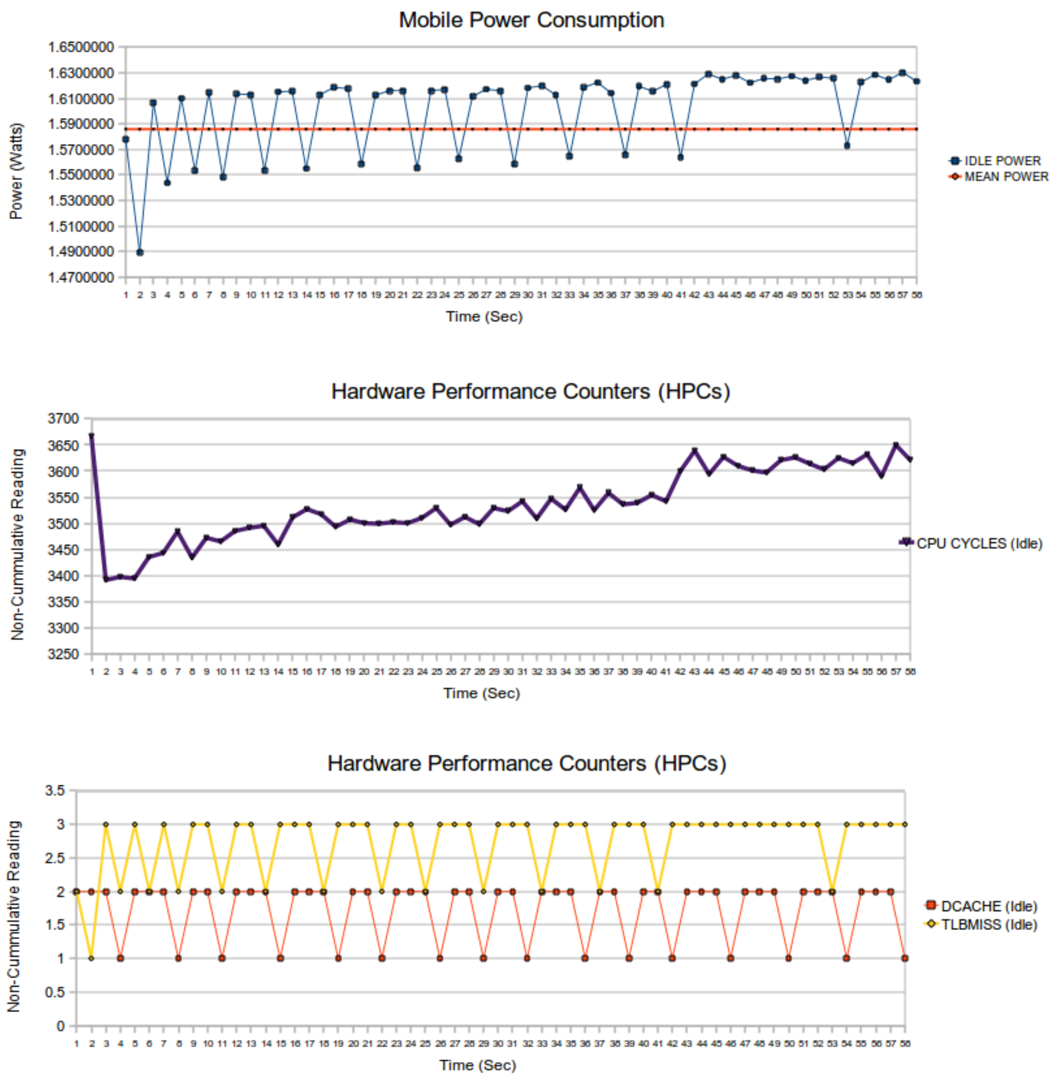


Figure 6.1: **Observations while mobile device is idle: Power Consumption and HPC Monitoring**

device has just the olsr daemon running in the background and a 20.57% increase compared to the idle state power consumption. We can also observe the HPC values being a little higher in this scenario compared to the previous two.

The fourth scenario was to observe the power consumption of the device

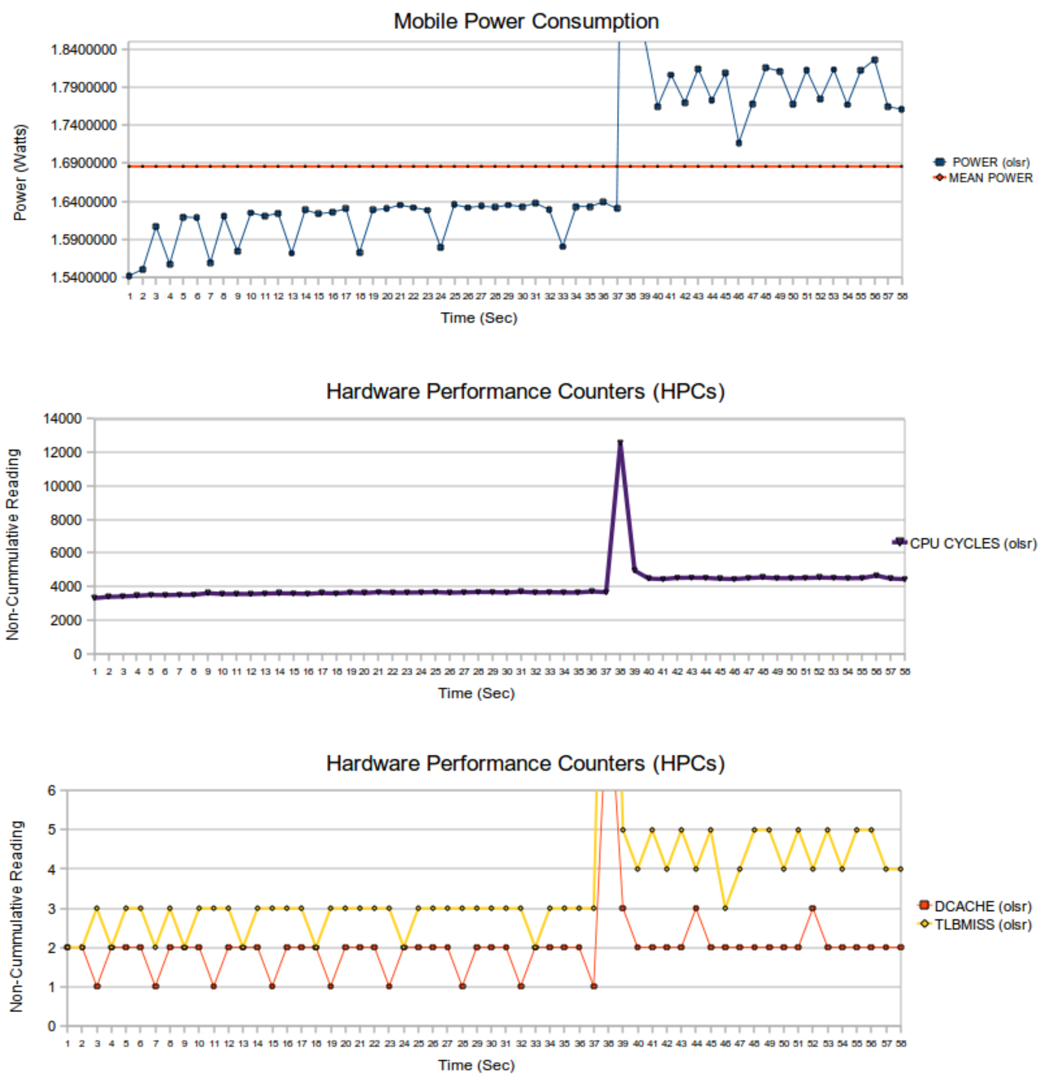


Figure 6.2: **Observations with OLSR daemon running: Power Consumption and HPC Monitoring**

while receiving a video stream. The video chosen was of low quality in 3GP format and was readily available in the server. A high video quality file was not used since the task of converting high quality videos to an acceptable resolution that the device supports was time consuming. The video was streamed at about 122 Kbps and downloaded at about 110 Kbps data rates.

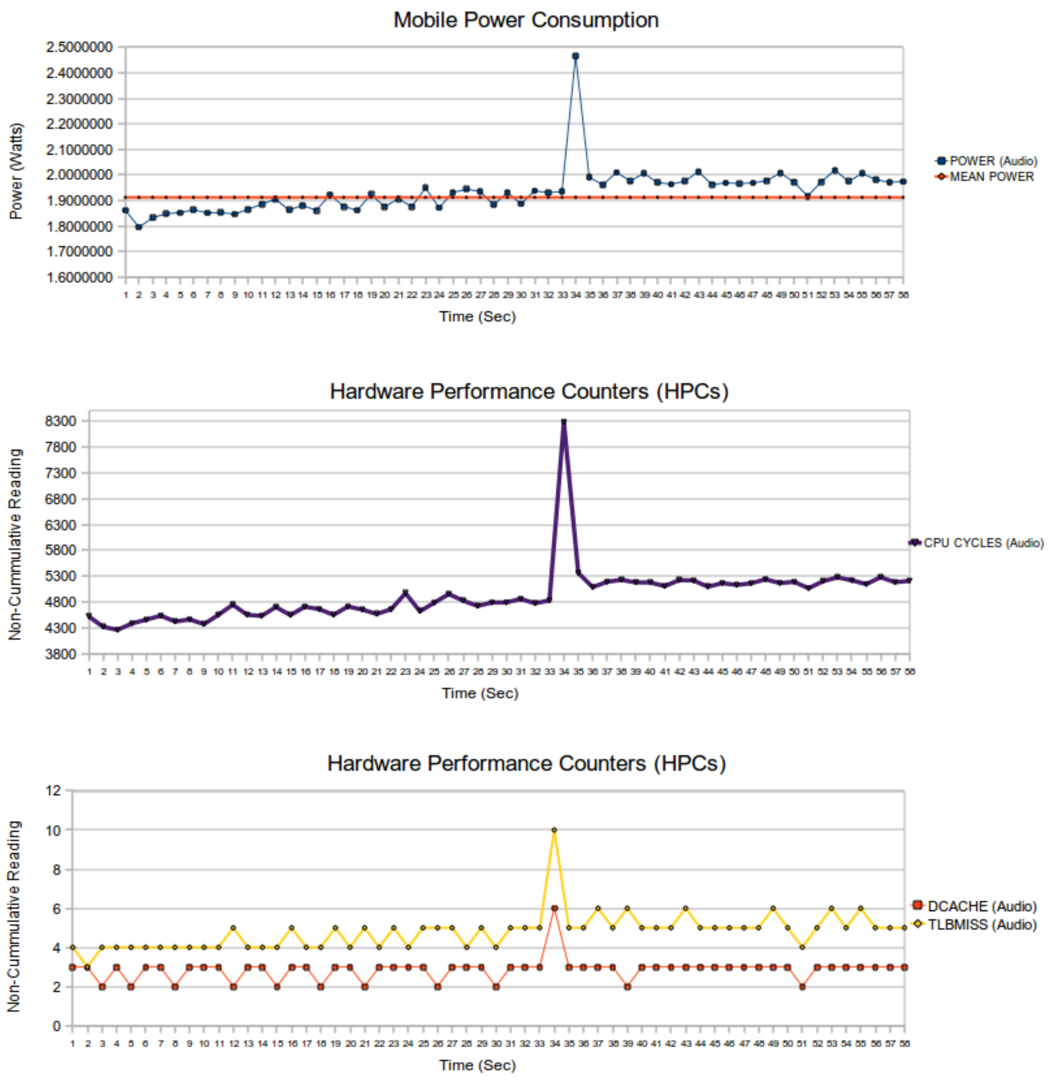


Figure 6.3: **Observations while streaming audio: Power Consumption and HPC Monitoring**

The mean power consumed was roughly 1.988 Watts which is a 3.96% increase compared to the power consumed for a audio stream and 25.36% increase compared to the idle state power consumption. It is with no doubt that a video with a higher resolution will consume slightly more power. The mobile device's brightness was set to the lowest value during this experiment and this parameter greatly determines the power consumed by the device while streaming video data.

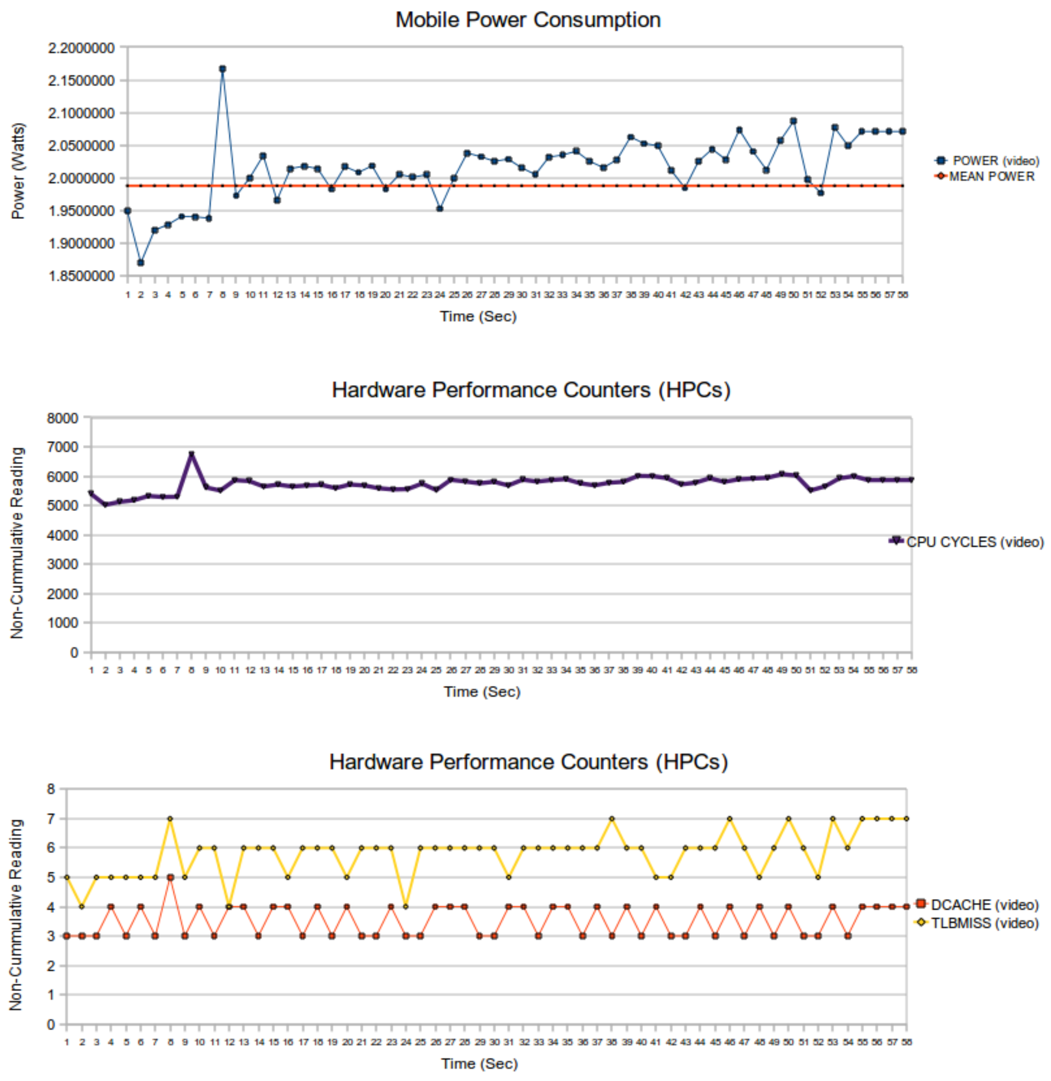


Figure 6.4: **Observations while streaming video: Power Consumption and HPC Monitoring**

There were several peaks that we observed while monitoring the HPCs and these peak values were way off from the average values we expect. We can assume that these peaks were a result of errors caused due to the periodic opreport dumping that puts heavy workload on the CPU.

6.6 Summary

Measurement of power consumption on a real world testbed shows that the mobile device consumes roughly 1.5858 Watts when in idle mode. With the olsr daemon running, the device consumes 6.31% more power which is about 1.6859 Watts. While streaming audio and video data, the device consumes 1.9121 Watts and 1.988 Watts respectively. For audio data, this is a 20.57% increase compared to the idle state power consumption and for video data the increase is roughly 25.36%. The brightness of the display was set to its lowest value during the experiment. Any increase in the brightness will result in the subsequent increase in the power consumption.

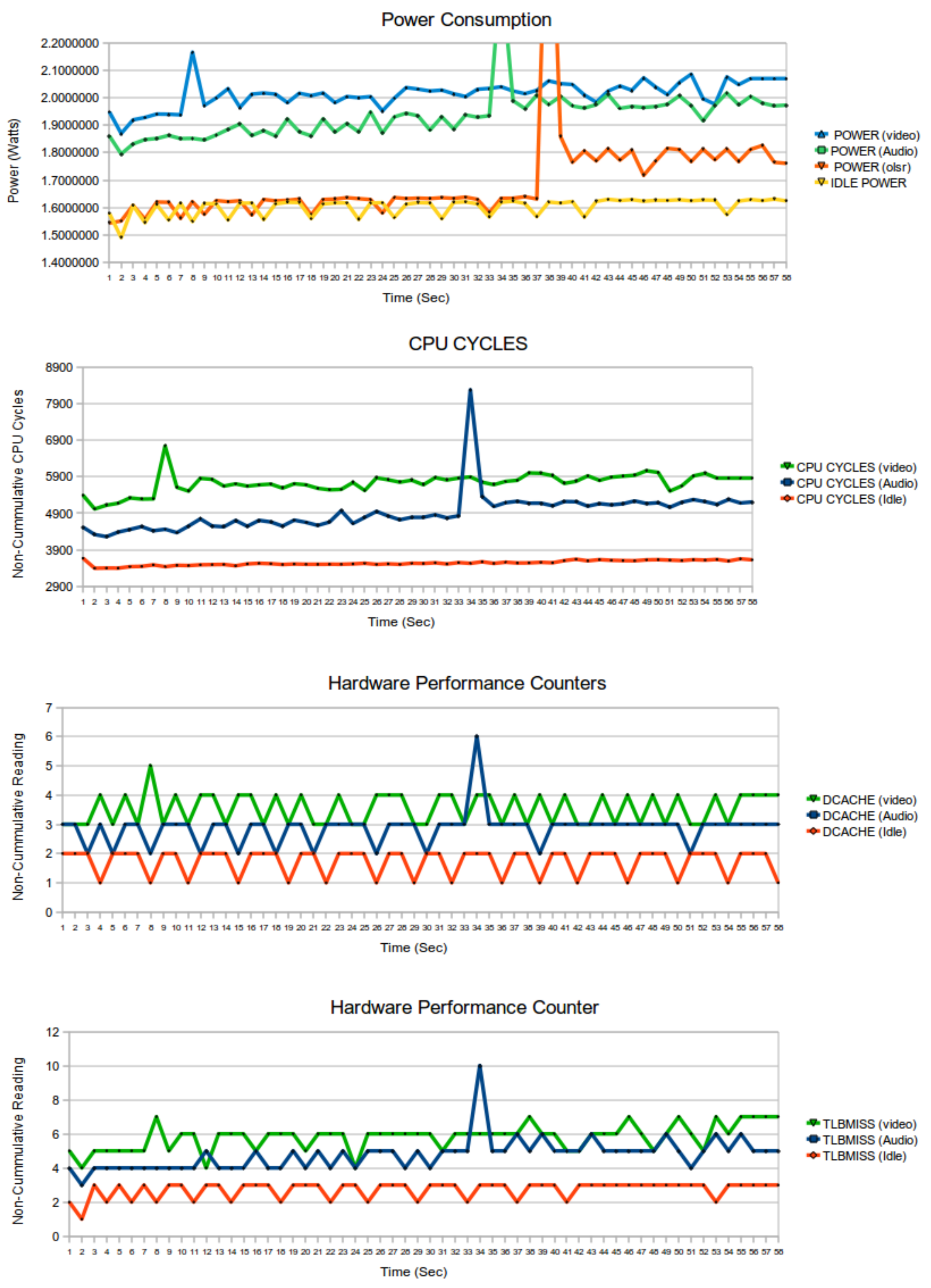


Figure 6.5: **Comparing results of Power Consumption and HPC Monitoring**

Chapter 7

Conclusion and Future Work

7.1 Conclusion

Today, hand held mobile devices are very popular and they come with rich functionality. Applications such as media streaming are trying to leverage all they can from such mobile devices. However, one of the major constrains is the battery life of these mobile nodes. The battery life is very limited and especially when such nodes are used in an ad hoc network for streaming data, they put high demands on the battery power. It is extremely important to maximize the efficient use of these contained resources. The optimization can occur in any layer of the OSI stack, however, this thesis work focuses on only the routing protocols used in the network layer. The influence of several external factors on the performance and energy consumption are also taken into consideration while performing the simulations and experiments. The results obtained from our observations provide both qualitative and quantitative analysis of the routing protocols. Furthermore, it also highlights how the behavior of the protocols are sometimes highly unpredictable, yielding results that we may not expect.

In this thesis work we have been able to evaluate the Energy Efficiency of the four most widely used MANET routing protocols (AODV, OLSR, DSDV and DSR) in terms of energy consumption and performance. The initial phase of the work was carried out using the Network Simulator 2(NS2) tool and towards the end a simple adhoc network was setup and the energy consumption was measured on a Nokia N810 Internet tablet. Our work was limited to the above stated protocols due to the lack of support for majority of the other protocols on NS2. Throughout all experiments the focus was to evaluate the energy measures under streaming like conditions. To the best of the author's knowledge, this is the first work that evaluates performance and energy consumption while streaming over mobile ad hoc networks. The simulations were conducted under several varying parameters such as transmission rates, mobility rates, traffic patterns, mobility patterns, network loads, etc. The results thus obtained would be more realistic than those obtained from simulation carried out under static environments that have non-varying parameters. Performance of the protocols has been evaluated using metrics such as throughput, normalized routing load, packet delivery fraction, etc while some of the energy metric that were used are total energy consumption in the network for each protocol, the energy expensed per byte of successfully received data, etc. It was interesting to observe that the results matched our expectations in many cases and were also contradictory to existing literature in many others. We have attempted to provide valid reasons for all behaviors that we observe for the four routing protocols that we have considered.

Several power models were taken into consideration while using simulations to compute the energy consumption. Of these, two models have been suggested in this thesis work and one was eventually used in calculating the energy consumption. Our choice of power model was influenced by the fact that the model as such was proposed for the Nokia N810 tablet's NIC. This model only accounted for the energy expended at the NIC and does not provide any information on the energy requirement at other layers of the stack.

The process of setting up the real world testbed and configuring the devices for streaming was a little challenging. However, all the installation and setup procedures have been clearly documented in this thesis work and can hopefully serve as a guidance tool for future works that need similar testbeds. The nodes in the testbed used OLSR protocol. The power model used to evaluate the power consumption of the mobile device took into account the three components that consume the maximum power in a streaming process, namely processor, display and network interface card.

7.2 Future Work

The set of protocols supported by NS2 was one of the major limitations during this thesis work. It would be interesting to study some of the 'Energy Aware' routing protocols either in simulation or in testbed if the support for any such protocol may be available in the near future. It would also be worth the effort to evaluate the benefits of using cross layer techniques in achieving higher energy efficiency. As far as the real world testbed is concerned, there were only two nodes in the ad hoc network. The results obtained from experiments on this testbed cannot be really extrapolated for larger networks. It would be wise to setup a larger and realistic MANET testbed for evaluating the energy metrics. Our testbed was limited to two nodes due to the unavailability of the necessary resources to setup a larger test environment. In addition, if the energy efficiency is to be measured in a MANET then one should think of a viable way to introduce mobility in the nodes. It is however hard to introduce true randomness in reality, into the traffic and mobility patterns.

This thesis work focused primarily on using the MANET for streaming purposes. We would recommend any future work along these lines to first evaluate energy efficiency for applications that are not as resource hungry as streaming. This suggestion comes due to the fact that, the streaming activity is so resource heavy that they hindered with the performance of softwares that were used for monitoring performance counters and thereby resulting in lots of errors. While using the Network Simulator tool we noticed that there were several hidden parameters that needed to be tweaked in order to mimic our required MANET environment and our node's hardware specification. These parameters are not directly obvious and unless someone stumbles on them by accident, it is hard to notice that these parameters were being used in the simulation with their default values. This might lead to unexpected results from our simulations. One must take care that all these parameters are identified before the simulation and are assigned the appropriate values right from the beginning. During our work we encountered several situations where we had to change the default parameter values.

"We strongly believe that understanding the energy behavior of our mobile devices is the key to enhancing the services that can be deployed on these devices. "

Appendix A

Configuring Softwares for the Nokia N810 tablet

A.1 Gaining Root access on the Nokia N810 tablet

Many of the operations that we perform on the mobile device will require that the user has local root access on the device. There is a package called 'rootsh'[16] which is available under the chinook extra repository that can be installed in order to obtain root access on the device. If the appropriate repository is added to the repository catalog then the rootsh package will appear in the application manager's Installable Applications list. After installing the package, a simple 'sudo gainroot' command in the terminal prompt will give the user root access.

A.2 Remote Login using OpenSSH

The SSH protocol allows devices to communicate with each other over an encrypted connection. This is quite useful during our experiment since it gives us the ability to either copy files to or from our N810 device using scp. It is also useful if the user intends to work on the device from another device through remote login. A SSH server must be installed on the device to which we are trying to connect and a SSH client must be available on the device that is trying to establish connection. Hence we need to install 'openssh-server'[1] on our Nokia N810 tablet. The package will be visible in the "Browse Installable Application" list if the 'maemo extras' repository is enabled in the repository catalog. During the installation of the package, the user is prompted to set a root password. Once the installation is complete, a system-wide key is generated and the ssh port 22 is opened. Now it will be possible to log into the device remotely by using the following command:

```
# ssh root@<ip address of the N810 tablet>
```

Both the mobile device and the device used for the remote login must be within the same network. The ip address of the N810 device can be one that is received from the DHCP or set manually by the user. When performing a remote login for the first time, the user will be asked if the device must add the new entity to the list of known hosts. On confirming this, the user can continue with working on the mobile device remotely.

A.3 Setting up ScratchBox

Scratchbox is a cross compilation tool which is used to build linux software and entire linux distributions [30]. With a cross compilation tool, one can use some host processor to compile software for another target processor that has a different architecture. Hence, it provides developers with an environment which is similar to their target environment even before the target environment is actually available to them. However, the software that is compiled will not work on the machine on which it is compiled since it is compiled for another processor with a different architecture. It is possible to use the Scratchbox tools from either inside or outside the scratchbox environment. Using the tools from the outside is equivalent to using the tools already available on the local host machine.

Before we can setup the Scratchbox environment, the following files must be downloaded from [6].

- scratchbox-core-1.0.17-i386.tar.gz
- scratchbox-libs-1.0.17-i386.tar.gz
- scratchbox-devkit-debian-1.0.6-i386.tar.gz
- scratchbox-devkit-cputransp-1.0.1-i386.tar.gz
- scratchbox-toolchain-cs2005q3.2-glibc-arm-1.0.5-i386.tar.gz

Now copy all the downloaded files to the root directory (/). An example is shown for copying one of the files to the root directory and the same has to be done for the rest of the files. Please note that copying to the root directory requires root privileges.

```
# cp home/user/Desktop/scratchbox-core-1.0.17-i386.tar.gz /
```

Create new '/myscratchbox' directory within the root and extract the files, that were previously copied, into the '/myscratchbox' directory within the root.

```
/myscratchbox# tar xfvz ../scratchbox-core-1.0.17-i386.tar.gz
/myscratchbox# tar xfvz ../scratchbox-libs-1.0.17-i386.tar.gz
/myscratchbox# tar xfvz ../scratchbox-devkit-debian-1.0.6-i386.tar.gz
```

Run the shell script 'run_me_first.sh' that is within the scratchbox folder. You will be prompted for some information, but simply click the return key to leave the values in their default ones.

```
/myscratchbox# scratchbox/run_me_first.sh
Do you want to use sudo mode? [yes/no] (no):
Give the name of the scratchbox group (sbox):
Stopping Scratchbox: umount, binfmt_misc.
Starting Scratchbox: binfmt_misc, mount.

Now you should add one or more users with
/myscratchbox/scratchbox/sbin/sbox_adduser
```

Add an user to the scratchbox. This user must already exist on the local machine.

```
root@system:/myscratchbox# scratchbox/sbin/sbox_adduser guestuser
Scratchbox user account for user guestuser added
```

Make sure that 'sbox_ctl' is started every time you want to use Scratchbox.

```
$ sudo ln -s /scratchbox/sbin/sbox_ctl /etc/rc2.d/S20scratchbox
```

Download and copy the arm dev platform files to the /scratchbox/packages/ directory. Then login to scratchbox and launch 'sb-menu' from within.

```
# cp ../home/user/Desktop/Maemo_Dev_Platform_v2.2_armel-rootstrap.tgz
                /myscratchbox/scratchbox/packages/
```

A.4 Configuring armel toolchain

Once the Scratchbox environment is setup, the armel toolchain must be setup next. Launch the sb-menu in the Scratchbox environment and follow the steps provided below:

- Select the Setup option and choose 'Create a new Target'
- In the textbox that appears, set the name to SDK_ARMEL
- Choose the compiler 'cs2005q3.2-glibc-arm' and then choose debian and cputrans as the devkits. Now select 'Done'
- For CPU transparency select 'qemu-arm-0.8.1-sb2'
- In the prompt that follows, select Yes for extracting the rootstrap for the current target and then provide the complete path for the Maemo rootstrap 2.2 for armel.
- At this stage, the user is prompted to install a few more files. Choose Yes and select all the checkboxes that appear in the window. Now simply Confirm to complete the installation

A.5 Flashing the N810's kernel

In order to be able to install oprofile on the N810 tablet, we need a kernel that supports oprofile. The kernel can be downloaded and compiled in the scratchbox environment that was setup previously. Login to the scratchbox environment and update the package list as shown below. Then install 'rx-34-kernel-oprofile'. This will install the kernel image to the /usr/share/osso/kernels/ directory. At the time of this experiment the version of the oprofile supporting kernel that was available is 2.6.21.0-200749osso2.

```
[sbox-CHINOOK_ARMEL:~] > apt-get update
[sbox-CHINOOK_ARMEL:~] > fakeroot apt-get install rx-34-kernel-oprofile
[sbox-CHINOOK_ARMEL:~] > ls -l /usr/share/osso/kernels/
total 1508
-rw-r--r-- 1 guestuser guestuser 1536876 Dec 7 2007
                                     zImage-oprofile-rx-34-200749
```

By copying the image to outside of the scratchbox chroot environment, you'll be able to easily access it when you're ready to flash the image. Download the flasher-3.0 from the website: http://tablets-dev.nokia.com/maemo-dev-env-downloads.php. Now copy the flasher-3.0 and the new image file compiled using scratchbox into a /tmp folder on your workstation. Proceed with the flashing of the device as indicated below. The user will be prompted to connect the N810 device to the workstation using a USB port and this has to be done in order to complete the flashing process.

```
$ sudo tmp/maemo_flasher-3.5_2.5.2.2/flasher-3.5 -f --kernel
                          tmp/zImage-oprofile-rx-34-200749 -R

Suitable USB device not found, waiting
USB device found found at bus 002, device address 008
Found device RX-44, hardware revision 0805
NOLO version 1.1.7
Version of 'sw-release': RX-44_2008SE_2.2007.50-2_PR_MR0
Sending kernel image (1500 kB)...
100% (1500 of 1500 kB, avg. 15634 kB/s)
Flashing kernel... done.
```

A.6 Installing Oprofile for the N810 device

Oprofile [4] is a low overhead system-wide profiler for linux which can be used to measure hardware performance counters such as CPU usage, either for the whole system or for individual processes within the system. Oprofile can be installed on a device only if its kernel supports it. All unsupported kernels must be flashed with another kernel that supports it. If the Chinook tools repository is enabled in the catalog list on the mobile device then the user can install Oprofile from the terminal using the apt command.

```
Nokia-N810:~# apt-get install oprofile
```

Now that Oprofile is installed on the device, the user can measure the required hardware performance counters. Start Opcontrol when the monitoring of the performance counters must start and stop it when the monitoring must end. The data that is collected can be viewed using the opreport command.

```
Nokia-N810:~# opcontrol --init
Nokia-N810:~# opcontrol --reset
Nokia-N810:~# opcontrol --setup --event CPU_CYCLES:100000
                  --event DCACHE_WB:100000 --event TLB_MISS:100000
Nokia-N810:~# opcontrol --start
<After the required period of monitoring issue the stop command>
Nokia-N810:~# opcontrol --stop
```

A snippet of the output from the opreport is provided below. The report contains the measure of three hardware performance counters, namely CPU_CYCLES, DCACHE_WB and TLB_MISS for the various processes that are running in the system.

```
CPU: ARM V6 PMU, speed 0 MHz (estimated)
Counted CPU_CYCLES events (clock cycles counter) with a unit
mask of 0x00 (No unit mask) count 100000
Counted DCACHE_WB events (data cache writeback, 1 event for every
half cacheline ) with a unit mask of 0x00 (No unit mask) count 100000
Counted TLB_MISS events (Main TLB miss ) with a unit
mask of 0x00 (No unit mask) count 100000
CPU_CYCLES:100000| DCACHE_WB:100000|  TLB_MISS:100000|
  samples|       %|  samples|       %|  samples|       %|
-------------------------------------------------------
     6830  62.5974          2  33.3333          3  50.0000 no-vmlinux
     1472  13.4910          1  16.6667          1  16.6667 ld-2.5.so
     1187  10.8789          2  33.3333          1  16.6667 libc-2.5.so
      908   8.3219          0        0          1  16.6667 busybox
      451   4.1334          1  16.6667          0        0 oprofiled
       41   0.3758          0        0          0        0 ophelp
```

A.7 OLSRd support

OLSRd (OLSR daemon)[7] is an implementation of the Optimized Link State Routing protocol. It is an proactive ad-hoc routing protocol and it works fine on any wifi card that can operate in ad-hoc mode. OLSRd has been tested on Linux and other operating systems, and it has worked fine without any major errors. There is also an OLSRd package that is available for the N810 tablet device that we use in our testbed. OLSR is said to be fast and it consumes very little CPU time. This aids in saving critical battery life on mobile devices that are normally constrained by their low energy resource. The protocol has been tested in a network that has had over 2000 nodes and this makes OLSR highly scalable.

For the mobile device the olsrd package can be found in the application manager. The package is automatically installed with a single click. For the workstation, the latest release of the protocol implementation can be found in the website: http://www.olsr.org/?q=download. The tar file has to unpacked, compiled and then installed.

```
# tar zvfx uolsrd-0.5.6-r4
# cd unik-olsrd-0.5.6-r4
# make
# make install
```

Once the installation is complete, the configuration file olsrd.conf can be found within the directory /etc. The configuration file must be edited to match our requirement and testbed setup. The values in the configuration file can also be overridden with the command line options that go along with the olsrd command. Some of the values that need the user's attention are the Debug Level, IP Version and the interface on which the protocol will be functioning. Initially the Debug Level can be set to a non-zero value so that the protocol runs in the forefront and not in the background. This will help the user figure out if the protocol has been configured properly and if it discovers other nodes within the network.

```
# Debug level(0-9)
# If set to 0 the daemon runs in the background
DebugLevel      1

# !!CHANGE THE INTERFACE LABEL(s) TO MATCH YOUR INTERFACE(s)!!
#Interface "eth1" "eth0" "wlan0" "wlan1" "ath0" "ath1"
Interface "wlan0"

# IP version to use (4 or 6)
IpVersion       4
```

After configuring OLSRd, it can be started by simply issuing the olsrd command in the terminal. As mentioned earlier, the configuration parameters can be given at the command line. Once this is done, the OLSR daemon begins to listen on the specified interface.

```
# olsrd -i wlan0 -d 1 -ipv4
```

A.8 Streaming Application

For our experiment, we would be streaming data from a server to a client node. Both nodes will be a part of an ad-hoc environment which uses the OLSR protocol. Creating such an experimental setup would require an application that is capable of streaming data from the server node and another client-end application that can receive and play this stream.

The workstation which acts as the server in our setup, has MediaTomb installed in it. MediaTomb [19]is an open source Universal Plug and Play (UPnP) media server which comes with a web based user interface that offers high flexibility in configuring and controlling the behavior of various features offered by the media server. It runs on all major architectures: x86, Alpha, ARM, MIPS, etc and supports multiple operating systems: Linux, FreeBSD, Mac OS X, etc. UPnP protocols facilitate devices in the process of discovering one another within the network and to seamlessly connect with each other without the need for any form of manual configurations by the user. Hence with an UPnP media server, a client node can easily discover the server, seamlessly connect to it and stream any shared media data, may it be audio or video. After MediaTomb is installed in the workstation, the user needs to open up the software and choose all the media files that the user wishes to share. This application however does not give the user the option of choosing the bit rate at which the media stream is uploaded.

The client node which receives the stream in our setup is our Nokia N810 tablet. It has 'Media Stream' application which is available in the application manager installed in it. This client-end application is capable of discovering all UPnP media servers present in the network. The user simply has to choose the right server, connect to it and stream any data that is shared by the media server.

Appendix B

Scripts

B.1 Main OTcl Script

```
#Define the components of the wireless node


set val(chan)   Channel/WirelessChannel ;# channel type

# radio propagation model—Two Ray Ground(far distance attenuation)
#/shadowing model/friss-space mode(near attenuation)
set val(prop)   Propagation/TwoRayGround ;

# Antenna Type (Omni Antenna OR Unity gain)
set val(ant)    Antenna/OmniAntenna     ;

set val(netif)  Phy/WirelessPhy         ;# network interface type
set val(mac)    Mac/802_11              ;# MAC type

# interface queue type Queue/DropTail/PriQueue or CMUPriQueue
set val(ifq) Queue/DropTail/PriQueue    ;

set val(ifqlen) 500     ;# maximum packets in the interface queue
set val(ll)     LL      ;# link layer type
set val(nn)     30      ;# Number of mobile nodes
set val(rp)     DSDV    ;# routing protocol (DSDV, AODV, DSR, OLSR)
set val(x)      1000    ;# length of the topography
set val(y)      600     ;# breadth of the topography
set val(stop)   800     ;# simulation period

# loading the scenario file created using ns2's setdest
set val(sc)     "scen-p0" ;
```

```
# udp-cbr traffic with maximum 8 connections in the simulation
# and 200kbps bitrate (50 pkts/sec @ 512Bytes each)
set val(cp)             "cbr-mc8-512"    ;

set val(energymodel)    EnergyModel      ;
set val(initialenergy)  5000    ;# Initial energy in Joules


Antenna/OmniAntenna set X_ 0
Antenna/OmniAntenna set Y_ 0
Antenna/OmniAntenna set Z_ 1.5
Antenna/OmniAntenna set Gt_ 1.0
Antenna/OmniAntenna set Gr_ 1.0

Phy/WirelessPhy set CPThresh_ 10.0
Phy/WirelessPhy set CSThresh_ 1.0e-10
Phy/WirelessPhy set RXThresh_ 1.296e-10 ;#1.559e-11
Phy/WirelessPhy set Pt_ 0.1
Phy/WirelessPhy set freq_ 9.14e+08
Phy/WirelessPhy set L_ 1.0

set tracefile [open DSDV_Sim_1.tr w]
set namtrace [open out.nam w]

set ns_ [new Simulator]
$ns_ trace-all $tracefile
$ns_ namtrace-all-wireless $namtrace $val(x) $val(y)

#set up topography object

        set topo [new Topography]
        $topo load_flatgrid $val(x) $val(y)
        set god_ [create-god $val(nn)]

#configure the mobile nodes

$ns_ node-config -adhocRouting $val(rp) \
                -macType $val(mac) \
                -ifqType $val(ifq) \
                -ifqLen $val(ifqlen) \
                -llType $val(ll) \
                -antType $val(ant) \
                -propType $val(prop) \
                -phyType $val(netif) \
                -channel [new $val(chan)] \
                -topoInstance $topo \
                -energyModel $val(energymodel) \
                -rxPower 1.181 \
                -txPower 1.258 \
                -idlePower 0.884 \
                -sleepPower 0.042 \
                -transitionTime 0.005 \
                -initialEnergy $val(initialenergy)\
                -agentTrace ON \
                -routerTrace ON \
                -macTrace ON \
                -movementTrace OFF

$ns_ set WirelessNewTrace_ ON


# create [val(nn)] nodes
for {set i 0} {$i<$val(nn)} {incr i} {
set node_($i) [$ns_ node]
$node_($i) random-motion 0 ;# disable random motion
}



#
# Define node movement model
#
#puts "Loading scenario file..."
source $val(sc)

#
# Define traffic model
#
#puts "Loading connection pattern..."
source $val(cp)



#define nodes's initial position in nam
for {set i 0} {$i<$val(nn)} {incr i} {
$ns_ initial_node_pos $node_($i) 30
 }

# telling nodes when the simulation ends
for {set i 0} {$i<$val(nn)} {incr i} {
$ns_ at $val(stop).000000001 "$node_($i) reset";
}

#ending nam and the simulation
$ns_ at $val(stop) "$ns_ nam-end-wireless $val(stop)"
$ns_ at $val(stop) "finish"
$ns_ at $val(stop).000000001 "puts \"NS EXITING...\" ; $ns_ halt"
```

```
proc finish {} {
global ns_ tracefile namtrace
$ns_ flush-trace
close $tracefile
close $namtrace
exec nam out.nam &
}

$ns_ run
```

B.2 Avg. Throughput of all connections

```
{
if ($0 ~/AGT/) {
   event = $1
   time = $2;
   node = $3;
   packetsize = $8;
   to=$15;

   sub(/^_*/, "", node);
  #substitue the occurance of "_" in the $3 with a empty string ""
  # This way we can strip of the unwanted
  # 'leading' underscore we have.

   sub(/_/, "", node);
  # This way we can strip of the unwanted
  # 'trailing' underscore we have.

   if ( event == "s" )
        {
        # to_left=$15;
        # sub(/^:.*/, "", to_left);
        # to_right=$15;
        # sub(/^:/, "", to_right);
        if ( time < start_time[to] || start_time[to] == 0 )
                {
                start_time[to] = time;
                #print to, start_time[to];
                }
        }
   else if ( event == "r" )
        {
        if ( time > end_time[to] )
                {
                end_time[to] = time;
                total_bytes[to] += packetsize;
```

```
                    }
          }
}
}
END {
accumulated_throughput=0;
count=0;
print ("\n ---------------------------------------------------------");
for ( to in start_time )
{
count+=1;
if ( total_bytes[to] > 0 )
          {
          tdelta = end_time[to] - start_time[to];
          accumulated_throughput+=(total_bytes[to]*8/(1024*tdelta));
          printf("%s %d %f %f %f %f\n",
                    to,
                    total_bytes[to],
                    end_time[to],
                    start_time[to],
                    tdelta,
                    total_bytes[to]*8 / (1024*tdelta));
          }
}
printf ("\n Average throughput: %f",
          ((accumulated_throughput)/count));
printf ("\n %f", ((accumulated_throughput)/count))
          >> "OLSR_throughput";
print ("\n ---------------------------------------------------------");
}
```

B.3 Overall Packet Delivery Fraction and Normalized Routing Load

Only in the case of DSDV, replace 'AODV' with 'message/'. In rest other cases replace with the appropriate name of the protocol (OLSR, DSR or AODV)

```
# Performance metric for AODV

BEGIN {
          send = 0;
          recv = 0;
          forward = 0;
          dropped = 0;
          routing_packets = 0;
          }
```

```
{
        if ($0 ~/^s.* AGT/) {send ++ ;}
        if ($0 ~/^r.* AGT/) {recv ++ ;}
        if ($0 ~/^f.* RTR/) {forward ++ ;}
        if ($0 ~/^D.* cbr/) {dropped ++ ;}
        if (($0 ~/^s.* RTR.* AODV/) || ($0 ~/^f.* RTR.* AODV/))
                {routing_packets ++ ;
                # count total number of routing packets
                #(includes both sent and forwarded pkts)
                }
}
END { print "\n", "Total_Sent:",send,
        "Total_Received:",recv,
        "Packet_Delivery_Fraction:",((recv/send)*100),
        "Total_Forwarded:",forward,
        "Total_dropped:",dropped,
        "Total_Routing_Packets:", routing_packets,
        "Normalized_Routing_Load:", (routing_packets/recv) ;
        }
```

B.4 Packet Delivery Fraction as a function of Time

```
BEGIN {
        send = 0;
        recv = 0;
        forward = 0;
        threshold= 50;
        }
{
time=$2 ;
if (time<=threshold)
        {
        if ($0 ~/^s.* AGT/) {send ++;}
        if ($0 ~/^r.* AGT/) {recv ++ ;}
        # if ($0 ~/^f.* RTR/) {forward ++ ;}
        }
else if (time>threshold && time<800.0)
 {
# print "\n", time, send, recv, (recv/send)*100, forward;
print time, (recv/send)*100;
threshold+=50;
  if (time<=threshold)
        {
        if ($0 ~/^s.* AGT/) {send ++;}
        if ($0 ~/^r.* AGT/) {recv ++ ;}
        # if ($0 ~/^f.* RTR/) {forward ++ ;}
        }
 }
}
END {print time, (recv/send)*100;
}
```

B.5 Total Energy Consumption of all nodes in the network

```


{
if ($0 ~/N/) {
   energy_left = $7
   energy_consumed = 5000 - energy_left
   time = $3;
   node_id = $5;


   if ( node_energy[node_id] <= energy_consumed ) {
        node_energy[node_id] = energy_consumed ;
        }
}}

END {
totalEnergyConsumed=0;
#print ("\n --------------------------------------------------");
for ( i=0; i<=29; i++ )
{
        if ( node_energy[i] > 0 )
         {
                totalEnergyConsumed+=node_energy[i];
                printf("%d %f \n", i, node_energy[i]);
     }
}
printf ("\n%f", (totalEnergyConsumed));
#print ("\n --------------------------------------------------");
}
```

B.6 Script to instantiate oprofile and take periodic opreport dumps

```
#!/bin/sh
```

```
opcontrol --init
opcontrol --reset
opcontrol --setup --event CPU_CYCLES:100000 --event DCACHE_WB:100000
                                           --event TLB_MISS:100000

opcontrol --start
opcontrol --dump

opreport  >test/0.txt

for j in `seq 60`
do
        end_name="$j".txt
        echo $end_name
                # ./sleep_ms_15
        opcontrol --dump
        opreport --no-header > test/$end_name
done

opcontrol --stop
opcontrol --shutdown
```

Bibliography

[1] About OpenSSH. http://www.openssh.com/.

[2] Technical specifications, Nokia N810 Internet Tablet support. http://europe.nokia.com/support/product-support/nokia-n810/specifications.

[3] IEEE Std 802.11-2007 - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, Jun 2007. http://standards.ieee.org/getieee802/download/802.11-2007.pdf.

[4] Oprofile Official Website, About Oprofile. http://oprofile.sourceforge.net/about/, Nov 2009.

[5] ITU-T: Telecommunication Standardization Sector of ITU, Series H: Audiovisual and Multimedia Systems, Advanced video coding for generic audiovisual services , March 2010.

[6] Andrea Grandi. Maemo Geek Blog, Installing Maemo SDK for Nokia 770. http://maemogeek.blogspot.com/2007/02/installing-maemo-sdk-for-nokia-770.html, Feb 2007.

[7] Andreas TÃ̃ÿnnesen and Thomas Lopatic and Hannes Gredler and Bernd Petrovitsch and Aaron Kaplan Sven-Ola TÃ̃ijcke and others. olsrd: an ad hoc wireless mesh routing daemon. http://www.olsr.org/?q=about.

[8] Muhammad Azhar Amin bin Mansor and Jiwa bin Abdullah. Evaluating the Communication performance of an Ad Hoc Wireless Network using Mesh Connectivity Layer (MCL) Protoco . *Journal of Basic and Applied Sciences*, 1(3), Oct 2009.

[9] Paul E. Black. Bellman-Ford algorithm, in Dictionary of Algorithms and Data Structures [online], U.S. National Institute of Standards and Tech-

nology., March 2005. http://www.itl.nist.gov/div897/sqg/dads/HTML/bellmanford.html.

[10] Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu, and Jorjeta Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *MobiCom '98: Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*, pages 85–97, New York, NY, USA, 1998. ACM.

[11] Juan-Carlos Cano and Pietro Manzoni. A performance comparison of energy consumption for mobile ad hoc network routing protocols. In *MASCOTS '00: Proceedings of the 8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, page 57, Washington, DC, USA, 2000. IEEE Computer Society.

[12] Ian Chakeres and Charles Perkins. Dynamic MANET On-Demand (DYMO) Routing. Internet-Draft (work in progress) draft-ietf-manet-dymo-10.txt, July 2007.

[13] Benjie Chen, Kyle Jamieson, Hari Balakrishnan, and Robert Morris. SPAN: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. In *ACM Wireless Networks Journal*, pages 85–96, 2001.

[14] T. Clausen and P. Jacquet. RFC 3626:Optimized Link State Routing Protocol (OLSR), Oct 2003. http://www.ietf.org/rfc/rfc3626.txt.

[15] S. Doshi and T. Brown. Minimum energy routing schemes for a wireless ad hoc network, 2002.

[16] Faheem Pervez. Rootsh - Get root easily. v1.4. http://maemo.org/downloads/product/OS2008/rootsh/, Oct 2008.

[17] Laura Marie Feeney. An energy consumption model for performance analysis of routing protocols for mobile ad hoc networks. *Mob. Netw. Appl.*, 6(3):239–249, 2001.

[18] Girling G, Wa JLK, Osborn P, and Stefanova R. The design and implementation of a low power ad hoc protocol stack. In *IEEE Wireless Communications and Networking Conference*, volume 4, pages 8–15, 1997.

[19] Gena Batyan and Sergey Bostandzhyan and Leonhard Wimmer. MediaTomb Official Website: Welcome to the MediaTomb website. http://mediatomb.cc/.

[20] S. Giannoulis, C. Antonopoulos, E. Topalis, A. Athanasopoulos, A. Prayati, and S. Koubias. TCP vs. UDP Performance Evaluation for CBR Traffic On Wireless Multihop Networks. 2009.

[21] Jae hwan Chang and Ros Tassiulas. Energy Conserving Routing in Wireless Ad-hoc Networks. pages 22–31, 2000.

[22] D. Johnson, Y. Hu, and D. Maltz. RFC 3550:The Dynamic Source Routing Protocol(DSR) for Mobile Ad Hoc Networks for IPv4, Feb 2007. `http://www.ietf.org/rfc/rfc4728.txt`.

[23] David B. Johnson and David A. Maltz. Dynamic Source Routing in ad-hoc wireless networks. In *Mobile Computing*, pages 153–181. Kluwer Academic Publishers, 1996.

[24] Vijay Kakadia, Wei Ye., and Padma Haldar. Energy Model Update in ns-2. `http://www.isi.edu/ilense/software/smac/ns2_energy.html`, Jun 2005.

[25] Ted Taekyoung Kwon and Mario Gerla. An ip-level mobility management framework based on quasi-registration in wireless technologies convergence, May 2002.

[26] Benjamin C. Lee and David M. Brooks. Accurate and efficient regression modeling for microarchitectural performance and power prediction. In *ASPLOS-XII: Proceedings of the 12th international conference on Architectural support for programming languages and operating systems*, pages 185–194, New York, NY, USA, 2006. ACM.

[27] Q. Li, J. Aslam, and D. Rus. Online Power-Aware Routing in Wireless Ad Hoc Networks.

[28] Cong Lin, Yong Xiang, and Heming Cui. A multi-rate mac protocol for mobile ad hoc networks and its cooperative extension. In *WICON '08: Proceedings of the 4th Annual International Conference on Wireless Internet*, pages 1–9, ICST, Brussels, Belgium, Belgium, 2008. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[29] Shu lin Wu, Yu-Chee Tseng, and Jang ping Sheu. Intelligent medium access for mobile ad hoc networks with busy tones and power control. *IEEE Journal on Selected Areas in Communications*, 18:1647–1657, 2000.

[30] Mankinen and Rahkonen. Cross-Compiling tutorial with Scratchbox. `http://www.scratchbox.org/documentation/user/scratchbox-1.0/html/tutorial.html`, Apr 2005.

[31] The Network Simulator NS-2. `http://www.isi.edu/nsnam/ns/`.

[32] S. Corson V. Park. Temporally-Ordered Routing Algorithm (TORA) Version 1 Functional Specification. `http://tools.ietf.org/html/draft-ietf-manet-tora-spec-04`.

[33] Charles E. Perkins, Elizabeth M. Belding-Royer, and Samir R. Das. Ad hoc On-Demand Distance Vector (AODV) Routing. RFC Experimental 3561, Internet Engineering Task Force, July 2003. `http://rfc.net/rfc3561.txt`.

[34] Charles E. Perkins and Pravin Bhagwat. Highly dynamic Destination-Sequenced Distance-Vector routing (DSDV) for mobile computers. *SIGCOMM Comput. Commun. Rev.*, 24(4):234–244, 1994.

[35] Francisco J. Ros and Pedro M. Ruiz. Introduction to DYMOUM. `http://masimum.dif.um.es/?Software:DYMOUM:Introduction`.

[36] Francisco J. Ros and Pedro M. Ruiz. Introduction to UM-OLSR. `http://masimum.dif.um.es/?Software:UM-OLSR:Introduction`.

[37] Martin Roth and Stephen Wicker. Termite: A Swarm Intelligent Routing Algorithm for Mobile Wireless Ad-Hoc Networks, Wireless Intelligent Systems Laboratory - Cornell University. **http://cone.informatik.uni-freiburg.de/lehre/seminar/adhoc-s08/literature.html**.

[38] Elizabeth M. Royer and Chai-Keong Toh. A review of current routing protocols for ad-hoc mobile wireless networks. *IEEE Personal Communications*, 6(2):46–55, Apr 1999.

[39] Yongsheng Shi and T. Aaron Gulliver. An energy-efficient mac protocol for mobile ad hoc networks. In *CNSR '06: Proceedings of the 4th Annual Communication Networks and Services Research Conference*, pages 76–88, Washington, DC, USA, 2006. IEEE Computer Society.

[40] Harkirat Singh and Suresh Singh. Energy consumption of TCP Reno, Newreno, and SACK in multi-hop wireless networks. In *SIGMETRICS '02: Proceedings of the 2002 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 206–216, New York, NY, USA, 2002. ACM.

[41] Karan Singh, Major Bhadauria, and Sally A. McKee. Real time power estimation and thread scheduling via performance counters. *SIGARCH Comput. Archit. News*, 37(2):46–55, 2009.

[42] Suresh Singh, Mike Woo, and C. S. Raghavendra. Power-aware routing in mobile ad hoc networks. In *MobiCom '98: Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*, pages 181–190, New York, NY, USA, 1998. ACM.

[43] Ivan Stojmenovic and Xu Lin. Power-aware localized routing in wireless networks. *IEEE Trans. Parallel Distrib. Syst.*, 12(11):1122–1133, 2001.

[44] Liansheng Tan, Peng Yang, and Sammy Chan. Error-aware and energy-efficient routing approach in manets. *Int. J. Commun. Syst.*, 22(1):37–51, 2009.

[45] C. K. Toh. Maximum battery life routing to support ubiquitous mobile computing in wireless ad hoc networks. *Communications Magazine, IEEE*, 39(6):138–147, August 2002.

[46] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra. Overview of the h.264/avc video coding standard. In *IEEE Transactions on Circuits and Systems for Video Technology*, volume 13, pages 560–576, July 2003.

[47] Kyungtae Woo, Chansu Yu, Dongman Lee, Hee Yong Youn, and Ben Lee. Non-blocking, localized routing algorithm for balanced energy consumption in mobile ad hoc networks. In *MASCOTS '01: Proceedings of the Ninth International Symposium in Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, page 117, Washington, DC, USA, 2001. IEEE Computer Society.

[48] Yu Xiao, Rajubrata Bhaumik, Zhirong Yang, Matti Siekkinen, Petri Savolainen, and Antti Yla-Jaaski. A System-level Model for Runtime Power Estimation on Mobile Devices.

[49] Yu Xiao, Petri Savolainen, Arto Karppanen, Matti Siekkinen, and Antti Yla-Jaaski. Practical power modeling of data transmission over 802.11g for wireless applications. In *Proceedings of the First ACM International Conference on Energy-Efficient Computing and Networking. e-Energy 2010.* ACM, Apr 2010.

[50] Ya Xu, John Heidemann, and Deborah Estrin. Geography-informed energy conservation for Ad Hoc routing. In *MobiCom '01: Proceedings of*

the 7th annual international conference on Mobile computing and networking, pages 70–84, New York, NY, USA, 2001. ACM.

[51] Peng Yang, Liansheng Tan, and Sammy Chan. An error-aware and energy-efficient routing protocol in manets. *16th International Conference on Computer Communications and Networks (ICCCN 2007).*

[52] Jungkeun Yoon, Mingyan Liu, and Brian Noble. Random waypoint considered harmful. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies* , volume 2, pages 1312–1321, Jul 2003.