University of Twente
The Netherlands

Power Awareness Routing

in Mobile Ad Hoc Networks

by

Johnny de Vries

Master of Science Thesis

University of Twente

Electrical Engineering

Design and Analysis of Communication Systems

27 August 2004

Examination committee:
Prof. Dr. Ir. Boudewijn Haverkort
Dr. Ir. Georgios Karagiannis (first supervisor)
Dr. Ir. Sonia Heemstra de Groot

Abstract

Currently, one of most innovative topics in computer communications is wireless networking. One area in wireless networking is ad hoc networking. The concept of ad hoc networking is based on the fact that users can communicate with each other using a mobile wireless network, without any form of centralized administration. Mobility with potentially very large number of mobile nodes, and limited resources (like bandwidth and power) make routing in ad hoc networks extremely challenging. Routing protocols for wireless ad hoc networks have to adapt quickly to the frequent and unpredictable changes of routing topology and must minimize the generated overall network overhead. To deal with these issues a large number of different routing protocols for ad hoc networking are developed, each with their own features and characteristics.

In a mobile ad hoc network, nodes are often powered by batteries. The power level of a battery is finite and limits the lifetime of a node. Every message sent and every computation performed drains the battery. One solution for power conservation in mobile ad hoc network is power awareness routing. This means that routing decisions made by the routing protocol should be based on the power-status of the nodes. Nodes with low batteries will be less preferably for forwarding packets than nodes with full batteries, thus increasing the life of the nodes. A routing protocol should try to minimize control traffic, such as periodic update messages to improve the lifetime of the nodes and network. However, not every routing protocol is suitable for implementing power awareness routing and different approaches on power awareness routing can be followed.

In this thesis a Power Awareness Routing prototype implementation is presented. Power Awareness Routing is implemented into the OSPF daemon of the GNU Zebra routing software. A battery lifetime model, based on a discrete-time model for batteries is used to emulate the battery lifetime cycle of a wireless device. Two different scheme for generating Link State Updates in the power awareness OSPF routing prototype (the *Power-awareness LSA update scheme* and the *Enhanced Power-awareness LSA update scheme*) are presented and implemented into the prototype.

With the User-Mode Linux solution a virtual network has been build on a single host computer. This virtual network is used to perform three experiments on the power awareness routing prototype. The results of these experiments have shown that the power awareness routing implementation in zebra is working correctly and can be implemented into a Linux environment. Moreover the experiments have shown that when the battery lifetime models are known, the operation of the power awareness routing scheme can be influenced such that the generated overall network overhead is significantly reduced. Furthermore it shows that User-Mode Linux can be used to create virtual networks that emulate real network scenarios, where network experiments can efficiently be performed.

Table of contents

1 Introduction

1.1 Background

Currently, one of most innovative topics in computer communications is mobile wireless networking. Recent technological advances in wireless data communication devices and laptops have lead to lower prices and higher data rates. This offers users new applications in mobile computing and has lead to a rapid growth in the number of wireless networks. Today, wireless networks (WLANs) can increasingly be found in office, education, and industrial environments.

In mobile computer networking there are two distinct approaches to enable wireless data communication. The first approach is that mobile units (nodes) communicate through a cellular network infrastructure. The major problems in this approach include the problem of “handoff”. When a mobile node travels out of range of one base-station and into the range of another a “handoff” occurs from the old base station to the new. This process must be performed smoothly without noticeable delay or packet loss. Another problem is that networks based on the cellular infrastructure are limited to places where there exits such a cellular network infrastructure.

The second approach is to form an ad hoc network among all users wanting to communicate with each other. The main feature of a mobile ad hoc network is that it does not require any fixed infrastructure for communication. This means that the communication range is limited by the individual nodes transmission ranges and is typically smaller compared to the range of cellular systems. Because ad hoc networks do not rely on any pre-established infrastructure, they can be deployed in places with no infrastructure. Therefore ad hoc networks can be useful in disaster recovery situations and in places with non-existing or damaged communication infrastructure where rapid deployment of a communication network is needed. Ad hoc networks can also be useful on conferences where people participating in the conference can form a temporary network without engaging the services of any pre-existing network. In this thesis the main focus will lay on the mobile ad hoc network.

1.2 Problem Description

The concept of ad hoc networking in computer communications is that users wanting to communicate with each other form a temporary network, without any form of centralized administration. Each node participating in the network acts both as host and router and must therefore be willing to forward packets for other nodes. For this purpose, a routing protocol is needed. Mobility, potentially very large number of mobile nodes, heterogeneity (terminals can have very different capabilities) and limited resources (like bandwidth and power) make routing in ad hoc networks extremely challenging. There are already several routing protocols developed for mobile ad hoc network what deal with these issues.

The characteristics of ad hoc networks impose new demands on the routing protocol. The most important characteristic is the dynamic topology, which is a consequence of node mobility. Nodes can change position quite frequently, which means that we need a routing protocol that quickly adapts to topology changes. The nodes in an ad hoc network can consist of laptops and personal assistants and are often very limited in resources such as CPU capacity, storage capacity, battery power and bandwidth.

In a mobile ad hoc network nodes are often powered by batteries. The power level of a battery is finite and limits the lifetime of a node. Every message sent and every

computation performed drains the battery. This means that the routing protocol should try to minimize control traffic, such as periodic update messages. To improve the lifetime of the nodes and network even further, one should also try to keep the data traffic as low as possible. This optimization can be achieved by utilizing power awareness routing. This means that routing decisions make by the routing protocol are based on the power-status of the nodes. Nodes with low batteries will be less preferably for forwarding packets than nodes with full batteries thus increasing the life of the nodes. However, not every routing protocol is suitable for implementing power awareness routing and different approaches on power awareness routing can be followed.

1.3 Objectives

The main objectives of this thesis are to study ad hoc networking and investigate the possibilities for power awareness routing in a mobile ad hoc network. This study must lead to an advice for a routing protocol. To demonstrate power awareness routing a routing protocol must be adapted for power awareness routing and implemented in a multihop ad hoc network.

The objectives of the assignment are to:

1. Study the background of mobile ad hoc networks.
2. Study the functionality of the IEEE 802.11b standard. Main focal points are ad hoc networking, multihop routing, and power-management.
3. Study power awareness routing.
4. Study the different ad hoc routing protocols and compare these with the conventional routing protocols. Ultimately this must lead to an advice for a routing protocol in which power awareness routing can implemented.
5. Specify a possible approach for implementing power awareness routing in combination with system dependent routing.
6. Design and build a multihop ad hoc network test bed, hereby implementing power awareness into an existing implementation of the OSPF (Open Shortest Path) routing protocol.
7. Specify and accomplish functionality experiments on the implemented power awareness routing prototype.

1.4 Outline of this thesis

This thesis is divided into eight chapters. Chapter 1 and 2 explore the concepts of wireless networking with the emphasis on the IEEE 802.11 standard and ad hoc networking. In chapter 3 the routing protocol will be discussed. Both ad hoc routing protocols and conventional routing protocols will be discussed in this chapter. Also a comparison of different protocols will be given in relation to their applicability in mobile ad hoc networks. In chapter 4 the aspects of power awareness routing in a mobile ad hoc network will be discussed. In chapter 5 the selected OSPF routing protocol for the implementation will be described. Chapter 6 describes the implementation of power awareness routing into an existing implementation of the OSPF routing protocol, and chapter 7 describes the experiments that are conducted on the implemented power awareness routing prototype. Finally, in chapter 8 some conclusions and recommendations will be given.

2 Wireless Networking

Wireless communication has become an increasingly popular topic in the computing industry. One major focal point within this area is the mobile wireless network. Currently we can distinguish two different variations of mobile wireless networks. The first type is known as the *infrastructured* network or the cellular wireless network. These cellular wireless networks require fixed and wired gateways. The gateways for these networks are known as base-stations or access points. A mobile unit (node) within these networks connects to, and communicates with, the nearest base-station that is within its communication radius. As the mobile travels out of range of one base-station and into the range of another, a "handoff" occurs from the old base station to the new, and the mobile is able to continue communication seamlessly throughout the network. Typical applications of this type of network include office wireless local area networks (WLANs).

The second type of a mobile wireless network is the *infrastructureless* mobile network. Because this type of network is often formed without pre-planning, and only as long as the network is needed, this type of network is commonly known as an ad hoc network. A mobile ad hoc network is a collection of wireless mobile nodes dynamically forming a temporary network. Ad hoc networks do not require any fixed infrastructure. However some schemes of ad hoc networks allow it to be connected with fixed networks to deliver extra services like Internet, E-mail and access to servers and printers. In ad hoc networks each node functions as a router and has to discover and maintain the routes to the other nodes in the network. Because ad hoc networks do not rely on any pre-established infrastructure, they can be deployed in areas without any infrastructure. Possible application areas of ad hoc networks are emergency search and rescue operations, military communications on the battlefield, meetings or conventions in which persons wish to quickly share information, and data acquisition operations in inhospitable terrain.

First in this chapter the most important wireless LAN standards will be discussed with their main differences and application areas. Because today IEEE 802.11 is one of the most deployed mobile network technology that support ad hoc networking, the IEEE 802.11 standard will be discussed in more detail. In the last section the most important aspects of mobile ad hoc networks will be discussed.

2.1 Wireless LAN standards

A wireless LAN (WLAN) can provide all the features and benefits of the traditional LAN technologies such as Ethernet and Token Ring without the limitations of wires or cables. WLANs use either infrared light (IR) or radio frequencies (RF) as transmission medium, instead of twisted-pair or fiber-optic cable in wired LANs. Of the two, RF is far more popular for its longer-range, higher bandwidth and wider coverage. Most wireless LANs today use the 2.4-gigahertz (GHz) frequency band, the only portion of the RF spectrum reserved around the world for unlicensed devices. Wireless networking can be applied both within buildings and between buildings. In general, wireless LANs are usually implemented as the final link between the existing wired network and a group of client computers, giving these users wireless access to the full resources and services of the wired network across a building or campus setting. Currently wireless LANs are primarily implemented in vertical applications such as manufacturing facilities, warehouses, and retail stores. Future applications of wireless LAN are expected in healthcare facilities, educational institutions, and corporate enterprise office spaces. These applications of WLANs require industry standardization to ensure product compatibility and reliability among the various manufacturers. In the next sections the most important of these industry standards will be briefly discussed.

2.1.1 Bluetooth

Bluetooth [1], a wireless standard developed by Ericsson, IBM, Intel, Nokia and Toshiba, is named after a Danish king who united Denmark and Norway in the 10th century. It is designed for short-range radio transmissions between devices no more than 10 meters apart. Bluetooth operates at a frequency of 2.4 GHz and transmits at speeds up to 1 Mbps (the next generation of Bluetooth will transmit at 2 Mbps.) Bluetooth is primarily for use in mobile devices to provide connectivity and synchronization; for example, two Bluetooth-enabled handheld devices a few meters apart can synchronize phone lists or schedules. The devices can connect on a one-to-one or one-to-many basis so that when near each other, Bluetooth devices create a "piconet": an ad hoc, peer-to-peer network of up to 8 nodes. For example, if all the participants at a meeting have Bluetooth-enabled laptops, they can create a piconet for sharing documents and messages. A Bluetooth-enabled printer in the room could be used by all without the need for cabling.

Bluetooth is already a widespread standard and is therefore widely supported among vendors. It supports both data and voice transmissions and does not require a line of sight. The main disadvantage is that Bluetooth's limited range makes it useful only for instances when the device is near another Bluetooth transmitter. Furthermore, the data transmission rates of Bluetooth are not nearly as fast as those of IEEE 802.11.

2.1.2 IEEE 802.11

IEEE 802.11 [2] is an extension of the Ethernet standard, adapted for wireless LANs. It consists of one MAC-layer standard and three physical-layer standards: two for radio transmissions (DSSS and FHSS) and one for infrared. 802.11 operates at 2.4 GHz and can transmit at speeds up to 2 Mbps at a range of 30–100 meters. IEEE 802.11b, ratified in 1999, boasts transmission rates of up to 11 Mbps, but only over the DSSS physical layer. Similar to its wired Ethernet counterpart, the 802.11 MAC layer uses a variation of CSMA/CD called carrier sense multiple access with collision avoidance (CSMA/CA). The latest extension of the IEEE 802.11 standard is IEEE 802.11g, which even increases the transmission rate to 54 Mbps.

The 802.11 standard define two modes: *infrastructure* mode and *ad hoc* mode. In infrastructure mode, the wireless network consists of at least one access point connected to the wired network infrastructure and a set of wireless end stations. This configuration is called a Basic Service Set (BSS). The second mode is the ad hoc mode (also called peer-to-peer mode or an Independent Basic Service Set, or IBSS). An ad hoc network is simply a set of 802.11 wireless stations that communicate directly with one another without using an access point or any connection to a wired network.

Like all IEEE 802 standards, the 802.11 standards focus on the bottom two levels of the ISO model, the physical layer and data link layer. Any LAN application, network operating system, or protocol, including TCP/IP and Novell NetWare, will run on an 802.11-compliant WLAN as easily as they run over Ethernet. The basic architecture, features, and services of 802.11b are defined by the original 802.11 standard. The 802.11b specification affects only the physical layer, adding higher data rates and more robust connectivity. With 802.11b WLANs, mobile users can get Ethernet levels of performance, throughput, and availability.

Because 802.11 is a true Ethernet specification, 802.11 devices can be integrated seamlessly into conventional Ethernet LANs. With a laptop and an 802.11 network adapter card, an employee can roam throughout a building, go from building to building, or even go to a remote office and always be connected to the network. The 802.11 standard is already commonly used in WLANs. The 802.11b standard provide

robust and reliable 11 Mbps performance. 802.11 standard products are widely available and are accepted and supported by most major networking and personal computer manufacturers and vendors.

Unlike Bluetooth, 802.11 does not support voice transmissions and additional overhead means 802.11 transmissions will always be slower than wired Ethernet. It is also feared that 802.11 transmissions can inadvertently disrupt critical Bluetooth transmissions, such as wireless medical or manufacturing monitoring devices. Another disadvantage of 802.11 is that it does not support, unlike HIPERLAN/1, multihop routing connections. Only direct peer-to-peer (Ad hoc mode) or direct 'wireless node' to 'access point' (infrastructure mode) communications are supported. In section 2.2 the IEEE 802.11 standard will be discussed in more detail.

2.1.3 HIPERLAN/1

High-Performance Radio LAN, Type 1 (HIPERLAN/1) [3] is a standard developed by the European Telecommunications Standards Institute (ETSI) to improve on the data throughput rates of 802.11. It is the first in a suite of HIPERLAN standards that operate in the 5 GHz range: HIPERLAN/2 is Wireless ATM, HIPERLAN/3 (renamed HIPERAccess) is for wireless local loop (the last segment between a home and the telephone system), and HIPERLAN/4 (renamed HIPERLink) is for wireless point-to-point connections.

The HIPERLAN/1 transmission scheme is the same as that for GSM, which means it uses TDMA as its air interface and Gaussian Minimum Shift Keying (GMSK) as its modulation scheme. HIPERLAN/1 can achieve data transfer rates up to 23.5 Mbps. With HIPERLAN the MAC layer is subdivided into the Channel Access Control (CAC) layer and the MAC layer. The CAC layer defines how a given channel access attempt will be made, depending on whether the channel is busy or idle and at what priority level the attempt will be made, if contention is necessary. Packets receive higher priority as they age.

Multihop routing support (not included in IEEE 802.11) is part of the HIPERLAN/1 specification. HIPERLAN-enabled devices choose a nearby "controller" and forward all outgoing traffic to that controller. The controller will then route the packet toward its destination. HIPERLAN-enabled devices also employ "hello" packets to announce their presence to other devices. In this sense, HIPERLAN-enabled devices behave similarly to conventional routers and are therefore able to structure their own network.

HIPERLAN/1 is compatible with wired and wireless Ethernet. It is very stable and flexible. It does support multihop routing, unlike the IEEE 802.11 standard. A disadvantage is that at the moment, unlike IEEE 802.11, HIPERLAN/1-compatible devices are not widely available and supported by most manufacturers and vendors.

2.2 The IEEE 802.11 standard

The Institute of Electrical and Electronics Engineers (IEEE) is recognized as the main LAN authority in the world. The IEEE 802 committee has established the main standards for the LAN industry for the past two decades, including 802.3 Ethernet, 802.5 Token Ring, and 802.3z 100BASE-T Fast Ethernet. In 1997, after seven years of work, the IEEE published 802.11, the first internationally sanctioned standard for wireless LANs. In September 1999 they ratified the 802.11b "High Rate" amendment to the standard, which added two higher speeds (5.5 and 11 Mbps) to 802.11.

Immediately after the Institute of Electrical and Electronics Engineers (IEEE) approved the 802.11b standard, they started work on the faster 802.11g standard. The final specification for 802.11g includes both mandatory and optional features. The official

802.11g standard includes a requirement for a data rate of only up to 24 Mbps. The additional data rates of 36, 48, and 54 Mbps are an optional component of the IEEE-approved standard. For competitive reasons, it's unlikely that any vendor would produce gear capable of only the minimum 24 Mbps.

A key difference between 802.11b and 802.11g wireless technologies is the modulation type. Complementary Code Keying (CCK) is used for 802.11b. Orthogonal Frequency Division Multiplexing (OFDM) is used for the higher data rates of 802.11g and CCK is used for the lower 802.11g data rates. Optional support for another modulation called Packet Binary Convolutional Code (PBCC) is also included in the 802.11g standard (22 Mbps to 33 Mbps). CCK modulation was included along with ODFM as a requirement in 802.11g to insure backward compatibility and co-existence with 802.11b. Although ODFM modulation is also used in 802.11a, 802.11g is not compatible with it because it operates in a different frequency band.

Table 1 illustrates the feature sets of each of the three versions of the 802.11 standard. 802.11g has to balance two separate modulation types to provide backwards compatibility with 802.11b.

	IEEE 802.11b	IEEE 802.11a	IEEE 802.11g
Max Mbps data rate	11	54	54
Modulation Type	CCK	ODFM	CCK and ODFM
Supported Data Rates	1, 2, 5.5, 11 Mbps	6, 9, 12, 18, 24, 36, 48, 54 Mbps	OFDM: 6, 9, 12, 18, 24, 36, 48, 54 Mbps CCK: 1, 2, 5.5, 11 Mbps
Frequencies	2.4–2.497 GHz	5.15–5.35 GHz 5.425–5.675 GHz 5.725–5.875 GHz	2.4–2.497 GHz

- Table 1: IEEE 802.11 technology comparison

The balancing of CCK and ODFM modulation requires a safety mechanism to control the traffic. The earlier 802.11b standard uses a Request to Send/Clear to Send (RTS/CTS) mechanism to determine if clear transmission is possible. The earlier 802.11b standard isn't ODFM-aware and therefore can only see other 802.11b transmissions. So the official 802.11g standard requires a protection mechanism for mixed b/g operation.

The 802.11 standard give WLANs Ethernet levels of performance, throughput, and availability. The standards-based technology makes it possible to combine 802.11 technology with other LAN technology. Like all IEEE 802 standards, the 802.11 standards focus on the bottom two levels of the ISO model, the physical layer and data link layer. Any LAN application, network operating system, or protocol, including TCP/IP and Novell NetWare, will run on an 802.11-compliant WLAN as easily as they run over Ethernet. The basic architecture, features, and services of 802.11b are defined by the original 802.11 standard. The 802.11b specification affects only the physical layer, adding higher data rates and more robust connectivity.

The 802.11 standard defines two pieces of equipment, a wireless *station*, which is usually a PC equipped with a wireless network interface card (NIC), and an *access point (AP)*, which can act as a bridge between a wireless and a wired network. An access point usually consists of a radio, a wired network interface (e.g., 802.3), and bridging software conforming to the 802.1d bridging standard. The access point can act as the base station for a wireless network, aggregating access for multiple wireless stations onto the wired network. Wireless stations can be a PC with an 802.11 PC Card, PCI, or ISA NICs, or embedded solutions in non-PC clients (such as an 802.11-based telephone handset or PDAs).

Wireless networks have fundamental characteristics that make them significantly different from traditional wired LANs. In wired LANs, an address is equivalent to a physical location. This is implicitly assumed in the design of wired LANs. In IEEE

802.11, the addressable unit is a station. The station is a message destination, but not (in general) a fixed location. Furthermore the physical layers used in IEEE 802.11 are fundamentally different from wired media. Thus the impact of the wireless medium on the design of IEEE 802.11 is that IEEE 802.11 physical layers:

- Use a medium that has neither absolute nor readily observable boundaries.
- Are unprotected from outside signals.
- Communicate over a medium significantly less reliable than wired cables.
- Have dynamic topologies.
- Lack full connectivity, and therefore the assumption normally made that every station can hear every other station is invalid (i.e., stations may be hidden form each other).
- Have time-varying and asymmetric properties.
- Because of limitations on wireless physical ranges, wireless LANs intended to cover reasonable geographic distances may be built from basic coverage building blocks.

2.2.1 IEEE 802.11 architecture

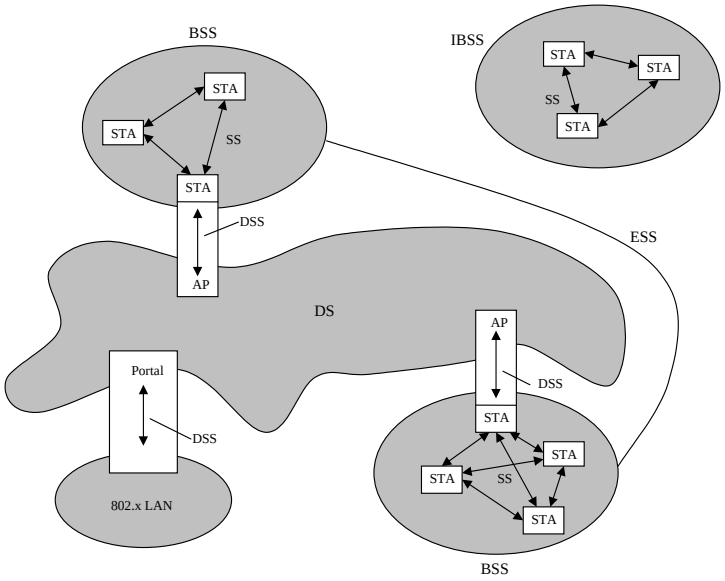
The IEEE 802.11 architecture consists of several components that can be used to build a wireless LAN.

The *Basic Service Set* (BSS) is the main building block of an IEEE 802.11 LAN. A BSS consists of a set of IEEE 802.11 stations controlled by a single Coordination Function (CF), which is generally an access point. It is useful to think of the BSS as the coverage area within which the member stations of the BSS may remain in communication. If a station moves out of its BSS, it can no longer directly communicate with other members of the BSS.

Physical limitations determine the direct station-to-station distance that may be supported. For some networks this distance is sufficient; for other networks, increased coverage is required. Instead of operating independently, a BSS may also form a component of an extended form of network that is build with multiple BSSs. The architectural component used to interconnect BSSs is the Distribution System (DS). IEEE 802.11 logically separates the wireless medium from the distribution system medium. As the IEEE 802.11 architecture is specified independently of any specific media type, the wireless medium and the distribution system medium may or may not be the same. The DS and BSSs allow IEEE 802.11 to create wireless networks of arbitrary size and complexity. This type of network is referred to as the *Extended Service Set* (ESS) network. The ESS consists of multiple BSS interconnected by the DS. The ESS appears as a single BSS to the IEEE 802.2 LLC layer. Stations connected to the distribution system are called the access points (AP), which provide the Distribution System Services (DSS) in order to enable to transport data between stations that cannot communicate over a single instance of the wireless medium. Note that all APs are also stations, thus they are addressable entities. The addresses used by an AP for communication on the wireless medium and on the distribution system medium are not necessarily the same. A Portal is the logical point where a non IEEE 802.11 LAN is connected to the DS. This allows the communication across different types of LANs. The operating configurations mentioned above are commonly known as the *infrastructure* mode. Since most corporate WLANs require access to the wired LAN for services (file servers, printers, Internet links) they will operate in infrastructure mode.

The *Independent Basic Service Set* (IBSS) is the most basic type of IEEE 802.11 LAN. Because an IBSS LAN is often formed without pre-planning, for only as long as the LAN is needed, this type of operation is often referred to as an ad hoc network. A minimum configuration of an ad hoc network may consist of only two stations. A

station operating in ad hoc mode (also called peer-to-peer mode) is simply a set of IEEE 802.11 wireless stations that communicate directly with one another without using an access point or any connection to a wired network. This mode is useful for quickly and easily setting up a wireless network in an environment where a wireless infrastructure does not exist or is not required for services. Examples are a hotel room, convention center, or airport, or where access to the wired network is barred (such as for consultants at a client site).



STA = Station
AP = Access Point
SS = Station Service
DSS = Distribution System Service
IBSS = Independent Basic Service Set
BSS = Basic Service Set
ESS = Extended Service Set
DS = Distribution System

- Figure 1: Components of the IEEE 802.11 architecture

Figure 1 comprises the components of the IEEE 802.11 architecture. All of the following situations are possible:

a) The BSSs may partially overlap. This is commonly used to arrange contiguous coverage within a physical volume.
b) The BSSs could be physically disjointed. Logically there is no limit to the distance between BSSs.
c) The BSSs may be physically collocated. This may be done to provide redundancy.
d) One (or more) IBSS or ESS network may be physically present in the same space as one (or more) ESS network. This may arise for a number of reasons. Two of the most common are when an ad hoc network is operating in a location that also has an ESS network, and when physically overlapping IEEE 802.11 networks have been set up by different organizations.

2.2.2 Services of IEEE 802.11 Networks

The services of IEEE 802.11 are sorted into two categories: the Station Services (SS) and the Distribution System Services (DSS). There are nine services specified by IEEE 802.11 (see Table 2). Six of the services are used to support MSDU (MAC Service Data Unit) delivery between stations. Three of the services are used to control IEEE 802.11 LAN access and confidentiality.

Station Services	Distribution System Services
Authentication	Association
Deauthentication	Disassociation
Privacy	Distribution
MSDU delivery	Integration
	Reassociation

- Table 2: Services of IEEE 802.11 networks

In an IBSS network or ad hoc network only the Station Services are available. Besides the *MSDU delivery*, which will be explained in the next section, the station service provide access and confidentiality control. The distribution system services are used to distribute messages in the distribution system (DS) and to support mobility.

The distribution services delivers MSDUs within the DS, but it is not specified in IEEE 802.11. The standard provides the DS with sufficient information in order to be able to determine the target AP of an MSDU. This is done by the three association-related services (*Association*, *Reassociation* and *Disassociation*). The *Integration* service enables the delivery of MSDUs between non 802.11 LAN and DS via a Portal. The *Integration* service depends on the *Distribution* service, and performs media and address space translation if necessary. The *Integration* service is outside the scope of IEEE 802.11. Mobility is supported by association services. IEEE 802.11 distinguishes three types of mobility:

- No-transition: Stations are stationary or move within a BSS.
- BSS-transition: Stations move from one BSS to another BSS within the same ESS.
- ESS-transition: Stations move from one BSS in one ESS to another BSS in a different ESS. This kind of mobility is not supported by 802.11; running services will be disrupted.

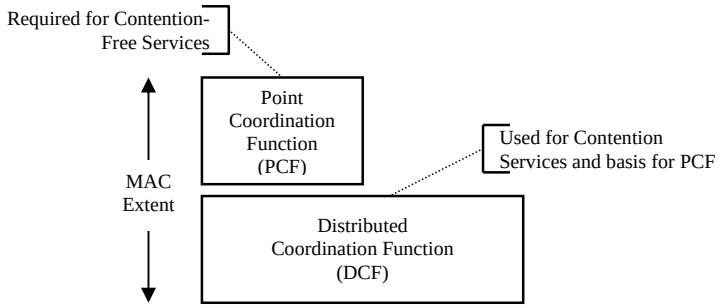
Each of the services is supported by one or more MAC frame types. Some of the services are supported by MAC management messages and some by MAC data messages. All of the messages gain access to the wireless medium via the IEEE 802.11 MAC sublayer medium access method described in the next section. Important to note is that IEEE 802.11 does not support multihop routing connections. No services are implemented for forwarding traffic to stations outside the physical range of the transmitting station. Only direct peer-to-peer or direct 'wireless node' to 'access point' (in infrastructure mode) communications are supported. Therefore possible multihop functionality should be implemented on a higher level (layer 3 or higher of the OSI model) than the two layers defined by the IEEE 802.11 standard.

2.2.3 The IEEE 802.11 Data Link Layer

The data link layer within IEEE 802.11 consists of two sublayers: the Logical Link Control (LLC) and the Media Access Control (MAC) layer. 802.11 uses the same 802.2 LLC and 48-bit addressing as other 802 LANs, allowing for very simple bridging from wireless to IEEE wired networks, but the MAC is unique to WLANs. The 802.11 MAC is very similar in concept to 802.3, in that it is designed to support multiple users on a shared medium by having the sender sense the medium before accessing it.

The IEEE 802.11 MAC protocol provides two type of service: asynchronous and contention-free. The asynchronous type of service is provided by the Distributed Coordination Function (DCF). The DCF provides the fundamental access method of the IEEE 802.11 MAC. This method is known as carrier sense multiple access with collision avoidance (CSMA/CA) protocol. The DCF is implemented in all the stations of the network and is used within both IBBS and infrastructure network configurations. The contention-free type of service is provided by the Point Coordination Function (PCF). The PCF provides time-bounded services and basically implements a polling

access method. Unlike the DCF, the implementation of PCF is not mandatory. Furthermore, the PCF itself relies on the asynchronous service provided by the DCF as shown in Figure 2.



- Figure 2: MAC architecture

The IEEE 802.11 MAC sublayer uses three types of messages:

- Data messages: which are used for data transmission;
- Control messages: which are used to control access to the medium (e.g. RTS, CTS and ACK);
- Management messages: which are frames that are transmitted the same way as data frames to exchange management information, but are not forwarded to upper layers.

Distributed Coordination Function (DCF)

The fundamental access method of the IEEE 802.11 MAC is a DCF known as *Carrier Sense Multiple Access with Collision Avoidance* (CSMA/CA). According to the DCF, a station must sense the medium before initiating the transmission of a packet. If the medium is sensed as being idle for a certain time interval then the station transmits the packet. Otherwise, the transmission is deferred and a backoff process is started. Specifically, the station computes a random number uniformly distributed between zero and a maximum called *Contention Window* (CW). The random number is multiplied by the slot time, resulting in the backoff interval used to set the backoff timer. This timer is decremented only when the medium is idle, whereas it is frozen when another station is transmitting. Each time the medium becomes idle, the station decrements the backup timer. As soon as the backup timer expires, the station is authorized to access the medium. If two or more stations start transmission simultaneously, a collision occurs.

Near/far problem

For 802.3 Ethernet LANs, the Carrier Sense Multiple Access with Collision Detection (CSMA/CD) protocol regulates how Ethernet stations establish access to the wire and how they detect and handle collisions that occur when two or more devices try to simultaneously communicate over the LAN. In an 802.11 WLAN, collision detection is not possible due to what is known as the “near/far” problem. To detect a collision, a station must be able to transmit and listen at the same time, but in radio systems the transmission drowns out the ability of the station to “hear” a collision. To account for this difference, 802.11 uses a slightly modified protocol known as Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA). CSMA/CA attempts to avoid collisions by using explicit packet acknowledgment (ACK), which means an ACK packet is sent by the receiving station to confirm that the data packet arrived intact. If the ACK is not received in a specified time interval, the station assumes that the transmitted packet was not successfully received, and retransmits the packet and

enters the backoff process again. However, to reduce the probability of collisions, after each unsuccessful transmission attempt the Contention Window is doubled until a predefined maximum (CW_{max}) is reached. After a successful transmission, the Contention Window is reset to CW_{min}. The DCF is implemented in all stations, for use with both IBSS and infrastructure network configurations.

Hidden-station problem

The hidden-station problem is the phenomenon that arises when a station is able to successfully receive frames from two different stations but that these two stations cannot receive signals from each other. In this case a station may sense the medium as being idle even the other is transmitting, this will result in a collision at the receiving station. To deal with the hidden-station problem, the IEEE 802.11 MAC protocol includes a mechanism based on the exchange of two short control frames. A Request-to-Send (RTS) frame is send by a potential transmitter to the receiver and a Clear-to-Send (CTS) frame that is send by the receiver in response to the received RTS frame. If the CTS frame is not received within a predefined time interval, the RTS frame is retransmitted by executing a backoff algorithm. After a successful exchange of the RTS and the CTS frames, the data frames can be sent by the transmitter.

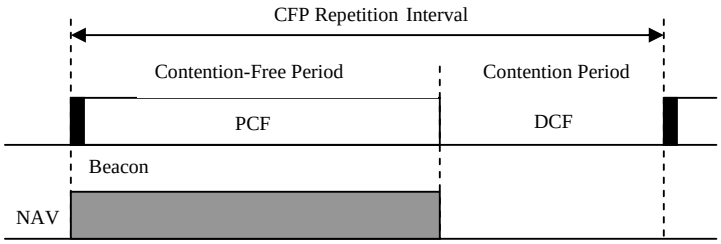
The RTS and CTS frames include a duration field that specifies the time interval necessary to completely transmit the data frame and the related acknowledgement. This information is used by stations that can hear either the transmitter or the receiver to update their network allocation vector (NAV). The network allocation vector (NAV) is an indicator, maintained by each station, of time periods when the station will not initiate transmission onto the wireless medium whether or not the station's senses that the wireless medium is busy. Since stations that can hear either the transmitter or the receiver refrain from transmitting until their NAV has expired, the probability of a collision occurring because of hidden-station is reduced. Of course, the drawback of using the RTS/CTS mechanism is an increased overhead, which may be significant for short data frames.

Furthermore, the RTS/CTS mechanism can be regarded as a way to improve the MAC protocol performance. When the mechanism is enabled, collisions can obviously occur only during the transmission of the RTS frame. Since the RTS frame is usually much shorter than the data frame, the waste of bandwidth and time due to the collision is reduced. The RTS/CTS is useful also while operating overlapping BSS or IBBS.

Point Coordination Function (PCF)

In order to support time-bounded services, the IEEE 802.11 standard defines the Point Coordination Function (PCF) to permit a Point Coordinator (PC) to have priority access to the medium. Unlike DCF, PCF is an optional access method and can only be used in infrastructure network configurations. Usually in an infrastructure based network an AP acts as a PC. Although PCF is optional, all stations are able to obey the medium access rules of the PCF, because it is based on the DCF. When the DCF and PCF are both used in the same BBS, the two access methods alternate, with a contention-free period (CFP) followed by a contention period (CP) (see Figure 3).

The Point Coordinator determines which station currently has the right to transmit. The operation is essentially that of polling, with the PC performing the role of the polling master. The CFP is periodically repeated in time and starts with the transmission of a beacon frame. This beacon contains the maximum duration of the CFP, and all stations in the BSS (other than the PC) set their NAVs to the maximum duration of the CFP.



- Figure 3: Relationship between CFP and CP

The PC gains control of the medium at the beginning of the CFP and attempts to maintain control for the entire CFP by waiting a shorter time between transmissions than the station using the DCF access procedure. The PC maintains a polling list, which consists of the *Association Identifier* (AID) of the stations requesting polling. A *CF-Pollable* station may request to be added to the polling list during *Association* or *Reassociation*. A *CF-Poll* is used by the PC to poll a station for the transmission of a data frame. *CF-Ack* is the acknowledgement to a successfully received frame under the PCF, either by a station or the PC. The *Null Function* is used to indicate that no data has to be transmitted. If all stations on the polling list have been polled and no data has to be transmitted by the PC during one CFP, the PC may prematurely stop the current CFP by sending a *CF-end*. On receiving a *CF-end*, all stations reset their NAV.

Synchronization

All stations within a single BBS will be synchronized to a common clock while maintaining a local timer. This is done by performing a Timing Synchronization Function (TSF). This TSF is different for infrastructure BBS and IBBS. The synchronization is maintained by broadcasting a so-called beacon. Beacons are transmitted periodically at the Target Beacon Transmission Times (TBTT). In case the medium is busy at TBTT, the transmission of the beacon is delayed.

In an infrastructure network the access point is the timing master and is responsible for the transmission of the synchronization beacon. At each TBTT, the AP schedules a beacon as the next frame to be transmitted according to the PCF procedures described earlier in the previous section. A receiving station shall always accept the timing information in beacons sent from the AP servicing its BSS. If a station's timer is different from the timestamp in the received beacon, the receiving station shall set its local timer to the received timestamp value.

In an IBBS network the beacon generation is distributed among all stations. All members of the IBBS participate in beacon generation. At each TBTT every station shall:

1. Suspend the decrementing of the backoff timer for any pending non-beacon transmission;
2. Calculate a random delay uniformly distributed in the range between zero and $2 \cdot CW_{min} \cdot$ slottime;
3. Wait for a period of random delay similar to the backoff algorithm;
4. If a beacon arrives in between the random delay timer and the beacon transmission are cancelled;
5. If the random delay timer expires, send a beacon.

CRC checksum and packet fragmentation

The IEEE 802.11 MAC layer also provides for two other robustness features: CRC checksum and packet fragmentation. Each packet has a CRC checksum calculated and attached to ensure that the data was not corrupted in transit. This is different from

Ethernet, where higher-level protocols such as TCP handle error checking. Packet fragmentation allows large packets to be broken into smaller units when sent over the air, which is useful in very congested environments or when interference is a factor, since larger packets have a better chance of being corrupted. This technique reduces the need for retransmission in many cases and thus improves overall wireless network performance. The MAC layer is responsible for reassembling fragments received, rendering the process transparent to higher-level protocols.

Power Management

IEEE 802.11 compliant devices will most probably be battery-powered. Therefore the 802.11 standard supports a power conservation mode to extend the battery life of these portable devices. A station may be in one of two power states:

- *Awake*; A station is fully powered.
- *Doze*; A station is able neither to receive nor transmit, and consumes very low power.

The standard supports two power-utilization modes, which determine how stations transit from one power state to the other:

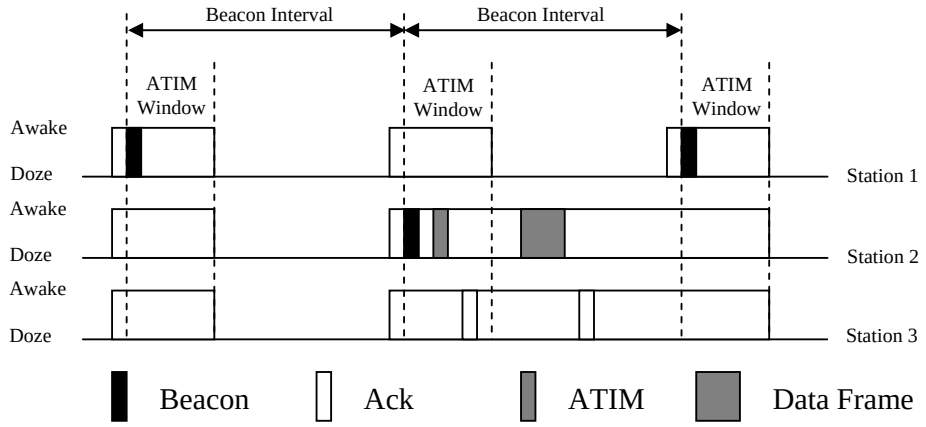
- *Active Mode* (AM); A station will be in the *Awake* state with the radio always on and drawing power. A station on the polling list of a PC will be in *Active Mode* for the duration of the CFP.
- *Power Save* (PS); A station will be in the *Doze* state with the access point queuing any data for it. The client radio will wake up periodically in time to receive regular *beacon* signals from the access point. The beacon includes information regarding which stations have traffic waiting for them, and the client can thus awake upon beacon notification and receive its data, returning to sleep afterward.

Power Management in an Infrastructure BSS

Stations changing power management modes have to inform the AP via successful frame exchange. Stations having buffered cells within the AP are announced in a *Traffic Indication Map* (TIM), which is included in all beacons generated by the AP. A station receiving and interpreting the TIM may poll the AP for the buffered frames during the *Contention Period*; therefore it is not necessary that the stations have to listen to each single beacon.

Power Management in an IBSS

As the beacons are generated by different stations within the IBSS, the TIM is not sufficient to announce traffic to stations in PS mode. Instead, a station announcing unicast traffic to another station uses the *Ad Hoc Traffic Indication Message* (ATIM). The receiving station acknowledges the reception of the ATIM, and remains in the Awake state for the whole beacon interval, waiting for the announced MSDUs to be delivered. All stations in PS mode have to wake up for a so-called *ATIM Window*, which follows the TBTT. During the ATIM Window, only beacons and ATIMs will be transmitted. After the ATIM, every station in PS mode knows whether it has to remain in the Awake state or is allowed to go to the Doze state. Figure 4 shows the basic operation of power management in an IBSS.



- Figure 4: Power management in an IBBS

2.2.4 The IEEE 802.11 Physical Layer

Originally there were three physical layers defined in IEEE 802.11. These included two spread-spectrum radio techniques and a diffuse infrared specification. The radio-based standards operate within the 2.4 GHz ISM band. These frequency bands are recognized by international regulatory agencies, such as the FCC (USA), ETSI (Europe), and the MKK (Japan) for unlicensed radio operations. The original IEEE 802.11 wireless standard defines data rates of 1 Mbps and 2 Mbps via radio waves using frequency hopping spread spectrum (FHSS) or direct sequence spread spectrum (DSSS). It is important to note that FHSS and DSSS are fundamentally different signaling mechanisms and will not interoperate with one another.

Using the frequency hopping technique, the 2.4 GHz band is divided into 75 one-MHz subchannels. The sender and receiver agree on a hopping pattern, and data is sent over a sequence of the subchannels. Each conversation within the 802.11 network occurs over a different hopping pattern, and the patterns are designed to minimize the chance of two senders using the same subchannel simultaneously. FHSS techniques allow for a relatively simple radio design, but are limited to speeds of no higher than 2 Mbps. This limitation is driven primarily by FCC regulations that restrict subchannel bandwidth to 1 MHz. These regulations force FHSS systems to spread their usage across the entire 2.4 GHz band, meaning they must hop often, which leads to a high amount of hopping overhead.

In contrast to FHSS, the direct sequence signaling technique divides the 2.4 GHz band into 14 twenty-two MHz channels. Adjacent channels overlap one another partially, with 3 of the 14 being completely nonoverlapping. Data is sent across one of these 22 MHz channels without hopping to other channels. To compensate for noise on a given channel, a technique called “chipping” is used. Each bit of user data is converted into a series of redundant bit patterns called “chips.” The inherent redundancy of each chip combined with spreading the signal across the 22 MHz channel provides for a form of error checking and correction; even if part of the signal is damaged, it can still be recovered in many cases, minimizing the need for retransmissions.

The main addition of 802.11b to the wireless LAN standard was to standardize the physical layer support of two new speeds, 5.5 Mbps and 11 Mbps. To accomplish this, DSSS had to be selected as the sole physical layer technique for the standard since, as noted above, frequency hopping cannot support the higher speeds without violating current FCC regulations. The implication is that 802.11b systems will interoperate with 1 Mbps and 2 Mbps 802.11 DSSS systems, but will not work with 1 Mbps and 2 Mbps 802.11 FHSS systems.

To support very noisy environments as well as extended range, 802.11b WLANs use *dynamic rate shifting*, allowing data rates to be automatically adjusted to compensate for the changing nature of the radio channel. Ideally, users connect at the full 11 Mbps rate. However when devices move beyond the optimal range for 11 Mbps operation, or if substantial interference is present, 802.11b devices will transmit at lower speeds, falling back to 5.5, 2, and eventually 1 Mbps.

2.3 Ad hoc networks

2.3.1 Main characteristics of ad hoc networks

In the field of wireless networks and mobile computing ad hoc networking has become one of the most focused research areas. A mobile ad hoc network consists of a number of nodes that form an autonomous network. These network nodes can move randomly and this results in unpredictable and frequently changes in the network topology. The nodes in mobile ad hoc networks communicate with one another via packet radios. Because of the limited radio propagation range of each node, multiple hops may be needed to reach others nodes. This characteristic makes the routing a difficult subject in ad hoc communications. The main characteristics of ad hoc networks are:

- The lack of an existing network infrastructure or centralized administration.
- All the communication is carried over the wireless medium.
- A dynamic topology

2.3.2 The problem areas in ad hoc networking

Due to the characteristics, described in the previous section, some specific problems have to be solved in ad hoc networking. Some key problem areas in ad hoc networking are:

- The network topology:
  Nodes are free to move arbitrarily; thus, the network topology (which is typically multihop) may change randomly and rapidly at unpredictable times, and may consist of both bi-directional and unidirectional links. Furthermore, radio communications are extremely vulnerable to propagation faults, this means that the connectivity between network nodes can not be guaranteed; The problem of vulnerability to propagation faults can result in a link or network node failure. The routing protocol should resolve these problems by switching to an alternative route and bypassing the problem area.

- The bandwidth:
  Wireless links will have significantly lower capacity than their hardwired counterparts. In addition, the realized throughput of wireless communications (after accounting for the effects of multiple access, fading, noise, and interference conditions, etc.) is often much less than a radio's maximum transmission rate. As a consequence of the limited bandwidth the communication needed for control and management functions in the network must be kept at a minimum, however this must not go at the expense of the effectiveness of the data communication. The time needed to setup the data communication must therefore be kept minimal to guarantee a certain quality of service (QoS). In a routing protocol these properties must be well balanced, depending on the specifications of the desired network. Another effect of the relatively low to moderate link capacities is that congestion is typically the norm rather than the exception, i.e. aggregate application demand will likely approach or exceed network capacity frequently. As the mobile network is often simply an extension of the fixed network

infrastructure, mobile ad hoc users will demand similar services. These demands will continue to increase as multimedia computing and collaborative networking applications rise.

- The energy consumption:
  As some of the mobile devices are expected to be handheld with limited battery power, the energy consumption must therefore be kept as low as possible. One solution for power conservation (as implemented in IEEE 802.11) could be to set unused stations in a sleeping mode, so that they stop transmitting and/or receiving (even receiving requires power) for arbitrary time periods. Another possible solution for solving the problem of the energy consumption could be to minimize the required transmission power, however the limiting of transmission power will result in further limiting the radio propagation range of each node. Consequently, this will result into more hops during a connection and therefore making the routing in ad hoc networks more complexly.

- The security:
  Mobile wireless networks are generally more vulnerable to physical security threats than fixed-cable networks. The increased possibility of eavesdropping, spoofing, and denial-of-service attacks should be carefully considered. Existing link security techniques are often applied within wireless networks to reduce security threats. As a benefit, the decentralized nature of network control in ah hoc networks provides additional robustness against the single points of failure of more centralized approaches.

- The scalability:
  Some envisioned networks (e.g. mobile military networks or highway networks) may be relatively large (e.g. tens or hundreds of nodes per routing area). The need for scalability is not unique to ad hoc networks. However, in light of the characteristics of ad hoc networks, some new mechanisms are required to achieve scalability.

These issues create a set of underlying assumptions and performance concerns for the design of a routing protocol. At the moment much research is done in these problem areas and there are already a variety of routing protocols developed to cope with these issues.

2.3.3 Routing performance issues

To operate efficiently in a mobile networking context, a protocol should be designed and deployed with an expected networking context firmly in mind. To judge the performance of a routing protocol one should exam a number of issues. These issues should be independent of any given routing protocol. The following is a list of desirable properties:

- Distributed operation: This is an essential property, but it should be stated nonetheless.
- Loop-freedom: Not required per se in light of certain performance criteria, but generally desirable to avoid worst-case phenomena, e.g. a small fraction of packets spinning around in the network for arbitrary time periods. Ad hoc solutions can bound the problem, but more structured and well formed approaches are generally desirable and oftentimes lead to better overall performance.
- Demand-based operation: Instead of assuming uniform traffic distribution within the network (and maintaining routing between all nodes at all times), let the routing algorithm adapt to the varying traffic pattern on a demand or as needed basis. If this is done intelligently, it will utilize network resources more efficiently.

- "Sleep" period operation: As a result of power conservation, or some other need to be inactive, some nodes may stop transmitting and/or receiving (even receiving requires power) for arbitrary time periods. A routing protocol should be able to accommodate such sleep periods without overly adverse consequences.

The following is a list of some quantitative metrics that are appropriate for assessing the performance of any routing protocol:

- End-to-end data throughput and delay: Statistical measures of data routing performance (e.g., means, variances, distributions) are important. These are the measures of routing protocol effectiveness as measured from the external perspective of other protocols that make use of routing.
- Efficiency: If data routing effectiveness is the external measure of a protocol's performance, efficiency is the internal measure of its effectiveness. To achieve a given level of data routing performance, two different protocols may expend differing amounts of overhead, depending on their internal efficiency. Protocol efficiency may or may not directly affect data routing performance. If control and data traffic must share the same channel, and the channel's capacity is limited, then excessive control traffic more severely impacts data throughput performance.

It is unlikely that one routing protocol or mode for mobile ad hoc networking is the best approach for all networking issues. Parameters that define a networking context and that should be considered during protocol design, simulation and comparison include:

- Network size: Measured as the number of nodes.
- Network connectivity: The average degree of a node (i.e. the average number of neighbors of a node).
- Topological rate of change: The rate with which a network's topology is changing.
- Link capacity: Effective link speed, measured in bits/second, after accounting for losses due to multiple access, coding, framing, etc.
- Fraction of unidirectional links: How effectively does a protocol perform as a function of the presence of unidirectional links?
- Fraction and frequency of sleeping nodes: How does a protocol perform in the presence of sleeping and awakening nodes?
- Traffic patterns: Different types of traffic distribution experienced within a network (e.g., (1) uniform: all nodes are equally likely receivers and sources providing equivalent network load, (2) non-uniform: certain routing nodes are sourcing and/or receiving more network traffic than others).
- Mobility: When, and under what circumstances, are temporal and spatial topological correlation relevant to the performance of a routing protocol?

The preceding lists are not exhaustive, and merely give an indication of the number of dimensions that should be considered in the evaluation of a routing protocol. These protocol evaluation issues highlight performance metrics that can help to compare and judge protocol performance.

2.3.4 Classification of Ad Hoc Routing Protocols

The issues described in section 2.3.2 and 2.3.3 constrain the utilization of conventional routing protocols in wireless ad hoc networks. Ad hoc routing protocols for wireless networks have to adapt quickly to the frequent and unpredictable changes of routing topology and must minimize the generated overall network overhead. Today a large number of different routing protocols for ad hoc networking are developed to cope with these issues, each with their own features and characteristics.

Based on a literature study, a classification of the wireless ad hoc routing protocols is made, according to their design aspects. Now these classifications of wireless ad hoc protocols will be described, appendix A gives a complete list with all the mentioned routing protocols with their abbreviations and references. In section 3.2 and 4.2 a selection of ad hoc routing protocols will be discussed in more detail.

1. Routing philosophy

The main classification of ad hoc routing protocols is according to their routing philosophy. The routing philosophy is the procedure that the routing protocol uses to establish and maintain the communication between the nodes in the network. The following categories in routing philosophy can be defined:

(1) Table-driven or Proactive routing protocols: The routing protocol attempts to maintain consistent, up-to-date routing information from each node to every other node in the network. This can be achieved in different ways, and thus divides the protocols into two subclasses: event driven and regular updated protocols. Examples are CGSR, DBF, DSDV, FSR, OLSR, STAR, TBRPF, TORA or WRP.

(2) Source-initiated (On-Demand) or Reactive routing protocols: A network using an on-demand protocol will not maintain correct routing information on all nodes for all times. Instead, such routing information is obtained on demand. Examples are ABR, AODV, CEDAR, DREAM or DSR.

(3) Hybrid routing protocols: protocols that utilize both proactive and on-demand routing. An example is ZRP.

2. Routing architecture

The routing architecture is the way of how the ad hoc network is configured. Two categories can be distinguished in the routing architecture of ad hoc networks.

(1) Hierarchical or clustered: Here are the network nodes partitioned into groups called clusters. Within each cluster, one node is chosen to perform the function of a cluster head. Routing traffic between two nodes that are in two different clusters travel always through the cluster heads of the source and destination clusters. Depending on the number of hierarchies, the depth of the network can vary. Examples are CBRP and CGSR. Another form of hierarchical routing is that some protocols (FSR, DREAM) introduce a set of scopes for routing information. In any of these protocols, close, fast moving nodes receive more information more frequently than others.

(2) Flat or non-hierarchical: All other protocols, no structured architecture is defined in the ad hoc network, and all nodes are equal. Connections are established between nodes that are in close enough proximity to allow sufficient radio propagation conditions to establish connectivity. When a connection must be established with nodes outside the radio range, a routing protocol must be used to determine the optimum (multihop) route to the desired destination. Routing between any two nodes is constrained only by the connectivity conditions and, possibly, by security limitations.

3. Position based Protocols

Position based routing algorithms claim that no routing tables need to be maintained and thus no overhead due to route discovery and route maintenance is imposed. But they need to obtain position data of their corresponding destinations, either by an internal discovery process, or by an independent position service (i.e. GPS),

which will then impose overhead to maintain the position information (either proactively or on-demand). Examples are DREAM, GLS, GPSAL, LAR or ZHLS.

4. Uniform vs. Non-Uniform Protocols

A uniform protocol does not assign any special roles to any node. In a non-uniform protocol some nodes may be assigned a special role, which needs to be performed in a distributed fashion. Typically clustering protocols are non-uniform. Examples of non-uniform protocols are CBRP, CGSR, CEDAR, LANMAR and OLSR.

5. Route Selection Strategy

The route selection strategy is an important aspect of a routing protocol. The main representatives and the protocols, which use them, will now be described.

(1) Signal Strength: Route packets along the connection with the best signal strength. This is mainly used by ABR and SSR.

(2) Link Stability: Route packets along the connections that appear most stable over a period of time. It is for instance used by FORP.

(3) Shortest Path/Link State: Select a shortest path according to some metric. This is used by many protocols: CEDAR, DDR, FSR, GSR, LANMAR, OLSR, STAR, TBRPF.

(4) Distance Vector: The common distance vector method, usual by hop count, is used by AODV, DSDV, DSR, WRP, ZRP.

(5) Directional Routing: This routes into the geographic direction of the target and is mainly used by location based protocols: DREAM, LAR.

(6) Link Reversal Routing: is a routing family which is used by LMR and TORA. It is based on flows in a graph.

6. Full vs. Reduced Topology Information

Most Routing Protocols transmit topology information, but not all distribute the complete topology information they are aware of. It is difficult to classify the protocols according to this characteristic. Also even if full topology information is maintained in each node, the messages usually only carry sufficient information to reflect the changes in topology but never the whole topology information, since that would not scale. Full topology is maintained in: DDR, GSR, OLSR, STAR (in ORA mode), TBRPF (in full topology mode). Reduced Topology is maintained in: FSR, LANMAR, STAR (in LORA mode), TBRPF (in partial topology mode), WRP, ZRP.

7. Use of Source Routing

A few routing protocols utilize source routing. This means, forwarding depends on the source of the message. Commonly, the source puts all the routing information into the header of a packet. Forwarding nodes utilize this information. In some cases, the forwarding nodes may alter the routing information in the packet to be forwarded. They are just a few protocols using source routing: e.g. CBRP and DSR.

8. Use of broadcast messages

Broadcast can have different meanings in a wireless environment. There is a full netwide broadcast, which means, a message is intended for every node in the network, and needs to be retransmitted by intermediate nodes. On the other hand,

there is a local broadcast, which is intended for any node within the senders reach, but which is not retransmitted at all. In between there are limited broadcasts, in which the maximum hop count (time to live) is limited as desired. Finally, directional routing protocols do not use broadcasts by intention, but would use local multicasts (like a local broadcast, but not addressed to all neighbours).

9. Recovery Mechanisms

Since the routing information in each node may become stale, some protocols may need a route recovery or route conservation mechanism. It is clear, that proactive routing protocols do not need a specific recovery mechanism, since they react to topology changes anyway within a short period. On-Demand protocols however, need to fix routes which are not available any more. The following protocols have some (explicit or implicit) recovery mechanism: ABR, AODV, CBRP, DREAM, DSR, FORP and ZRP.

10. Alternate Path Routing

One approach in ad hoc network routing is the use of multiple routes. This kind of routing is called Alternate Path Routing (APR) or Multipath routing. Alternate path routing can be used to improve the quality of service (QoS) of networks. One example of a multipath routing protocol is the Split Multipath Routing protocol (SMR).

11. Power Awareness Routing

Another relative new field of study in ad hoc networking is power awareness routing. Because in a mobile ad hoc network nodes are often powered by batteries, the power level of a battery is finite and limits the lifetime of a node. To improve the lifetime of the nodes and network one should utilize power aware routing. This means that routing decisions are make on the basis of the power information of the nodes which must result in increasing the lifetime of the nodes. There are different approaches on power awareness routing, examples of power aware protocols are ISAIAH, PARO and PAMAS.

3 Routing Protocols

The Internet Engineering Task Force (IETF) Mobile Ad hoc Networks (MANET) working group is working on routing specifications for ad hoc networks. This working group proposed a number of routing protocols for mobile ad hoc networks. In this chapter several of these routing protocols will be discussed. To compare the ad hoc routing protocols with the conventional routing protocols, mainly used in the Internet, the two most common Internet routing protocols will also be discussed. In section 3.2 the most popular mobile ad hoc routing protocols will be described, hereby making a distinction in routing philosophy between the “table-driven” and the “source-initiated” routing protocols. Finally the different routing protocols will be compared in relation to their applicability in mobile ad hoc networks.

3.1 Conventional Routing Protocols

In network environments many different routing protocols are used. The Internet, for example, is divided into a collection of autonomous systems, each of which is normally administrated by a single entity. A Corporation or University may define an autonomous system. Every autonomous system uses a routing protocol to communicate between the routers in the autonomous system. These protocols are called Interior Gateway Protocols (IGP). The most popular IGP has been RIP. OSPF is a newer IGP that intends to replace RIP, at least in large networks. Another type of routing protocols are the Exterior gateway protocols, such as the Border Gateway Protocol (BGP), they perform the routing between different autonomous systems. The in this thesis described routing protocols are all Interior Gateway Protocols.

3.1.1 Routing Information Protocol (RIP)

Routing Information Protocol (RIP) [4] is a *distance vector routing protocol* based on the Bellman-Ford algorithm, meaning it bases its routing path on the distance (number of hops) to the destination. A router implements RIP by storing information in its routing table. A destination column indicates all possible destination networks, a next hop field identifies the router port to send the packet next, and the distance field refers to the number of hops it will take to reach the destination network. A RIP routing table only contains the best route to a particular destination. If the router receives new routing information from another node, it will overwrite the entry. RIP maintains optimum routing paths using three mechanisms:

- Each node sends out a regular routing update every 30 seconds.
- A triggered routing update message will be send whenever the metric for a route has been changed.
- Routing update messages can also be sent in response to a specific query.

For example, if a router finds that a particular link is faulty, it will update its routing table, then send a copy of the modified table to each of its neighbors. The neighbors will update their tables with the new information and send updates to their neighbors, and so on. Within a short period, all routers will have the new information. To adjust for rapid network-topology changes, RIP specifies a number of stability features that are common to many routing protocols. RIP, for example, implements the split-horizon and hold-down mechanisms to prevent incorrect routing information from being propagated. In addition, the RIP hop-count limit prevents routing loops from continuing indefinitely. RIP uses numerous timers to regulate its performance. These include a *routing-update timer*, *a route timeout*, and a *route-flush timer*. The routing-update timer clocks the interval between periodic routing updates. Generally, it is set to 30 seconds,

with a small random number of seconds added each time the timer is reset to prevent collisions. Each routing-table entry has a route-timeout timer associated with it. When the route-timeout timer expires, the route is marked invalid but is retained in the table until the route-flush timer expires.

3.1.2 Open Shortest Path First (OSPF)

Because RIP is not very robust, it lacks the ability of handling larger networks and the capability to effectively determine alternate paths, the *Open Shortest Path First* (OSPF) protocol [5] has been developed as an improvement to the RIP protocol. The basis of OSPF is the Shortest Path First (SPF) algorithm.

OSPF is a *link state routing protocol* (Some people also refer to OSPF as a distributed-database protocol). OSPF maintains a topological database that stores information related to the state of links within an autonomous network. To each link a dimension-less cost is assigned based on QoS quantities, like throughput, round trip time or reliability. The information stored in the database focuses on the topology of the networks with a directed graph. Routers and networks form the vertices of the graph. Periodically this information is broadcast (flooded) to all the routers in the autonomous system. An OSPF router computes the shortest path to all the other routers in the autonomous system regarding itself as the working node (the root). With OSPF, when a router is booted it announces its presence by sending a Hello message to each of its possible neighbors. Periodically, each neighbor sends a Link State Update message. This message shows the status of the router and the cost that is used in the topological database. Each Link state message has a sequence number, so the routers can distinguish the freshness of that message. Routers use these messages also when a link becomes operational or non-operational or when the cost of the link changes. This method is much faster then the distance-vector protocols, especially in case of changes in the links in the network.

OSPF is supporting speedy recovery from topology changes because OSPF routers can reroute data traffic as necessary. OSPF also minimizes overhead packet traffic when announcing changes by only sending information regarding only the change, instead of the entire routing table.

3.2 Routing protocols for ad hoc networks

Mobility, potentially very large number of mobile nodes, and limited resources (like bandwidth and power) make routing in ad hoc networks extremely challenging. The two main functions of the routing protocol are the selection of possible routes for the source node to the destination node and to forward the messages to their correct destination. Routing protocols for wireless ad hoc networks have to adapt quickly to the frequent and unpredictable changes of topology and must keep the communication and processing time minimal. At the moment a large number of different routing protocols for ad hoc networking are already developed, each with their own features and characteristics.

The existing wireless routing protocols can generally be classified into two main categories according to their routing philosophy:

- Table-driven (also named proactive protocols)
- Source-initiated (also named demand-driven or reactive protocols)

3.2.1 Table-driven Routing Protocols

The main characteristic of table-driven routing protocols is that the routing protocol attempts to maintain consistent, up-to-date routing information from each node to every other node in the network. Each node has to maintain one or more tables to store this routing information. The protocol has to respond to changes in network topology by propagating updates throughout the network in order to maintain a consistent network view. The areas in which the various table-driven routing protocols differ are the number of necessary routing-related tables and the methods by which changes in network structure are broadcasted. Some of the most popular table-driven ad hoc routing protocols will now be discussed.

Destination Sequenced Distance Vector Routing Protocol (DSDV)

The Destination Sequenced Distance Vector Routing Protocol (DSDV)[6] is a table-driven algorithm based on the classical Bellman-Ford routing mechanism. Improvements made to the Bellman-Ford algorithm include freedom from loop in routing tables. Each mobile node in the network maintains a routing table in which all of the possible destinations within the network and the number of hops to each destination are recorded. In the table each entry is marked with a sequence number assigned by the destination node. These sequence numbers enable the nodes to distinguish old routes from new ones, thereby avoiding the formation of routing loops. The routing tables are periodically updated by transmitting special packets through the network. To minimize the amount of network traffic during these updates, route updates can employ two possible types of packets. The first is known as a full dump. This type of packet includes all available routing information. During periods with only occasionally changes to the topology, these packets are transmitted infrequently. When smaller incremental packets are used to relay only that information which has changed since the last full dump.

The sequence number shows the freshness of a route and routes with a higher sequence number are more favorable. New route broadcasts contain the address of the destination, the number of hops to reach the destination, the sequence number of the information received regarding the destination, as well as a new sequence number unique to the broadcast. The route labeled with the most recent sequence number is always used. In the event that two updates have the same sequence number, the route with the lower number of hops is used in order to optimize (shorten) the path.

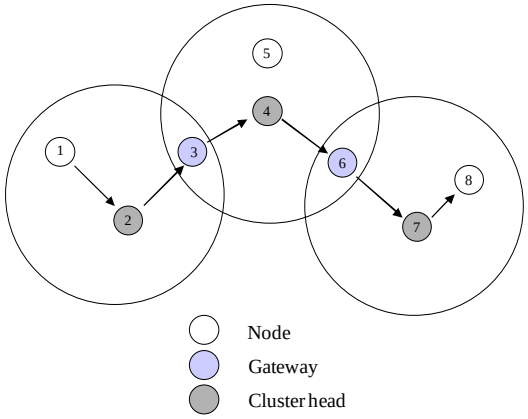
The nodes also keep track of the settling time of routes, or the weighted average time what routes to a destination fluctuate before the route with the best metric is received. By delaying the broadcast of routing updates by the length of the settling time, nodes can reduce network traffic and optimize routes by eliminating those broadcasts what would occur if a better route was discovered in the very near future.

Clusterhead Gateway Switch Routing Protocol (CGSR)

The Clusterhead Gateway Switch Routing Protocol (CGSR) [7] differs from the previous protocol in the type of addressing and network organization scheme employed. Instead of a "*flat*" network, CGSR is a clustered multihop mobile wireless network with several heuristic routing schemes. The main feature of CGSR is the use of cluster heads. Each cluster head controls a group of ad hoc nodes. To elect a node as the cluster head a distributed algorithm is used in the cluster. The disadvantage of having a cluster head scheme is that frequent cluster head changes can adversely affect routing protocol performance since nodes are busy in cluster head selection rather than packet relaying. Therefore, instead of invoking cluster head reselection every time the cluster membership changes, a Least Cluster Change (LCC) clustering algorithm is introduced. Using LCC, cluster heads only change when two cluster

heads come into contact and a cluster head becomes surplus, or when a node moves out of contact of all other cluster heads and a new cluster head is needed.

CGSR uses DSDV as the underlying routing scheme and has therefore much of the same overhead as DSDV. However, it modifies DSDV by using a hierarchical cluster-head-to-gateway routing approach to route traffic from source to destination. Gateway nodes are nodes that are within communication range of two or more cluster heads. A packet sent by a node is first routed to its cluster head, and then the packet is routed from the cluster head to a gateway to another cluster head, and so on until the cluster head of the destination node is reached. The packet is then transmitted to the destination. Figure 5 illustrates an example of this routing scheme.



- Figure 5: CGSR, routing from node 1 to node 8

Using this method, each node must keep a “cluster member table" where it stores the destination cluster head for each mobile node in the network. Each node will periodically broadcast this cluster member table using the DSDV algorithm. Nodes update their cluster member tables on reception of such a table from a neighbor. In addition to the cluster member table, each node must also maintain a routing table, which is used to determine the next hop in order to reach the destination. On receiving a packet, a node will consult its cluster member table and routing table to determine the nearest cluster head along the route to the destination. Next, the node will check its routing table to determine the next hop used to reach the selected cluster head. It then transmits the packet to this node.

Wireless Routing Protocol (WRP)

The Wireless Routing Protocol (WRP) [8] is a table-based protocol with the goal of maintaining routing information among all nodes in the network. WRP is also based on the distance vector algorithm. Each node in the network maintains the following four tables:

- The distance table;
- The routing table;
- The link-cost table;
- The message retransmission list (MRL) table.

Each entry of the MRL contains the sequence number of the update message, a retransmission counter, an acknowledgment-required flag vector with one entry per neighbor, and a list of updates sent in the update message. The MRL records which

updates in an update message need to be retransmitted and which neighbors should acknowledge the retransmission.

Nodes inform each other of link changes through the use of update messages. An update message is sent only between neighboring nodes and contains a list of updates (the destination, the distance to the destination, and the predecessor of the destination), as well as a list of responses indicating which nodes should acknowledge (ACK) the update. Nodes send update messages after processing updates from neighbors or detecting a change in a link to a neighbor. In the event of the loss of a link between two nodes, the nodes send update messages to their neighbors. The neighbors then modify their distance table entries and check for new possible paths through other nodes. Any new paths are relayed back to the original nodes so that they can update their tables accordingly.

Nodes learn of the existence of their neighbors from the receipt of acknowledgments and other messages. If a node is not sending messages, it must send a hello message within a specified time period to ensure connectivity. Otherwise, the lack of messages from the node indicates the failure of that link; this may cause a false alarm. When a mobile receives a hello message from a new node, that new node is added to the mobile's routing table, and the mobile sends the new node a copy of its routing table information. In WRP, routing nodes communicate the distance and second-to-last hop information for each destination in the wireless networks. WRP avoids the “count-to-infinity” problem by forcing each node to perform consistency checks of predecessor information reported by all its neighbors. This ultimately (although not instantaneously) eliminates looping situations and provides faster route convergence when a link failure event occurs.

Optimized Link State Routing protocol (OLSR)

The Optimized Link State Routing protocol [9], inherits the concept of forwarding and relaying from HIPERLAN (a MAC layer protocol) which is standardized by ETSI. The OLSR protocol operates as a table driven or proactive protocol and exchanges topology information with other nodes of the network at regular intervals.

The key concept used in the protocol is that of multipoint relays (MPRs). MPRs are selected nodes, which forward broadcast packets during the flooding process. This technique substantially reduces the packet overhead as compared to pure flooding mechanism where every node retransmits the packet when it receives the first copy of the packet. A second optimization is achieved by minimizing the contents of the control packets flooded in the network. In contrary to the classic link state algorithm, only a small subset of links with the neighbor nodes is declared instead of all the links, namely the links to those nodes which are its multipoint relay selectors. This information is then used by the OLSR protocol for route calculation. Consequently, the routes contain only the MPRs as intermediate nodes from a source to a destination. OLSR provides optimal routes (in terms of number of hops). The protocol is particularly suitable for large and dense networks as the technique of multipoint relays works well in this context.

The protocol does not depend on a central entity; all the routers in the network have their own routing tables and do not depend on any specific node. Periodically the protocol sends the information about its multipoint relay selectors, to help the other nodes to build routes to it. OLSR keeps the routes for all destinations in the network. The protocol provides shortest path routes based on the number of hops. The protocol may optimize the reactivity to topological changes by reducing the time interval for periodic control message transmission. Because the protocol uses a link state algorithm, the routing is loop-free when in a stable state.

The idea of multipoint relays is to minimize flooding of broadcast messages in the network by reducing duplicate retransmissions in the same region. Each node in the network selects a set of nodes in its neighborhood, which may retransmit its packets. This set of selected neighbor nodes is called the multipoint relay (MPR) set of that node. The neighbors of a node N, who are not in its MPR set, receive and process broadcast messages but do not retransmit broadcast messages received from node N.

Each node selects its MPR set among its one-hop neighbors. This set is selected such that it covers (in terms of radio range) all the nodes that are two hops away. Each node maintains information about a set of its neighbors. This is the set of neighbors, called the “Multipoint Relay Selectors”, which have selected the node as a MPR. A node obtains this information from the periodic Hello messages received from the neighbors. OLSR calculates the routes to a destination through the MPR nodes. The MPR nodes are selected as intermediate nodes in the path between a source and a destination. To implement this, each node in the network periodically broadcast the information describing which neighbors have selected it as a multipoint relay. Upon receipt of this "MPR Selectors" information, each node calculates or updates the route to each known destination. So principally, the route is a sequence of hops through the multipoint relays from source to the destination.

The two message types OLSR mainly uses are:

- Hello messages, performing the task of neighbor sensing.
- TC messages, performing the task of multipoint relay information declaration.

As mentioned earlier each node broadcasts Hello messages, containing information about neighbors and their link status. These control messages are broadcast to all one-hop neighbors, but are not relayed to further nodes. In order to build the topology information database needed for routing the packets, each relay node broadcasts specific service messages called Topology Control (TC) messages. TC messages are forwarded, like usual broadcast messages, to all nodes in the network and take advantage of multipoint relays. A TC message is sent by a node in the network to declare its MPR Selector set. The information diffused in the network by these TC messages will help each node to calculate its routing table. A node may transmit additional TC-messages to increase its reactiveness to link failures.

Each node maintains a routing table, which allows it to route the messages for the other destinations in the network. The routing table is based on the information contained in the neighbor table and the topology table. When a change is detected in these tables, the routing table is re-calculated to update the route information about each destination in the network. The update of this routing information does not generate or trigger any messages to be transmitted, neither in the network, nor in the one-hop neighborhood.

Topology Broadcast Based on Reverse-Path Forwarding (TBRPF)

TBRPF [10] is a table-driven, link state routing protocol designed for mobile ad hoc networks. It maintains optimal paths to all destinations at all times, unlike on-demand routing protocols. It does not require the periodic broadcast of topology information, unlike OLSR. Instead, only differential changes in topology are reported in order to minimize overhead. TBRPF has two modes: full topology (FT) and partial topology (PT). The protocol does not provide support for unidirectional links; instead it uses only bi-directional links (as in IEEE 802.11).

Unlike TBRPF-PT, the full topology (FT) mode of TBRPF provides each node with the state of every link in the network. TBRPF-FT uses the concept of reverse-path forwarding to broadcast each Link State Update in the reverse direction along the

spanning tree formed by the paths from all nodes to the source of the update. That is, each Link State Update is broadcast along the minimum-hop-path tree rooted at the source of the update, there being one tree per source. The broadcast trees are updated dynamically using the topology information that is received along the trees themselves, thus requiring very little additional overhead for maintaining the trees. Minimum-hop path trees are used because they change less frequently than shortest-path trees based on a metric such as delay. Based on the information received along the broadcast trees, each node computes its parent and children for the broadcast tree rooted at each source u. Each node forwards updates originating from source u to its children on the tree rooted at source u.

TBRPF-FT achieves reliability despite topology changes, using sequence numbers. Since the leaves of the broadcast tree rooted at a particular source do not forward updates originating from that source, a dramatic reduction in control traffic is achieved compared to link-state flooding (e.g., OSPF). TBRPF-FT is recommended for sparse networks and when full topology information is needed (e.g., if multiple paths need to be computed to each destination). The method for assigning costs to links is not specified. As an example, the cost of a link could simply be one (for minimum-hop routing), or the link delay plus a constant bias.

TBRPF-PT achieves a further reduction in control traffic, especially in large, dense networks, by providing each node with the state of only a relatively small subset of the network links, sufficient to compute minimum-hop paths to all other nodes. As in the FT mode, a node forwards an update only if the node is not a leaf of the broadcast tree rooted at the source of the update. In addition, a node forwards an update only if it results in a change to the node's source tree (which provides min-hop paths to all other nodes). As a result, each node reports only changes to a relatively small portion of its source tree. TBRPF-PT also allows the computation of “approximately” optimal paths (with the degree of approximation determined by a configurable parameter), in order to achieve a further reduction in control traffic and scalability to networks having a large diameter.

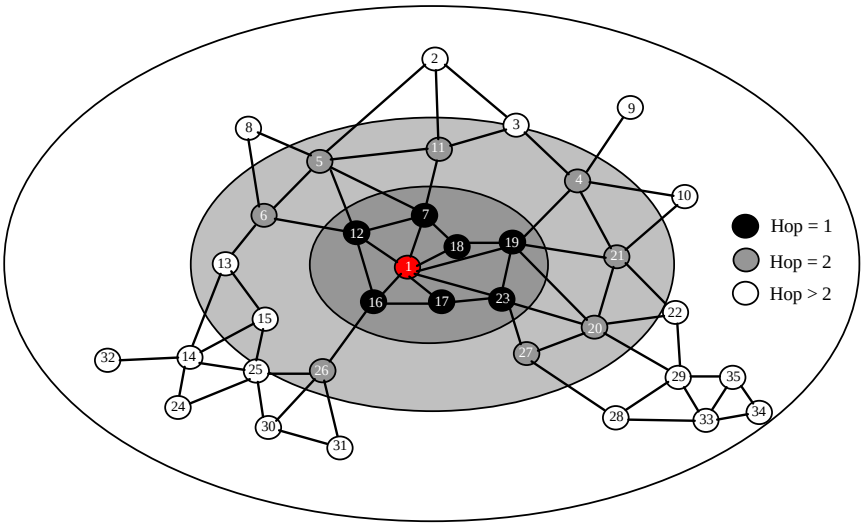
The FT and PT modes of TBRPF use the same neighbor discovery protocol (TND). TND is a new protocol whose HELLO messages are much smaller than existing neighbor discovery protocols such as the one used by OSPF. A HELLO message in TND contains only the IDs of nodes that have recently been heard but with which a 2-way link has not yet been established. In contrast, a HELLO message in OSPF contains the IDs of all neighbors, resulting in a much larger message, especially in dense networks. The use of TND thus contributes to the efficiency of TBRPF. In addition, since HELLO messages are smaller, they can be sent more frequently, resulting in a faster response to topology changes.

Fisheye State Routing Protocol (FSR)

Fisheye State Routing (FSR) [11] is a table-driven or proactive routing protocol. It is based on the link state protocol and has the ability of immediately providing route information when needed. The FSR algorithm for ad hoc networks introduces the notion of multi-level "scope" (the fisheye scope technique) to reduce routing update overhead in large networks.

FSR is functionally similar to link state routing in that it maintains a topology map at each node. The key difference is the way in which routing information is disseminated. In the link state routing protocol OSPF, link state packets are generated and flooded into the network whenever a node detects a topology change. In FSR, link state packets are not flooded. Instead, nodes maintain a link state table based on the up-to-date information received from neighboring nodes, and periodically exchange it with their local neighbors only (no flooding). Through this exchange process, the table entries with larger sequence numbers replace the ones with smaller sequence

numbers. The frequency of these periodically exchanges depends on the hop distance to a destination (i.e., the "scope" relative to that destination, see figure 6). Link State Updates corresponding to far away destinations are propagated with lower frequency than those for close by destinations. The sequenced periodic table update resembles the vector exchange in Destination-Sequenced Distance-Vector Routing (DSDV) where the distances are updated according to the time stamp or sequence number assigned by the node originating the update. Moreover, like in OSPF, a full topology map is kept at each node and shortest paths are computed using this map. The protocol does provide support for unidirectional links, since FSR is link state based routing protocol and directional link states can be included in the FSR update messages.



- Figure 6: Scope of fisheye

FSR provides an implicit hierarchical routing structure. Through updating link state information with different frequencies depending on the fisheye scope distance, FSR scales well to large network size and keeps overhead low without compromising route computation accuracy when the destination is near. The routing accuracy of FSR is comparable with an ideal Link State scheme. By retaining a routing entry for each destination, FSR avoids the extra work of "finding" the destination (as in on-demand routing) and thus maintains low single packet transmission latency. As mobility increases, routes to remote destinations become less accurate. However, when a packet approaches its destination, it finds increasingly accurate routing instructions as it enters sectors with a higher refresh rate. As a result, FSR is more desirable for large mobile networks where mobility is high and the bandwidth is low. By choosing proper number of scope levels and radius size, FSR can be optimized to maintain accurate routes in ad hoc networks.

FSR doesn't trigger any control messages when a link failure is reported. Thus it is suitable for high topology change environment. The broken link will not be included in the next fisheye scope link state message exchange. Sequence number and table refreshment enables the FSR to maintain the latest link state information and loop-free in an unreliable propagation media and highly mobile network.

Summarizing, the main features of FSR are:

- Usage of up-to-date shortest routes
- Robustness to host mobility
- Exchange Partial Routing Update with neighbors
- Reduced Routing Update Traffic

3.2.2 Source-initiated Routing Protocols

Source-initiated on-demand routing protocols are using a different approach than table-driven routing. In on-demand routing a route is only created when desired by the source node. When a node requires a route to a destination, it initiates a route discovery process within the network. This process is completed once a route is found or all possible route permutations have been examined. Once a route has been established, it is maintained until either the destination becomes inaccessible along every path from the source or until the route is no longer desired. Some of the more popular on-demand routing protocols will now be discussed.

Ad hoc On-demand Distance Vector Routing (AODV)

The Ad Hoc On-Demand Distance Vector (AODV) routing protocol [12] builds on the DSDV algorithm described in section 3.2.1. AODV is an improvement on DSDV because it typically minimizes the number of required broadcasts by creating routes on a demand basis, as opposed to maintaining a complete list of routes as in the DSDV algorithm.

The AODV protocol maintains for each node table entry the following information:

- Destination IP Address: IP address for the destination node.
- Destination Sequence Number: Sequence number for this destination.
- Hop Count: Number of hops to the destination.
- Next Hop: The neighbor, which has been designated to forward packet to the destination for this route entry.
- Lifetime: The time for which the route is considered valid.
- Active neighbor list: Neighbor nodes that are actively using this route entry.
- Request buffer.

When a source node wants to send a message to some destination node and does not already have a valid route to that destination, it initiates a path discovery process to locate the other node. It broadcasts a route request (RREQ) packet to all its neighbors. They will then forward the RREQ to their neighbors, and so on, until either the destination or an intermediate node with a “fresh enough” route to the destination is located. During the process of forwarding the RREQ, intermediate nodes record in their route tables the address of the neighbor from which the first copy of the broadcast packet is received, thereby establishing a reverse path. If additional copies of the same RREQ are later received, these packets are discarded. Once the RREQ reaches the destination or an intermediate node with a fresh enough route, the destination/intermediate node responds by unicasting a route reply (RREP) packet back to the neighbor from which it first received the RREQ. As the RREP is routed back along the reverse path, nodes along this path set up forward route entries in their route tables, which point to the node from which the RREP came.

To maintain the routes, the AODV algorithm normally periodically transmits “hello” messages (a special RREP) to its immediate neighbors. These messages are the indication to the neighbors that the node is still present and can be used for routing. If the hello messages stop coming from a particular node, the neighbors can assume that the node has moved away and mark the link to this node as broken. Then all affected nodes are notified by sending a link failure notification (a special RREP) to these nodes.

Dynamic Source Routing (DSR)

Dynamic Source Routing (DSR) [13] allows nodes to dynamically discover a route across multiple networks hops to any destination. Source routing means that each packet in its header carries the complete ordered list of nodes through which the

packet must pass. DSR uses no periodic routing messages, thereby reducing the network bandwidth overhead, conserving battery power and avoiding large routing updates throughout the ad hoc network. DSR relies instead on the MAC layer, the MAC layer should inform the routing protocol about link failures. The two basic mechanisms in DSR are route discovery and route maintenance.

To perform a route discovery, the source node will broadcast a Route Request (RREQ) packet by flooding it though the network. Every node that receives this RREQ searches to its route cache for a route to the requested destination. DSR stores all known routes in its route cache. If no route is found, it will forward the RREQ further and will add its own address to the recorded hop sequence. This request propagates through the network until either the destination or a node with a route to the destination is reached. Then a Route Reply (RREP) will be unicasted back to the originator. The RREP packet contains the sequence of network hops through which it can reach the target.

The function of the route maintenance mechanism is to detect if new routing is required when changes in the network topology occur. These changes could be for instance a host, listed in a source route, which moves out of the wireless transmission range or is turned off and therefore making a route unusable. A failed link is detected by either actively monitoring acknowledgements or passively by running in promiscuous mode, overhearing that a packet is forwarded by a neighboring node.

When route maintenance detects a problem with a route in use, a route error packet is sent back to the source node. When this error packet is received, the hop in error is removed from the hosts route cache and all routes that contain this hop are truncated at this point. The node can then attempt to find another route to its destination. This new route may already be available in its route cache or can be found by invoking a new route discovery.

Temporally Ordered Routing Algorithm (TORA)

Temporally Ordered Routing Algorithm (TORA) [14] is a highly adaptive loop-free distributed routing algorithm based on the concept of link reversal. TORA is proposed to operate in a highly dynamic mobile networking environment. TORA is designed to minimize reaction to topological changes. The main feature in its design is that control messages are typically localized to a very small set of nodes near the occurrence of a topological change. To accomplish this, nodes need to maintain routing information about adjacent (one-hop) nodes. The protocol performs three basic functions:

- Route creation
- Route maintenance
- Route erasure

During the route creation and route maintenance, the nodes use a "*height*" metric to establish a directed acyclic graph (DAG) rooted at the destination. Thereafter, links are assigned a direction (upstream or downstream) based on the relative height metric of neighboring nodes. When a node moves and the DAG route is broken, route maintenance is necessary to reestablish a DAG rooted at the same destination. Upon failure of the last downstream link, a node generates a new reference level what results in the propagation of that reference level by neighboring nodes, effectively coordinating a structured reaction to the failure. Links are reversed to reflect the change in adapting to the new reference level. This has the same effect as reversing the direction of one or more links when a node has no downstream links.

Timing is an important factor for TORA because the "*height*" metric is dependent on the logical time of a link failure; TORA assumes that all nodes have synchronized

clocks (accomplished via an external time source such as the Global Positioning System). TORA's metric is composed of five elements, namely:

- Logical time of a link failure
- The unique ID of the node that defined the new reference level
- A reflection indicator bit
- A propagation ordering parameter
- The unique ID of the node

The first three elements represent the reference level. A new reference level is defined each time a node loses its last downstream link due to a link failure. The route erasure mechanism essentially involves flooding a broadcast clear packet (CLR) throughout the network to erase invalid routes. In TORA there is a potential for oscillations to occur, especially when multiple sets of coordinating nodes are concurrently detecting partitions, erasing routes, and building new routes based on each other. This instability problem of TORA is similar to the “count-to-infinity” problem in distance-vector routing protocols, except that such oscillations are temporary and route convergence will ultimately occur.

Associativity Based Routing (ABR)

The Associativity-Based Routing protocol (ABR) [15] uses a totally different approach to mobile routing. The Associativity-Based Routing protocol is free from loops, deadlock, and packet duplicates, and defines a new routing metric for ad hoc mobile networks. This metric is known as the degree of association stability. In ABR, a route is selected based on the degree of association stability of mobile nodes. Each node periodically generates a beacon to signify its existence. When received by neighboring nodes, this beacon causes their associativity tables to be updated. For each beacon received, the associativity tick of the current node with respect to the beaconing node is incremented. Association stability is defined by connection stability of one node with respect to another node over time and space. A high degree of association stability may indicate a low state of node mobility, while a low degree may indicate a high state of node mobility. Associativity ticks are reset when the neighbors of a node or the node itself move out of proximity. A fundamental objective of ABR is to derive longer-lived routes for ad hoc mobile networks.

The three mechanisms of ABR are:

- Route discovery
- Route reconstruction (RRC)
- Route deletion (RD)

The route discovery is accomplished by a broadcast query and await-reply (BQ-REPLY) cycle. A node desiring a route broadcasts a BQ message in search of mobiles that have a route to the destination. All nodes receiving the query (that are not the destination) append their addresses and their associativity ticks with their neighbors along with QoS information to the query packet. A successor node erases its upstream node neighbors associativity tick entries and retains only the entry concerned with itself and its upstream node. In this way, each resultant packet arriving at the destination will contain the associativity ticks of the nodes along the route to the destination. The destination is then able to select the best route by examining the associativity ticks along each of the paths. When multiple paths have the same overall degree of association stability, the route with the minimum number of hops is selected. The destination then sends a REPLY packet back to the source along this path. Nodes propagating the REPLY mark their routes as valid. All other routes remain inactive, and the possibility of duplicate packets arriving at the destination is avoided.

Route reconstruction (RRC) may consist of partial route discovery, invalid route erasure, valid route updates, and new route discovery, depending on which node(s) along the route move. Movement by the source results in a new BQ-REPLY process. When a route is no longer desired, the source node initiates a route delete (RD) broadcast so that all nodes along the route update their routing tables. The RD message is propagated by a full broadcast, as opposed to a directed broadcast, because the source node may not be aware of any route node changes that occurred during RRCs.

3.2.3 Alternate Path Routing (APR)

A new approach in ad hoc network routing is the use of multiple routes. This kind of routing is called Alternate Path Routing (APR) or Multipath routing. Alternate path routing can be used to improve the quality of service (QoS) of networks. This has led to a number of approaches for alternate path routing. The research in alternate path routing has focused primarily on two key areas:

- the construction of alternate route sets;
- the implementation of policies for traffic distribution among these multiple routes.

To find the set of possible routes in a network a routing protocol, like described in the previous two sections, can be used. With a set of alternate routes in hand, policies are needed to control the use of these routes. In the existing routing protocols it is customary to designate one route as the primary route. The primary route is used until it is no longer able to meet the demands of incoming traffic (for example, due to route failure or congestion). Through a crankback process, the alternate routes are tried, one-by-one, until a route is identified that satisfies the additional traffic load. This approach reduces the fragmentation of network resources that would prevent new connections from being established (due to insufficient bandwidth along one path).

Because Internet traffic consists of packet data streams it is possible to distribute a session's traffic among multiple routes. In high-speed packet networks, network congestion is generally a temporary and local phenomena. Under these circumstances, a reasonable alternate path routing strategy would be to direct traffic along a single shortest hop route, bypassing temporarily congested areas when necessary.

Another key consideration is the frequency of route switching. As the frequency of route transition increases the end-to-end throughput and delay will improve. Although congestion control has been the primary focus of alternate path routing research, alternate path routing can also be used to compensate for route failures. These potential benefits of alternate path routing make it appear an ideal candidate for the bandwidth limited and dynamic mobile ad hoc networks. A proposal for a multipath routing protocol is the Split Multipath Routing protocol (SMR).

Split Multipath Routing (SMR)

The Split Multipath Routing (SMR) protocol for ad hoc networks [16] is an on-demand protocol that builds maximally disjoint routes. The two basic mechanisms, route discovery and route maintenance, are working similar as the in the Dynamic Source Routing (DSR) routing protocol. The main difference is that the SMR scheme uses two routes for each session; the shortest delay route and the one that is maximally disjoint with the shortest delay route. It attempts to build maximally disjoint routes to avoid having certain links from being congested, and to efficiently utilize the available network resources. Providing multiple paths is useful in ad hoc networks because when one of the routes is disconnected, the source can simply use other available routes without performing the route recovery process.

3.3 Comparison of the routing protocols

In this section the routing protocols described in the previous sections will be compared and their distinctive features and their applicability in mobile ad hoc networks will be discussed. Of each of the described routing protocol an overview of their main characteristics will be shown in a table. These characteristics are the routing architecture, loop-free and multicast capabilities, route update and maintenance schemes, and the routing metric algorithm.

3.3.1 Conventional routing protocols

In Table 3 an overview is given with the main characteristics of the two described conventional routing protocols. From the two protocols RIP is the more simple routing protocol. It can be used in smaller networks operating with low to moderate utilization. A maximum of 15 for the metric limits the sizes of networks on which RIP can be used. Another drawback of RIP is that this protocol uses fixed "metrics" to compare alternative routes. It is not appropriate for situations where routes need to be chosen based on real-time parameters such a measured delay, reliability, or load. RIP also takes a long time (some minutes) to stabilize after the failure of a router or a link. During this time, routing loops may occur ("count-to-infinity" problem). Another drawback is the use of only one entry in the routing table. However the routing tables of RIP are relative easier to manipulate than the databases of OSPF. Summarizing, the main problems for using RIP in mobile ad hoc networks are:

- Topology changes are slowly propagated.
- The count-to-infinity problem.
- Moving nodes create confusion (they carry connectivity data, which at the new place are wrong. This will result in fatal routing loops.).
- Table exchange eats bandwidth.

Parameters	RIP	OSPF
Routing architecture	Flat	Hierarchical or flat
Loop-free	No	Yes
Multicast capability	Yes	Yes
Route maintained in	Routing table	Routing database
Frequency of required tables	Only when needed	Periodically
Updates transmitted to	Neighbors	All (by flooding)
Utilizes sequence numbers	No	Yes
Utilizes hello messages	No	Yes
Critical nodes	No	Yes*
Routing metric	Shortest path (distance-vector)	Shortest path (link-state)

* In multi-access networks OSPF uses one designated router to exchange information to all other routers within a LAN

- Table 3: Comparison of the characteristics of the conventional routing protocols

With OSPF each router contains a routing directory (called a "routing database"). The database contains information about interfaces at the router that are operable as well as status information about each neighbor to a router. This database is the same for all participating routers. The need to maintain the up-to-date version of the entire network topology at every node may result in excessive storage and communication overhead in highly dynamic network environments. Periodically the information of the databases is broadcasted (flooded) to all the routers, which will result in a temporally increase of bandwidth consumption. In ad hoc networks this can be a problem due to the characteristic of limited bandwidth of mobile ad hoc networks. Especially when an ad hoc network has a large number of nodes this can greatly affect the performance of the network. A main advantage of this protocol is that separate cost metrics can be computed for each link. OSPF is a link state protocol, which produces a more stable network by getting the routers to act on network changes predictably and simultaneously. Link-state algorithms are also free of the count-to-infinity problem. In

general, OSPF is a more efficient protocol and can effectively determine alternate paths and it minimizes overhead packet traffic

3.3.2 Table-driven ad hoc routing protocols

In Table 4 an overview is shown of the main characteristics of the discussed table-driven routing protocols. The DSDV routing protocol is essentially a modification of the basic Bellman-Ford routing algorithm as used in RIP. The modifications include the guarantee of loop-free routes and a simple route update protocol. DSDV provides only one path to any given destination and its selects the shortest path based on the number of hops to the destination. A small update message can be used for incremental updates so that the entire routing table need not be transmitted for every change in the network topology. However, DSDV is inefficient because of the requirement of periodic update transmissions, regardless of the number of changes in the network topology. This effectively limits the number of nodes that can connect to the network. In CGSR, DSDV is used as the underlying routing protocol. Routing in CGSR occur over cluster heads and gateways. A special cluster head table is necessary in addition to the routing table.

The WRP protocol requires each node to maintain four routing tables. This can lead to substantial memory requirements, especially when the number of nodes in the network is large. Furthermore, the WRP protocol requires the use of hello packets whenever there are no recent packet transmissions from a given node. The hello packets consume bandwidth and disallow a node to enter sleep mode.

Parameters	DSDV	CGSR	WRP	OSLR	TBRPF	FSR
Routing architecture	Flat	Hierarchical	Flat*	Flat	Flat	Hierarchical
Loop-free	Yes	Yes	Yes, but not instantaneous	Yes	Yes	Yes
Multicast capability	No	No	No	Yes	Yes	Yes
Number of required tables	Two	Two	Four	Three	Two	Three
Frequency of required tables	Periodically and as needed	Periodically	Periodically and as needed	Periodically and as needed	When needed	Periodically
Updates transmitted to	Neighbors	Neighbors and cluster head	Neighbors	Neighbors	All nodes (FT) or neighbors (PT)	Neighbors
Utilizes sequence numbers	Yes	Yes	Yes	Yes	Yes	Yes
Utilizes hello messages	Yes	Yes	Yes	Yes	Yes	Yes
Critical nodes	No	Yes (cluster head)	Yes	No	No	No
Routing metric	Shortest path (distance-vector)	Shortest path (distance-vector)	Shortest path (distance-vector)	Shortest path (number of hops)	Shortest path (link-state)	Shortest path (link-state)

* While WRP uses flat addressing it can be used hierarchically

- Table 4: Comparison of the characteristics of table-driven ad hoc routing protocols

Unlike the other described routing protocols OSLR uses a link state routing method. Each node maintains a view of the network topology with a cost for each link. To keep these views consistent, each node periodically broadcasts the link costs to its outgoing links to the other nodes. As a node receives this information, it updates its view of the network topology and applies a shortest path algorithm to choose it next hop for each destination. Some of the link cost in a node's view can be incorrect because of long propagation delays, partitioned networks, etc. This might lead to the formation of routing loops. However, these routing loops will be short-lived because they will disappear in the time it takes a message to traverse the diameter of the network. The use of multipoint relays substantially reduces the packet overhead as compared to pure flooding mechanism technique. A second optimization is achieved by minimizing the contents of the control packets flooded in the network. In contrary to the classic link state algorithm, only a small subset of links with the neighbor nodes is declared instead of all the links, namely the links to those nodes which are its multipoint relay selectors. OLSR provides optimal routes (in terms of number of hops). The OLSR protocol is particularly suitable for large and dense networks as the technique of multipoint relays works well in this context.

Like OSLR, TBRPF is also a link state routing protocol. Unlike OLSR, it does not require the periodic broadcast of topology information. Instead, only differential changes in topology are reported in order to minimize overhead. TBRPF-FT achieves reliability despite topology changes, using sequence numbers. Since the leaves of the broadcast tree rooted at a particular source do not forward updates originating from that source, a dramatic reduction in control traffic is achieved compared to link-state flooding. TBRPF-PT achieves a further reduction in control traffic, especially in large, dense networks, by providing each node with the state of only a relatively small subset of the network links, sufficient to compute minimum-hop paths to all other nodes. Another advantage over other neighbor discovery protocols, such as the one used by OSPF, is that the average size of a HELLO message is much smaller, resulting in reduced message overhead. In addition, since HELLO messages are smaller, they can be sent more frequently, resulting in a faster response to topology changes.

In the link state routing protocol FSR, link state packets are not flooded. Instead, they are exchanged periodically with their local neighbors only. A node maintains a link state table based on the up-to-date information received from neighboring nodes. Sequence numbers are used for entry replacements. The sequenced periodic table update resembles the vector exchange in DSDV, where the distances are updated according to the time stamp or sequence number assigned by the node originating the update. However, in FSR link states rather than distance vectors are propagated. Moreover, like in OSPF, a full topology map is kept at each node and shortest paths are computed using this map. In a wireless environment, a radio link between mobile nodes may experience frequent disconnects and reconnects. The OSPF protocol releases a Link State Update for each such change, which floods the network and causes excessive overhead. FSR avoids this problem by using periodic, instead of event driven, exchange of the topology map, greatly reducing the control message overhead. FSR is more desirable for large mobile networks where mobility is high and the bandwidth is low.

Reviewing the operation and characteristics of each of the existing table-driven routing protocols, one can highlight the following differences. During link failures, WRP works more efficient than DSDV since it only informs neighboring nodes about link status changes. In terms of communication efficiency, since DSDV, CGSR, and WRP use distance vector shortest path routing as the underlying routing protocol, they all have the same degree of complexity during link failures and additions. When using OLSR a change in the topology is detected, the routing table is recalculated to update the routing information. This does not generate or trigger any messages to be transmitted. During link additions, hello messages are used as a presence indicator such that the routing table entry can be updated. In the FSR protocol the updates are only exchanged periodically and not in reaction to topology changes. TBRPF uses the opposite approach, only when a change in the topology is detected a new update will be send.

3.3.3 Source-initiated ad hoc routing protocols

In Table 5 a comparison is presented of the source-initiated on-demand routing protocols. The route discovery procedure employed by AODV is similar to that of DSR; however, there are a couple of important distinctions. The most notable of these is that the overhead of DSR is potentially larger than that of AODV since each DSR packet must carry full routing information, whereas in AODV packets need only contain the destination address. Similarly, the route replies in DSR are larger because they contain the address of every node along the route, whereas in AODV route replies need only carry the destination IP address and sequence number. Also, the memory overhead may be slightly greater in DSR because of the need to remember full routes, as opposed to only next hop information in AODV. A further advantage of AODV over the other on-demand protocols is its support for multicast.

An advantage of DSR over some of the other on-demand protocols is that DSR does not make use of periodic routing advertisements, thereby saving bandwidth and reducing power consumption. Hence, the protocol does not incur any overhead when there are no changes in network topology. Additionally, DSR allows nodes to keep multiple routes to a destination in their cache. Hence, when a link on a route is broken, the source node can check its cache for another valid route. If such a route is found, route reconstruction does not need to be reinvoked. In this case, route recovery is faster than in many of the other on-demand protocols. However, DSR is not scalable to large networks. Furthermore, as previously stated, the need to place the entire route in both route replies and data packets causes greater control overhead than in AODV.

Parameters	AODV	DSR	TORA	ABR
Routing architecture	Flat	Flat	Flat	Flat
Loop-free	Yes	Yes	Yes	Yes
Multicast capability	Yes	No	No	No
Beaconing requirements	No	No	No	Yes
Multiple route possibilities	No	Yes	Yes	No
Route maintained in	Route table	Route cache	Route table	Route table
Utilizes route cache/table expiration timers	Yes	No	No	No
Route reconfiguration methodology	Erase route: notify source	Erase route: notify source	Link reversal: route repair	Localized broadcast query
Routing metric	Freshest and shortest path (distance vector)	Shortest path	Shortest path	Associativity and shortest path and others*

* ABR also uses the route relaying load and cumulative forwarding delay as routing metrics

- Table 5: Comparison of the characteristics of on-demand ad hoc routing protocols

TORA is a “link reversal” algorithm that is best suited for networks with large dense populations of nodes. One of the advantages of TORA is its support for multiple routes. Route reconstruction is not necessary until all known routes to a destination are deemed invalid, and therefore bandwidth can potentially be conserved because of the necessity for fewer route rebuildings. However, TORA's reliance on synchronized clocks limits its applicability. If a node does not have GPS or some other external time source, it cannot use the algorithm. Additionally, if the external time source fails, the algorithm will cease to operate.

ABR is a compromise between broadcast and point-to-point routing, and uses the connection-oriented packet forwarding approach. Route selection is primarily based on the aggregated associativity ticks of nodes along the path. Hence, although the resulting path does not necessarily result in the smallest possible number of hops, the path tends to be longer-lived than other routes. A long-lived route requires fewer route reconstructions and therefore yields higher throughput. ABR, relies on the fact that each node is beaconing periodically. This beaconing requirement may result in additional power consumption.

4 Power Awareness Routing

One of the main objectives of this assignment is to investigate power awareness routing in a wireless IEEE 802.11b ad hoc network. In this chapter the different approaches to power awareness routing in mobile ad hoc networks will be discussed and some of the existing power aware routing protocols will be described. Furthermore a number of specifications which are desirable for the ad hoc network will be proposed and also which routing protocol is the most suitable candidate for the implementation of power awareness routing is discussed.

4.1 Approaches to Power Awareness Routing

In a mobile ad hoc network nodes are often powered by batteries. The power level of a battery is finite and limits the lifetime of a node. Every message sent and every computation performed drains the battery. The main goal of power awareness routing in an ad hoc network is to optimize the lifetime of the nodes and network. In mobile ad hoc networking the power consumption of a node can be divided according to functionality into:

- The power utilized for the transmission of a message;
- The power utilized for the reception of a message;
- The power utilized while the system is idle.

To illustrate, Table 6 lists power consumption numbers for several wireless cards (all have a power supply of 5V).

Wireless PC Card	Doze mode	Receiver mode	Transmit mode
Orinoco PC Card	9 mA	185 mA	285 mA
Cisco Aironet 350	15 mA	270 mA	450 mA
BreezeCom SA-PCR	30 mA	285 mA	360 mA

- Table 6: Power consumption of wireless PC cards

This suggests two complementary levels at which power consumption can be optimized in wireless communication:

- minimizing power consumption during the idle time by switching to sleep mode; this is known as *Power Management*;
- minimizing power consumption during communication, that is, while the system is transmitting and receiving messages; this is known as *Power Control*.

Existing power awareness routing protocols can roughly be classified as a power management or power control protocol. In IEEE 802.11 there is already a type of power management implemented (see section 2.2.3). The implantation of power awareness routing in this assignment (see chapter 6) is only focusing on power control schemes. However, to make power awareness schemes for wireless ad hoc networks most effective, an efficient message routing algorithms, coupled with good solutions for optimizing power consumption during the idle time can be used.

In power control routing protocols several metrics can be used to optimize power awareness routing. Minimizing the energy consumed for each message is an obvious solution that optimizes locally the power consumption. Other useful metrics include minimizing the variance in each computer power level, minimizing the ratio of cost/packet, and minimizing the maximum node cost. An effective routing protocol should not only focus on individual nodes in the system but also focus on the system as a whole. Otherwise this might quickly lead to a system in which nodes have high residual power but the system is not connected because some critical nodes have been depleted of power. This can be optimized by focusing on a global metric in the

routing path calculation to maximize the lifetime of the network. A effective routing scheme should consume less energy and should avoid nodes with small residual energy since we would like to maximize the minimum lifetime of all nodes. Different routing schemes can be utilized, but the two most extreme solutions to power awareness routing for a message are:

- compute a path that maximizes the minimal power consumption; that is, use the path that requires the least power to transmit and receive a message, hereby keeping the power consumption needed to communicate as low as possible;
- compute a path that maximizes the minimal residual power in the network; that is, use a path according to the residual energy of the nodes, hereby maximizing the lifetime of all nodes and the lifetime of the network.

Obviously, both of these can not be optimized at the same time, which means there is a tradeoff between the two. In the beginning when all the nodes have plenty of energy, the minimum total consumed energy path is better off, whereas towards the end avoiding the small residual energy node becomes more important. Ideally, the link cost function should be such that when the nodes have plenty of residual energy, the power consumption term should be applied, while if the residual energy of a node becomes small the residual energy term should be applied.

4.2 Power Awareness Routing Protocols

Today, there are already a number of power awareness routing protocols developed and they can roughly be classified as a *power management* or *power control* protocol. In this section a selection of existing power awareness routing protocols will be briefly discussed to get a view on some of the different approaches to power awareness routing. Of the discussed power awareness routing protocols, COMPOW, ISAIAH, MRPC and PARO are more focused on power control, and BECA and AFECA are focused on power management.

COMPOW protocol

COMPOW [17] is a protocol for power control in ad hoc networks. COMPOW tries to maximize the traffic carrying capacity of the entire network, extend battery life through providing low power routes, and reduce the contention at the MAC layer. COMPOW assumes bidirectional links and that the transmission power of the nodes is adjustable. The essence of the COMPOW protocol is an asynchronous, distributed, and adaptive algorithm which finds the smallest common power (COMmon POWer) level (transmission power level) at which the network is still connected. Multiple routing tables are maintained, one for each of the transmit power levels available. A routing table for a certain power level is constructed by sending and receiving hello messages at that particular power level. The optimum power level selected for the node is the smallest power level whose routing table has the same number of entries as that of the routing table at the maximum power level. This is done by a *power control agent* which takes input from the various routing tables and decides the optimum power level. COMPOW is designed to be used in conjunction with any routing protocol that pro-actively maintains a routing table.

Infra-Structure Aodv for Infrastructured Ad Hoc networks (ISAIAH)

ISAIAH [18] is an ad hoc routing protocol based on the Ad-hoc On-demand Distance Vector (AODV) routing protocol. In ISAIAH is assumed that in wireless networks some nodes may be equipped with virtually unlimited power supplies, while others have to rely on battery power. This allows the creation of “infrastructured” ad hoc networks by the deployment of pseudo base-stations (PBSs). PBSs are nodes that have constant

power supply (e.g., through a power outlet or a car battery), do not move, and are present just to act as routers and forward packets for other nodes, thus allowing the mobile nodes to save power. ISAIAH tries to select routes that go through PBSs instead of through mobile nodes to reduce the amount of power spent by these mobile nodes. Furthermore, it allows nodes to enter a power-saving mode, significantly reducing the power consumption compared to AODV.

Maximum Residual Packet Capacity (MRPC)

MRPC [19] is a power awareness routing algorithm for energy-efficient routing that increases the operational lifetime of multihop wireless networks. MRPC identifies the capacity of a node not just by its residual battery energy, but also by the expected energy spent in reliably forwarding a packet over a specific link. This scheme better captures scenarios where link transmission costs also depend on physical distances between nodes and the link error rates. Using a max-min formulation, MRPC selects a path, given the current battery power levels at the nodes, that maximizes the total number of packets that may be ideally transmitted over that path (assuming that all other flows sharing that path do not transmit any further traffic). A variant of MRPC is CMRPC, that switches from minimum energy routing to MRPC only when the packet forwarding capacity of nodes falls below a threshold.

Power-Aware Routing Optimization (PARO)

PARO [20] is a power awareness routing optimization protocol for wireless networks where all nodes are located within the maximum transmission range of each other. PARO minimizes the transmission power necessary to forward packets between wireless devices. In PARO, intermediate nodes forward packets between source and destination pairs even if source-destination pairs are located within direct transmission range of each other, hereby reducing the total transmission power consumed by wireless devices (transmitting a packet over one long distance requires more power than transmitting the packet over two shorter distances). This operation requires that radios are capable of adjusting transmission power on a per-packet basis. PARO uses packet forwarding to increase the operational lifetime of the nodes in the network, by reducing the transmission power necessary to deliver packets in the network. PARO is applicable to wireless networks where all nodes are located within transmission range of each other. In wireless networks with nodes out-of-range with each other, a layer 3 ad hoc routing protocol (e.g., MANET routing protocol) should be used above PARO.

BECA/AFECA algorithms

BECA/AFECA [21] are two algorithms for routing in energy-constrained, ad hoc, wireless networks. Nodes running these algorithms can trade off energy dissipation and data delivery quality according to application requirements. These algorithms work above existing on-demand ad hoc routing protocols, such as AODV and DSR, without the need to modify the underlying routing protocols. The basic schema of the *basic energy-conserving algorithm* (BECA) is that nodes to not need to be listening and consuming power when they are not involved in sending, forwarding, or receiving data. In BECA three states are defined: *sleeping*, *listening* and *active*. This concept is similar to the Power Management in the MAC level of IEEE 802.11 (see section 2.2.3), except that BECA uses higher-level information to turn off the radio to reduce the substantial energy dissipated during the idle state. The second algorithm is the *adaptive fidelity energy-conserving algorithm* (AFECA). This algorithm uses observations about node density to increase the time the radio is powered off. When many equivalent nodes are able to forward data, they power off for longer intervals. In a sense, AFECA adapts the number of nodes participating in ad hoc routing to keep a constant number of nodes that will route packets to reduce energy consumption.

4.3 The implementation of power awareness routing

4.3.1 The Specifications of the network

As we have seen in the previous chapters there are many different routing protocols. Generally, these protocols are designed for specific configurations of networks. Each protocol design aims to optimize certain specific features and characteristics of the mobile ad hoc network. Which routing protocol is best suited for implementing power awareness routing depends on the specifications that we propose for the mobile ad hoc network test bed.

As we have seen in the earlier chapters there are a number of issues, which are important for the design of an ad hoc network. To demonstrate power awareness routing a mobile ad hoc network test bed that can perform this task is needed. This requires extra functionality for the design of the ad hoc network. Depending on the needed functionality and the available resources for the mobile ad hoc network a number of specifications are proposed. Not all of these specifications are required for the implementation of power awareness routing and the experiments performed on the implemented power awareness routing prototype. However for possible future work (for instance the implementation of the power awareness routing prototype into a real wireless ad hoc network) these specifications may be desirable.

The following specifications are proposed:

- Power awareness routing implementation;

  One major goal of this assignment is to implement power awareness routing. This demands certain qualifications of the routing protocol. Depending on their design philosophy not every routing protocol is suitable for implementing power awareness routing. Furthermore the routing protocol should cope efficiently with the characteristics of ad hoc networks. Mobility, potentially very large number of mobile nodes, unpredictable changes of topology, and limited resources (like bandwidth and power) are some of the issues a routing protocol has to deal with and determine its efficiency. These issues make the selection of the routing protocol of vital importance for the functioning of the ad hoc network.

- All wireless nodes must be able to communicate with each other;

  Every wireless node must be able to communicate with every other node participating in the ad hoc network. When two nodes are outside their transmission ranges and they cannot directly communicate, they should find a path via other nodes to communicate. This means that the ad hoc network requires multihop capabilities. This shows the necessity of implementing a routing protocol. The two main functions of the routing protocol are the selection of possible routes for the source node to the destination node and to forward the messages to their correct destination.

- All wireless nodes must have the same functionality;

  Within the mobile ad hoc network there must be no difference between the nodes, each node participating in the network must be able to acts both as host and a router and must therefore be willing to forward packets for other nodes. This requires a flat network architecture. New nodes must always be able to join the ad hoc network without the need of extra configurations. However it may be preferable to allow all wireless nodes to communicate with an access point, to provide extra services like Internet, and access to servers and printers.

- The ad hoc network must support mobility of the wireless nodes;

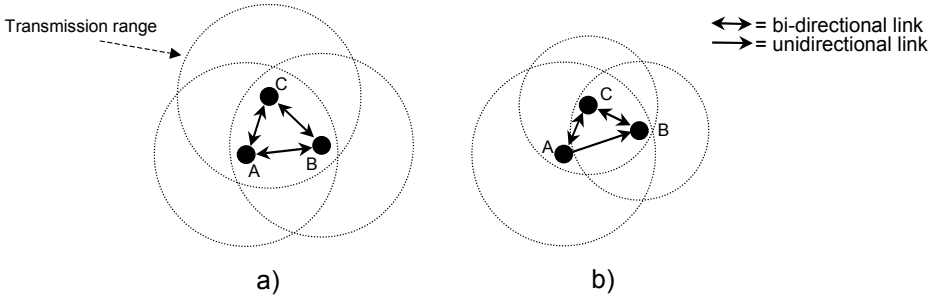
  Wireless ad hoc networks have by definition a dynamic topology. Wireless nodes will be mobile and their positions will change in time. Our ad hoc network must be able to cope smoothly with the mobility of the different nodes. In other words, the position of a wireless node must not have any influence in the routing process.

- All the wireless devices are compliant with the IEEE 802.11b standard;

  Assumed is that all the wireless nodes will be equipped with 802.11b network devices. The use of IEEE 802.11b as the underlying wireless technology puts limitations to the functionality of the network. For instance, IEEE 802.11b does not support any form of multihop routing or the use of unidirectional links (see section 2.2). Because IEEE 802.11 only defines the two lowest layers of the OSI model (the data link and physical layer), any functionality like multihop of unidirectional links therefore must be implemented on a higher level (layer 3 or higher).

- Support for unidirectional links;

  Another aspect that is favorable is the support for unidirectional links. Bi-directional links are typically assumed in the design of routing algorithms, and many algorithms are incapable of functioning properly over unidirectional links. Nevertheless, unidirectional links can and do occur in wireless networks. For example, when we have an ad hoc network with nodes who lie in each other transmission range (see figure 7a), the nodes can all talk directly to each other and they can all established bi-directional connections with the other nodes. However, in figure 7b, we have the situation that the nodes have different transmission ranges (for instance because they are equipped with different radio transmitters). In this situation it can occur that node A with a greater range can reach node B with a smaller range, but that node B cannot reach node A because node A lies beyond its communication range. To reach node A node B has to relay its traffic through node C. In a routing protocol that does not support unidirectional links, node A also has to relay its traffic for node B through node C, because the unidirectional link with node B is not supported. However, when the routing protocol would support unidirectional connections, node A could send its traffic directly to node B, hereby saving one hop in the routing process and making routing more efficient. Therefore it is favorable that a routing protocol supports unidirectional links. However, some MAC protocols, such as IEEE 802.11 do not support unicast data packet transmissions over bi-directional links, due to the required bi-directional exchange of RTS and CTS packets in these protocols and due to the link-level acknowledgement feature in IEEE 802.11.



- Figure 7: Example of bi- and unidirectional links

4.3.2 Candidates for power awareness routing implementation

One of the main objectives of this assignment is to implement power awareness routing into an existing routing protocol. In this section the different design philosophies of the routing protocols described in chapter 2 and 3 will be discussed and the routing protocol which is the most suitable candidate for the implementation of power awareness routing will be introduced.

As we have seen in chapter 2 and 3 the existing wireless routing protocols can generally be classified into two categories according to their design philosophy, namely: table-driven protocols (proactive) and source-initiated or demand-driven protocols (reactive). Another classification that can be make is on the basis of the routing algorithm, namely: distance vector routing or link state routing. Historically, the first type of routing scheme used in early packet networks was the distance vector routing. In the distance vector routing algorithm (also known as the Bellman-Ford algorithm) every router maintains a table with the best distance to every destination and the path to reach it. The tables will be kept up-to-date by exchanging information with its neighbors. The main advantages of distance vector routing are its simplicity and computation efficiency. However, important drawbacks are that distance vector routing algorithms suffer from slow convergence and have the tendency of creating routing loops. The ad hoc routing protocols based on the distance vector routing algorithm distinguish in the approach how to solve the looping problem. However, for the problem of slow convergence there is not yet a good solution proposed.

The link-state approach does not have the problems of slow convergence and routing loops. In link state protocols like OSPF, global network topology information is maintained in all routers by the periodic flooding of Link State Update by each node. Any link change triggers an immediate update. As a result, the time required for a router to converge to a new topology is much less than in the distance vector approach. Due to global topology knowledge, preventing routing loops is much easier.

The implementation of power awareness routing in an ad hoc network requires the ability to calculate the optimum route in an ad hoc network based on certain quality-of-service (QoS) parameters. In principle, there are no restrictions to which QoS parameter is used as metric. Examples of possible metrics are the available bandwidth, the number of hops, delays or the power status. Combinations of different parameters can also be used as metric. Most ad hoc routing protocols are designed to determine the route with a shortest path algorithm based on the number of hops. Some of the distance vector protocols can be adapted to use other metrics. In distance vector routing the path is calculated based on the prices of the separate links. However, for full QoS routing we also want the ability to make routing decisions based on the metric of the entire route. This requires that a full overview of the topology is available when the optimum route is calculated. For instance the knowledge of the current topology of the network can also be used to predict future changes in the network topology, this information can then be used to guarantee a certain QoS in the future. Therefore the best option is a link-state table-driven protocol.

There are a number of link state ad hoc routing protocols available, however we have to make a distinction between partial-topology link state protocols and full-topology link state protocols. Partial-topology link state protocols, like OLSR, provide each node with sufficient topology information to compute at least one path to each destination. However, in contrary to the classic link state algorithm only the link costs of a small subset of links with the neighbor nodes are declared instead of the link costs of all the links. This method is used to reduce the packet overhead as compared to the pure flooding mechanism of the classic link state algorithm.

For full QoS routing we have to use a full-topology link state protocol. Full-topology link state protocols have the following advantages over partial-topology protocols:

- alternate paths and disjoint paths are immediately available, allowing faster recovery from failures and topology changes;
- optimum paths can be computed subject to any combination of quality-of-service (QoS) constraints and objectives.

Another aspect that is favorable is the support for unidirectional links. Bi-directional links are typically assumed in the design of routing algorithms, and many algorithms are incapable of functioning properly over unidirectional links. Nevertheless, unidirectional links can and do occur in wireless networks. Oftentimes, a sufficient number of duplex links exist so that usage of unidirectional links is of limited added value. However, in situations where a pair of unidirectional links (in opposite directions) form the only bi-directional connection between two ad hoc regions, the ability to make use of them is valuable. Some MAC protocols, however, such as IEEE 802.11 limit unicast data packet transmissions to bi-directional links, due to the required bi-directional exchange of RTS and CTS packets in these protocols and due to the link-level acknowledgement feature in IEEE 802.11.

Which full-topology link state routing protocol is the best option for implementing into the power awareness routing prototype? The three main candidates are TBRPF-FT, FSR and the conventional link state protocol OSPF. All three protocols can be used for full QoS routing. The two ad hoc routing protocols TBRPF-FT and FSR are based on the OSPF protocol and are improved for usage in ad hoc networks. However, to achieve this some limitations were introduced compared to OSPF. The main drawback of the usage of OSPF in mobile ad hoc networks is that OSPF uses flooding to broadcast the link state messages. This will periodically result in a temporally increase of bandwidth consumption. In ad hoc networks this can be a problem due to the characteristic of limited bandwidth of mobile ad hoc networks. The two ad hoc protocols are designed to limit the amount of control traffic needed.

In FSR, the fisheye scope technique allows exchanging link state messages at different intervals for nodes within different fisheye scope distance, which reduces the link state message size. Further optimization allows FSR only broadcast topology message to neighbors in order to reduce the flooding overhead. With these optimizations, FSR significantly reduces the topology exchange overhead and scales well to large network size. However, when we want to use FSR for power awareness routing these features can become a problem. Because in FSR the information of far away destinations is not always up-to-date, routing decisions can be made on incorrect link state information. Especially with power awareness routing this can become a problem, because at certain times it may be necessary not to use certain nodes with a low power status.

To reduce the control traffic TBRPF-FT uses the concept of reverse-path forwarding instead of link-state flooding. Furthermore TBRPF uses a new neighbor discovery protocol (TND), which uses HELLO messages that are much smaller than existing neighbor discovery protocols such as the one used by OSPF. The use of TND thus contributes to the efficiency of TBRPF. However, one disadvantage of TBRPF is that it does not support unidirectional links; instead it uses only bi-directional links (as in IEEE 802.11).

Summarizing, the FSR protocol is of the three protocols probably the least eligible candidate for implementing into the power awareness routing prototype, due to the usage of the fisheye scope technique. The TBRPF-FT protocol can well be used for full QoS routing, however unidirectional links are not supported. The main drawback of OSPF in ad hoc networks is the use of the flooding technique for Link State Updates. However, full QoS routing and unidirectional links are supported by OSPF. TBRPF broadcasts, unlike OSPF, only Link State Updates when changes in topology are reported. This minimizes the overhead and making TBRPF more efficient in mobile ad

hoc networks than OSPF. Therefore will TBRPF-FT be the best option for implementing power awareness routing in a mobile ad hoc network. However, at the start of this assignment OSPF was the only available routing protocol which could be implemented in our test network and adapted for power awareness routing. Therefore the power awareness routing prototype will be implemented into OSPF.

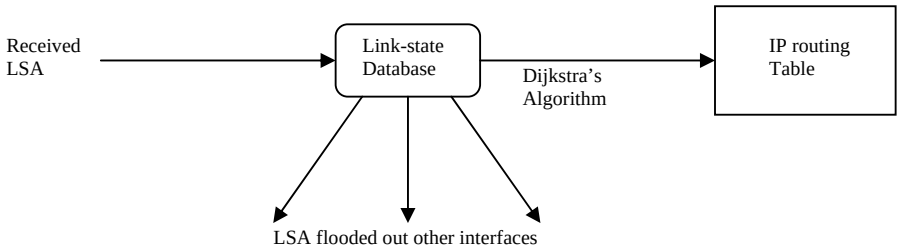
5 The OSPF Routing Protocol

In the previous chapter was concluded that the main candidate for the implementation of the power awareness routing scheme is the conventional link state protocol OSPF [5]. In this chapter the OSPF protocol will be described in more detail.

5.1 Introduction

The *Open Shortest Path First* (OSPF) protocol is a link-state Interior Gateway Protocols (IGP) originally designed to compete with RIP. OSPF is designed to provide quick convergence with only a small amount of routing control traffic, even in autonomous systems (ASs) with a large number of routers. At the moment OSPF is the recommended IGP for the Internet. This is a recommendation made by the *Internet Engineering Task Force* (IETF) to encourage all makers of Internet routers to implement the OSPF protocol.

As a link state protocol, the core of OSPF consists of creating and maintaining a distributed replicated database (called the *link-state database*). Each OSPF router originates one or more *link-state advertisements* (LSAs) to describe its local part of the routing domain. Taken together, the LSAs form the link-state database, which is used as input to the routing calculations. As long as every OSPF router has an identical link-state database, OSPF calculates loop-free paths; most of the protocol machinery within OSPF is dedicated to keeping the database synchronized between routers. Figure 8 shows schematically how OSPF operates.



- Figure 8: Operation of the OSPF protocol. OSPF LSAs received on one interface are installed in the link-state database and flooded out the router's other interfaces. From the link-state database, an OSPF router calculates its routing table, using Dijkstra's Shortest Path First (SPF) algorithm.

5.2 Link-State Algorithm

OSPF is a link state protocol, which means that routing decisions are made based on the status of the connections (links) between the routers in the network. We could think of a link as being an interface on the router. The state of the link is a description of that interface and of its relationship to its neighboring routers. A description of the interface would include, for example, the IP address of the interface, the mask, the type of network it is connected to, the routers connected to that network and so on. The collection of all these link-states would form the link-state database.

The link-state algorithm forms the foundation of the OSPF protocol. This algorithm is used by OSPF to build and calculate the shortest path to all known destinations. The algorithm is quite complicated, but the algorithm can be described, in a very high level, simplified way, by the following steps:

1. Upon initialization or due to any change in routing information, a router will generate a link-state advertisement (LSA). This advertisement will represent the collection of all link-states on that router.

2. All routers will exchange link-states by means of flooding. Each router that receives a link-state update should store a copy in its link-state database and then propagate the update to other routers.

3. After the database of each router is completed, the router will calculate a Shortest Path Tree to all destinations. The router uses the Dijkstra algorithm to calculate the shortest path tree. The destinations, the associated cost and the next hop to reach those destinations will form the IP routing table.

4. In case no changes in the OSPF network occur, such as cost of a link or a network being added or deleted, OSPF should be very quiet. Any changes that occur are communicated via link-state packets, and the Dijkstra algorithm is recalculated to find the shortest path.

5.3 Shortest Path Algorithm

The shortest path is calculated using the Dijkstra algorithm. The algorithm places each router at the root of a tree and calculates the shortest path along the actual links of the network to each destination based on the cumulative cost required reaching that destination. Each router will have its own view of the topology even though all the routers will build a shortest path tree using the same link-state database.

The cost (also called metric) of an interface (link) in OSPF is an indication of the overhead required to send packets across a certain interface. By default, the cost of an interface is calculated based on the available bandwidth. The cost of an interface is then inversely proportional to the bandwidth of that interface. A higher bandwidth indicates a lower cost. There is more overhead (higher cost) and time delays involved in crossing a 56k serial line than crossing a 10M Ethernet line. The standard formula used to calculate the cost is:

$$Cost = \frac{100000000}{Bandwidth\ (in\ bps)}$$

For example, to cross a 10M Ethernet line the costs will be 10EXP8/10EXP7 = 10 and to cross a T1 line the cost will be 10 EXP8/1544000 = 64.

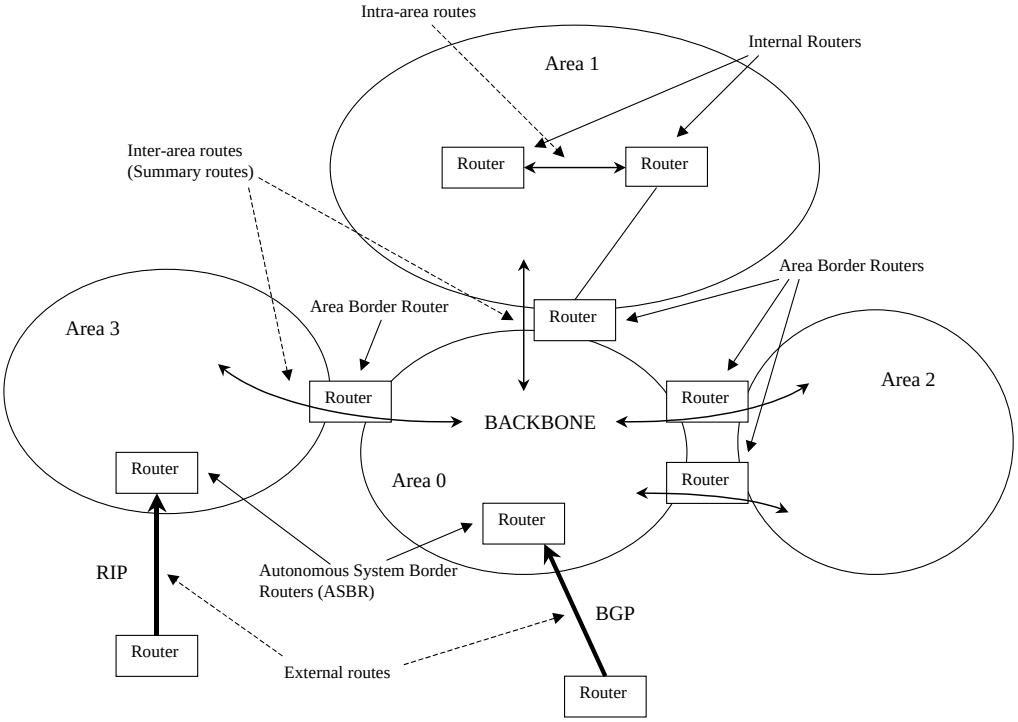
However, in the OSPF protocol a network administrator is free a choosing another parameter as metric. In principle, there are no restrictions to which parameter is used as metric. In addition to the available bandwidth other possible metrics could be the number of hops, the time delays or combinations of various parameters. Important to note is that the value of the cost is limited (depending on the router software) and usually must lie in the following interval: $1 \leq \text{cost} \leq 65535$.

5.4 Areas and Border Routers

As previously mentioned, OSPF uses flooding to exchange Link State Updates between routers. Any change in routing information is flooded to all routers in the network. To limit the number of Link State Updates and to put a boundary on the explosion of Link State Updates in an OSPF domain a routing hierarchy can be implemented. The routing domain can be divided into regions called OSPF *areas*. Flooding and calculation of the Dijkstra algorithm on a router is limited to changes within an area. All routers within an area have the exact link-state database.

An area is interface specific. A router that has all of its interfaces within the same area is called an internal router (IR). A router that has interfaces in multiple areas is called an *area border router* (ABR), they have the additional duty of disseminating routing information or routing changes between areas. Routing information from other protocols can be imported into the OSPF domain and readvertised by OSPF routers as *external routing information*. Routers that act as gateways (redistribution) between OSPF and other routing protocols (like RIP, BGP) or other instances of the OSPF routing process are called *autonomous system border routers* (ASBR). Any router can be an ABR or an ASBR.

There are special restrictions when multiple areas are employed in OSPF. If more than one area is configured, one of these areas has to be defined as area 0 (often written as area 0.0.0.0, since OSPF area ID's are typically formatted as IP addresses). This area is called the backbone. The backbone must be at the center of all other areas; i.e. all areas have to be physically connected to the backbone. The reasoning behind this is that OSPF expects all areas to inject routing information into the backbone and in turn the backbone will disseminate that information into other areas. The following figure illustrates the routing hierarchy of an OSPF network:



- Figure 9: The routing hierarchy in an OSPF network

5.5 OSPF Routing Protocol Packets

The OSPF protocol runs directly over IP. OSPF does not provide any explicit fragmentation/reassembly support. When fragmentation is necessary, IP fragmentation/reassembly is used. OSPF protocol packets have been designed so that large protocol packets can generally be split into several smaller protocol packets. This practice is recommended; IP fragmentation should be avoided whenever possible.

According to their function OSPF distinguished five types of packets. The OSPF packet types are listed below in Table 7.

	Type Packet name	Protocol function
1	Hello	Discover/maintain neighbors
2	Database Description	Summarize database contents
3	Link State Request	Database download
4	Link State Update	Database update
5	Link State Ack	Flooding acknowledgement

- Table 7: OSPF packet types

The OSPF protocol uses Hello packets to discover and maintain neighbor relationships. The Database Description and Link State Request packets are used in the forming of adjacencies. OSPF's reliable update mechanism is implemented by the Link State Update and Link State Acknowledgment packets.

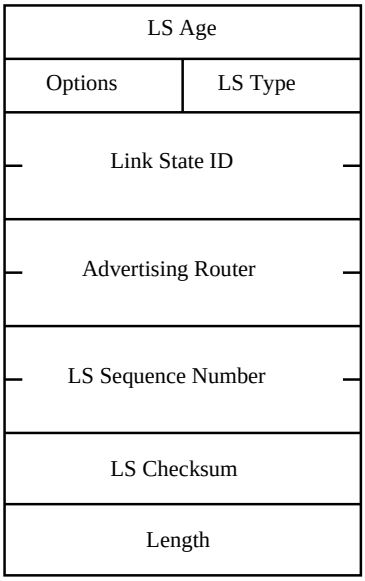
Each Link State Update packet carries a set of new link state advertisements (LSAs) one hop further away from their point of origination. A single Link State Update packet may contain the LSAs of several routers. Each LSA is tagged with the ID of the originating router and a checksum of its link state contents. Each LSA also has a type field; the different types of OSPF LSAs are listed below in Table 8. OSPF routing packets (with the exception of Hellos) are sent only over adjacencies.

LS Type	Advertisement Description
1	Router Link advertisements. Originated by all routers. This LSA describes the collected states of the router's interfaces to an area. Flooded throughout a single area only.
2	Network Link advertisements. Originated for broadcast and NBMA networks by the Designated Router (DR). This LSA contains the list of routers connected to the network. Flooded throughout a single area only.
3 or 4	Summary Link advertisements. Originated by area border routers, and flooded throughout the LSA's associated area. Each summary-LSA describes a route to a destination outside the area, yet still inside the AS (i.e., an inter-area route). Type 3 summary-LSAs describe routes to networks. Type 4 summary-LSAs describe routes to AS boundary routers.
5	AS external link advertisements. Originated by AS boundary routers, and flooded throughout the AS. Each AS-external-LSA describes a route to a destination in another Autonomous System. Default routes for the AS can also be described by AS-external-LSAs.

- Table 8: The types of link-state advertisements

As indicated above, the router links are an indication of the state and cost of the interfaces on a router belonging to a certain area. Each router will generate a router link for all of its interfaces. Network Links are generated by a *Designated Router* (DR) on a segment (DRs will be discussed later). This information is an indication of all routers connected to a particular multi-access segment such as Ethernet, Token Ring and FDDI (NBMA also). Summary links are generated by ABRs; this is how network reachability information is disseminated between areas. Normally, all information is injected into the backbone (area 0) and in turn the backbone will pass it on to other areas. ABRs also have the task of propagating the reachability of the ASBR. This is how routers know how to get to external routes in other ASs. External Links are an indication of networks outside of the AS. These networks are injected into OSPF via redistribution. The ASBR has the task of injecting these routes into an autonomous system.

In order to maintain and to organize the link-state database and to enable the orderly updating and removal of LSAs, each LSA must provide some bookkeeping information, as well as topological information. All OSPF LSAs begins with a standard 20-byte header, which carries this bookkeeping information. This LSA header is shown in Figure 10.



- Figure 10: The LSA header

The LS age field indicates the age of the LSA in seconds. It is set to 0 when the LSA is originated. Under normal circumstances, the LSA Age field ranges from 0 to 30 minutes; if the age of an LSA reaches 30 minutes, the originating router will refresh the LSA by flooding a new instance of the LSA, incrementing the LS sequence number and setting the LS Age to 0 again. If the originating router has failed, the age of the LSA continues to increase until the value of “MaxAge” (default 1 hour). At that time, the LSA will be deleted from the database. The maximum value that the LS Age field can have is 1 hour. To ensure that all routers remove the LSA more or less at the same time, without depending on a synchronized clock, the LSA is reflooded at that time. All other routers will then remove their database copies on seeing the “MaxAge” LSA being flooded.

Because of this process it is possible that when an LSA’s originating router has failed, it can take as long as an hour for the LSA to be removed from other routers link-state databases. However, OSPF guarantees that the LSA will not interfere with the routing table calculation, by requiring that a link must be advertised by the routers at both ends of the link before using the link in the routing calculation. The LS Age field is also examined when a router receives two instances of an LSA, were both having identical LS sequence numbers and LS checksums. An instance of age “MaxAge” is then always accepted as most recent; this allows old LSAs to be flushed quickly from the routing domain. Otherwise, if the ages differ by more than “MaxAgeDiff” (default 15 minutes), the instance having the smaller age is accepted as most recent.

The Options field can indicate that an LSA deserves special handling during flooding or routing calculations. An OSPF link-state database might consist of many thousands of LSAs. Individual LSAs must be distinguished during flooding and the various routing calculations. OSPF LSAs are identified by three fields found in the common LSA header: LS Type, Link State-ID, and Advertising Router. The LS Type (link-state type) field indicates the function of the LSA. OSPF distinguishes five types of LSA, numbered 1 to 5 (see Table 8).

The Link State ID field identifies the piece of the routing domain that is being described by the LSA. Depending on the LSA's LS type, the Link State ID takes on the values listed in Table 9.

LS Type	Link State ID
1	The originating router's Router ID.
2	The IP interface address of the network's Designated Router.
3	The destination network's IP address.
4	The Router ID of the described AS boundary router.
5	The destination network's IP address.

- Table 9: The LSA's Link State IDs.

The Advertising Router field specifies the OSPF Router ID of the LSA's originator. For router-LSAs, this field is identical to the Link State ID field. A router can easily identify its self-originated LSAs as those LSAs whose Advertising Router is set to the routers's own Router ID. Routers are allowed to update or to delete only self-originated LSAs. Network-LSAs are originated by the network's Designated Router (DR), while Summary-LSAs are originated by ABRs and AS-external-LSAs are originated by ASBRs.

The LS sequence number field is a signed 32-bit integer. It is used to detect old and duplicate LSAs. The space of sequence numbers is linearly ordered. The larger the sequence number (when compared as signed 32-bit integers) the more recent the LSA.

The LS checksum field is the checksum of the complete contents of the LSA, excepting the LS age field. The LS age field is excepted so that an LSA's age can be incremented without updating the checksum. The checksum is used to detect data corruption of an LSA. This corruption can occur while an LSA is being flooded, or while it is being held in a router's memory. The LS checksum field cannot take on the value of zero; the occurrence of such a value should be considered a checksum failure. In other words, calculation of the checksum is not optional.

The Length field contains the length, in bytes, of the LSA, counting both LSA header and contents. An LSA can range in size from 20 bytes (the size of the LSA header) to over 65,000 bytes. However, because the LSAs must eventually be transported within an IP packet, the length cannot go all the way to 65,535 bytes (the maximum size of an IP packet). In practice, almost all LSAs will be small, the size will not likely exceed more than a few hundred bytes.

5.6 OSPF Network Types

We have already mentioned that the OSPF protocol is designed for use in the Internet. Because the Internet contains networks of many different network technologies, like for instance Ethernet, 805.5 Token Ring, FDDI, Frame Relay, ATM, packet radio, and so on, OSPF must be able to run over all these networks. OSPF runs over all these networks, although its operation can differ depending on the type of network. The differences in the way that OSPF runs over these various network technologies can be grouped as follows:

- Neighbor discovery and maintenance. OSPF always accomplishes this task through its Hello protocol, but the Hello protocol runs differently on different network types.
- Database synchronization. How does one synchronize the link-state database over the subnet? Which routers become adjacent, and how does reliable flooding takes advantage of any special properties that the network might provide?
- Abstraction. In the OSPF link-state database, how does one represent the subnet and router connectivity over the network?

OSPF divides the various network technologies into the following classes:

- Point-to-Point networks;
- broadcast networks;
- non-broadcast multi-access (NBMA) networks;
- Point-to-MultiPoint networks.

The most straightforward network model is a point-to-point network. A point-to-point network joins a single pair of routers. A 56Kb serial line is an example of a point-to-point network.

Broadcast networks support many (more than two) attached routers, together with the capability to address a single physical message to all of the attached routers (broadcast). Neighboring routers are discovered dynamically on these nets using OSPF's Hello Protocol. The Hello Protocol itself takes advantage of the broadcast capability. The OSPF protocol makes further use of multicast capabilities, if they exist. Each pair of routers on a broadcast network is assumed to be able to communicate directly. Examples of broadcast networks are Ethernet, Token Ring and FDDI networks.

Non-broadcast networks support many (more than two) routers, but have no broadcast capability. Neighboring routers are maintained on these nets using OSPF's Hello Protocol. However, due to the lack of broadcast capability, some configuration information may be necessary to aid in the discovery of neighbors. On non-broadcast networks, OSPF protocol packets that are normally multicast need to be sent to each neighboring router, in turn. Examples of non-broadcast networks are Frame Relay, X.25 Public Data Network (PDN), ATM and packet radio networks.

OSPF runs in one of two modes over non-broadcast networks. The first mode, called non-broadcast multi-access or NBMA, simulates the operation of OSPF on a broadcast network. The second mode, called Point-to-MultiPoint, treats the non-broadcast network as a collection of point-to-point links. Non-broadcast networks are referred to as NBMA networks or Point-to-MultiPoint networks, depending on OSPF's mode of operation over the network.

5.7 The IP Subnet Model

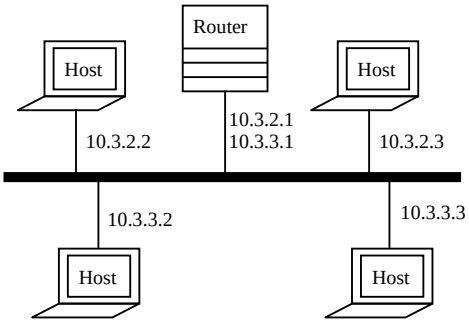
Before we can further explore OSPF and the different subnet technologies, we must first understand how OSPF uses the IP subnet model. In TCP/IP, every link (or physical subnet) is assigned one or more address prefixes. Each address prefix is called an *IP subnet*. For example, we can assign to an Ethernet subnet the set of addresses beginning with the first 24 bits of 10.3.2.0 (usually written as 10.3.2.0/24). One can also say that the subnet number of the Ethernet segment is 10.3.2.0 with a subnet mask of 255.255.255.0 (in hexadecimal 0xffffff00). This means that all routers or hosts connected to this Ethernet segment have an IP address in the range of 10.3.2.0 to 10.3.2.255.

IP routes to subnets, not to individual hosts. IP routing protocols advertise routers to address prefixes, and each entry in an IP routing table is an address prefix. The IP subnet model generally contains the following rules:

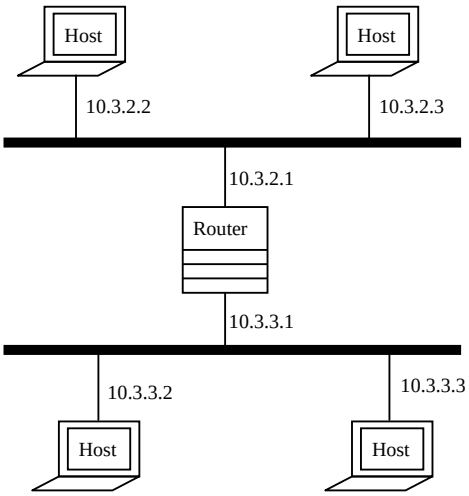
- Two hosts on different IP subnets cannot send IP packets directly to each other (or, as we say, cannot talk directly) but instead must go through one or more IP routers.
- In the converse of the preceding rule, it is assumed that two host/routers on a common subnet can send packet directly to each other.
- Two routers cannot exchange routing information directly unless they have one or more IP subnets in common. (This is only true for routers exchanging information via Interior Gateway Protocols. When running Exterior Gateway

Protocols, such as BGP, two routers generally need either a common IP subnet or to belong to the same Autonomous System.)

Multiple subnets (in other words, IP prefixes) can be assigned to a physical link. In this case, the IP subnetting rules say that two hosts on separate subnets cannot talk directly, even if they are on the same physical link. Figure 11 shows an example of an Ethernet segment that has been assigned two IP subnets: 10.3.2.0/24 and 10.3.3.0/24. Each subnet has two hosts, and a router has addresses on both. The physical connectivity shows all hosts and the router on a common link, and the IP connectivity shows two separate segments connected by an IP router (see figure 12).



- Figure 11: Ethernet segment with two IP subnets.



- Figure 12: Resulting IP connectivity.

However, there are some exceptions to these IP rules. In Ipv6, two nodes can talk directly whenever they are connected to the same physical media, regardless of their addresses. The Point-to-MultiPoint network model in the OSPF protocol also breaks the rule that two routers on the same subnet must be able to talk directly. This feature can be useful in wireless (radio) network environments were there is only one physical medium and no full connectivity (stations may be situated outside each other radio range).

5.8 Adjacencies

OSPF creates adjacencies between neighboring routers for the purpose of exchanging routing information. Not every two neighboring routers will become adjacent. Two situations are possible, depending on whether a Designated Router is elected for the network. On physical point-to-point networks, Point-to-MultiPoint networks and virtual links, neighboring routers become adjacent whenever they can communicate directly. In contrast, on broadcast and NBMA networks only the Designated Router and the Backup Designated Router become adjacent to all other routers attached to the network. This section covers the mechanisms involved in creating adjacencies.

5.8.1 The Hello protocol

Routers that share a common segment become neighbors on that segment. The Hello Protocol is responsible for establishing and maintaining neighbor relationships. It also ensures that communication between neighbors is bidirectional. Hello packets are sent periodically (by default every 10 seconds) out of all router interfaces. Bidirectional communication is indicated when the router sees itself listed in the neighbor's Hello Packet. On broadcast and NBMA networks, the Hello Protocol is also responsible for electing one router to become the designated router (DR) and one router to be a backup designated router (BDR). Using a Designated Router (DR) will reduce the amount of information exchange. The DR maintains a central link-state database, and all other routers keep their link-state databases synchronized with the DR, using the normal procedures of database exchange and reliable flooding.

In order to ensure two-way communication between adjacent routers, the Hello packet contains the list of all routers on the network from which Hello Packets have been seen recently. The Hello packet also contains the router's current choice for Designated Router and Backup Designated Router. The Hello Packet contains the router's Router Priority (used in choosing the Designated Router), and the interval between Hello Packets sent out the interface. The Hello Packet also indicates how often a neighbor must be heard from to remain active. Both these parameters must be the same for all routers attached to a common network. The Hello packet also contains the IP address mask of the attached network (Network Mask).

The Hello Protocol works differently on broadcast networks, NBMA networks and Point-to-MultiPoint networks. On broadcast networks, each router advertises itself by periodically multicasting Hello Packets. This allows neighbors to be discovered dynamically. These Hello Packets contain the router's view of the Designated Router's identity, and the list of routers whose Hello Packets have been seen recently.

On NBMA networks some configuration information may be necessary for the operation of the Hello Protocol. Each router that may potentially become the Designated Router has a list of all other routers attached to the network. A router, having Designated Router potential, sends Hello Packets to all other potential Designated Routers when its interface to the NBMA network first becomes operational. This is an attempt to find the Designated Router for the network. If the router itself is elected Designated Router, it begins sending Hello Packets to all other routers attached to the network.

On Point-to-MultiPoint networks, a router sends Hello Packets to all neighbors with which it can communicate directly. These neighbors may be discovered dynamically through a protocol such as Inverse ARP, or they may be configured. After a neighbor has been discovered, bidirectional communication ensured, and (if on a broadcast or NBMA network) a Designated Router elected, a decision is made regarding whether or not an adjacency should be formed with the neighbor. If an adjacency is to be

formed, the first step is to synchronize the neighbors' link-state databases. This is covered in the next section.

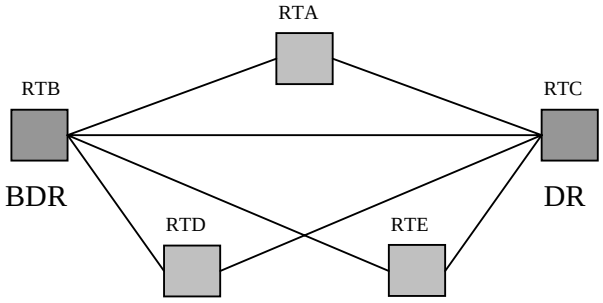
5.8.2 Database Synchronization

The next step after the neighboring process is the forming of adjacencies. Adjacent routers are routers who go beyond the simple Hello exchange and proceed into the database exchange process. When in the subnet more than two routers are used, OSPF makes the distinction between broadcast multi-access networks (like Ethernet, Token Ring and FDDI) and non-broadcast multi-access (NBMA) networks (like Frame Relay, X.25 and ATM). If you try to synchronize databases between every pair of routers, you end up with a large number of Link State Updates and Acknowledgments being send over the subnet. Therefore to minimize the amount of information exchange on a particular segment, OSPF elects one router to be a designated router (DR), and one router to be a backup designated router (BDR), on each multi-access segment. The BDR is elected as a backup mechanism in case the DR goes down. The idea behind this is that routers have a central point of contact for information exchange. Instead of each router exchanging updates with every other router on the segment, every router exchanges information with the DR and BDR. The DR and BDR will relay the information to everybody else. Because the DR is the central point of contact for all the routers in the network is it necessary that the DR has a direct link with each router. Because in non-broadcast networks this is not always the case other solutions have to be used.

Broadcast networks

A broadcast subnet is a data link whereby an attached node can send a single packet that will be received by all other nodes attached to the subnet. The best example is Ethernet. When a station on an Ethernet sends an Ethernet packet, all other stations hear the packet, but their Ethernet adapters discard the packet unless it is addressed to ether the adapter's own unique 48-bit Ethernet MAC address or to the Ethernet broadcast address. An additional useful capability of some broadcast networks is *multicast*. Multicast is the ability of a node to send onto the subnet a single packet that will be accepted by some subset of nodes on the subnet. Some network technologies, such as Ethernet and FDDI, offer very good multicast capabilities. Other broadcast subnets offer no multicast capabilities, or their multicast is of such limited scope (such as Token Ring and SMDS) as to be of no use to OSPF.

As mentioned in the previous section each router will periodically multicast Hello packets. All routers in the subnet will receive the Hello packets and use it to maintain its relationships with the other OSPF routers. The utilization of multicast improves the efficiency because each time OSPF routers only have to send one Hello packet instead of sending a separate Hello packet to each neighbor on the Ethernet.



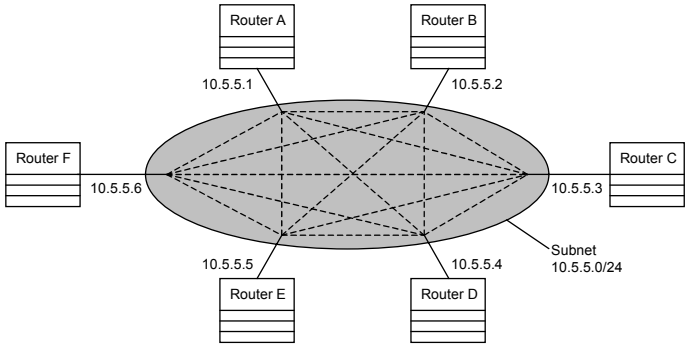
- Figure 13: The reliable flooding process

To synchronize the database broadcast OSPF uses the DR process. It uses reliable flooding to send the LSAs. For example suppose that, in figure 13, Router E (RTE) receives a new LSA from one of its other links. The router then installs the LSA in its link-state database and then wants to flood it to both RTC and RTB (the Designated Router (DR) and Backup Designated Router (BDR), respectively). To do so, RTE multicasts a Link State Update to the DR and BDR. When the DR and BDR receives this LSA, the DR will send the LSA back into the Ethernet segment in a Link State Update to all the other routers, hereby updating them. The responsibility for the flooding mechanism lies at first by the DR. However, if the BDR does not see the Link State Update from the DR within the LSA retransmission interval (typically 5 seconds), it will step in and flood the LSA back onto the Ethernet subnet in order to keep the database synchronization going.

The Designated Router election process works as follows, using data transmitted in Hello packets. The first OSPF router on an IP subnet always becomes the Designated Router. When a second router is added, it becomes the Backup Designated Router. Additional routers added to the subnet segment defer to the existing Designated Router and Backup Designated Router. The only time the identity of the Designated Router or Backup Designated Router changes is when the existing Designated Router or Backup Designated Router fails. In the event of the failure of a Designated Router or Backup Designated Router, there is and orderly changeover. Each OSPF router can be configured with a Router Priority value (a value between 0 and 127). In the event of a failure the Router with the highest Router Priority value will become the next Designated Router or Backup Designated Router. Routers with a Router Priority value of 0 do not participate in this process.

NBMA networks

Non-broadcast multi-access networks can support more than two routers and allow any two routers to communicate directly but they do not support any broadcast capabilities. The OSPF protocol distinguishes two types of models to configure non-broadcast network. The NBMA subset utilizes a Designated Router and Backup Designated Router and the Point-to-MultiPoint subnet does not. NBMA segments are efficient in terms of neighbor maintenance, database synchronization, and database representation. The mechanisms used are similar to those used for broadcast subnets. However, NBMA segments have weird failure modes when two attached routers cannot communicate directly, hereby making the Point-to-MultiPoint model in some situations the more robust, although less efficient, choice. An example of an NBMA segment is shown in Figure 14. A collection of six routers is attached to a single subnet, with each router connected to every other router. A single IP subnet will be assigned to the NBMA segment, with all routers having IP interface address on the segment.

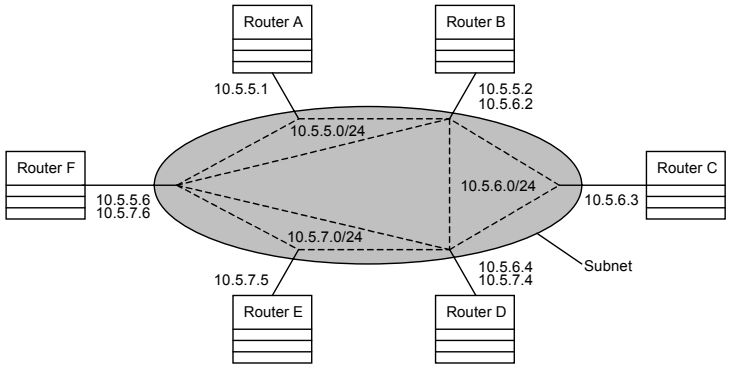


- Figure 14: Non-broadcast subnet with a full mesh topology

Because there is no broadcast capability on an NBMA subnet, static configuration information may be necessary in order for the Hello Protocol to function. On NBMA networks, every attached router that is eligible to become Designated Router (those routers with a nonzero Router Priority value) becomes aware of all of its neighbors on the network (either through configuration or by some unspecified mechanism). To minimize the number of Hello Packets sent, the number of eligible routers on an NBMA network should be kept small. In routers eligible to become Designated Router, the identity of all routers attached to the NBMA subnet must be configured, as whether those routers themselves are eligible to become Designated Router.

A router's hello-sending behavior varies depending on whether the router itself is eligible to become Designated Router. If the router is eligible to become Designated Router, it must periodically (typically every 10 seconds) send Hello Packets to all neighbors that are also eligible. In addition, if the router is itself the Designated Router or Backup Designated Router, it must also send periodic Hello Packets to all other neighbors. This means that any two eligible routers are always exchanging Hello Packets, which is necessary for the correct operation of the Designated Router election algorithm. If the router is not eligible to become Designated Router, it must periodically send Hello Packets to both the Designated Router and the Backup Designated Router (if they exist). It must also send a Hello Packet in reply to a Hello Packet received from any eligible neighbor (other than the current Designated Router and Backup Designated Router). This is needed to establish an initial bidirectional relationship with any potential Designated Router. When sending Hello packets periodically to any neighbor, the interval between Hello Packets is determined by the neighbor's state. If the neighbor is in state Down and does not respond, Hello Packets are sent every “PollInterval seconds” (typically 120). Otherwise, Hello Packets are sent every “HelloInterval” seconds (typically 10).

Database synchronization on NBMA networks works the same as on broadcast networks. A Designated Router and Backup Designated Router are elected, all other router initially perform Database Exchange with the Designated Router and Backup Designated Router, and flooding over the NBMA subnet always goes through the Designated Router on the way to the other routers attached to the NBMA subnet. The only difference is that, where on broadcast subnets the Link State Updates are multicasted, on NBMA subnets, Link State Update must be replicated and sent to each adjacent router separately. Many of the non-broadcast subnet cannot support a large number of routers, with each router being able to communicate directly. For example, to connect 100 routers in a full mesh (like in Figure 14) over a non-broadcast subnet would require 4,950 separate links. This limits the practical use of the NBMA model. However, a partial mesh can be turned into multiple NBMA networks, although then the configuration can get quite complicated (see Figure 15).

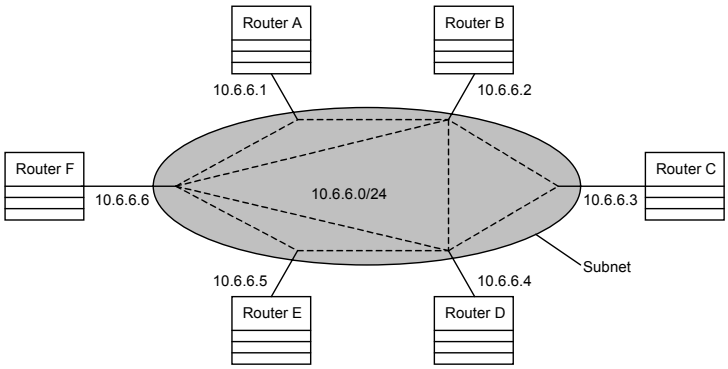


- Figure 15: Non-broadcast subnet partial mesh implemented as multiple NBMA subnets.

For example, in Figure 15 not every pair of routers is interconnected, but there can be three overlapping router subnets found, each of which is full-mesh connected: {F,A,B}, {B,C,D}, and {D,E,F}. Each of these subnets is assigned its own IP subnet. Those routers attached to more than one subnet then end up with multiple IP addresses and OSPF interfaces, each attaching to a different subnet. This method requires much and complicated configurations, which makes NBMA networks vulnerable to faults and difficult to maintain. To overcome these problems the Point-to-MultiPoint model can always be applied.

Point-to-MultiPoint networks

The Point-to-MultiPoint model can be used on any data-link technology that the NBMA model can be used on. Usually these subnets are connection-oriented subnets, such as Frame Relay and ATM. The Point-to-MultiPoint model drops the requirement that all routers on the subnet must be able to communicate directly, making it possible to model partial meshes as single Point-to-MultiPoint networks. Dropping the full mesh requirement also allows the modeling of more exotic data-link technologies, such as packet radio and wireless networks, as Point-to-MultiPoint networks.



- Figure 16: Non-broadcast network with partial mesh, implemented as a Point-to-MultiPoint subnet.

For example, the partial mesh in Figure 15 can be turned into the single OSPF Point-to-MultiPoint subnet with prefix 10.6.6.0/24 as pictured in Figure 16. Each OSPF router would have a single IP address on the subnet and a single OSPF interface to the subnet, although multiple OSPF neighbor relationships would form over that interface: Router A would form neighbor relationship with F and B; B with A, C D, and F; and so on. In principle we would say that a Point-to-MultiPoint subnet is a collection of several Point-to-Point connections. The advantage of using the Point-to-MultiPoint model is in the possibility of autoconfiguration and the model's robustness. Since the Point-to-MultiPoint model deal well with partial connectivity between attached routers, a link- failure will cause no problem.

There are no Designated Routers or Backup Designated Routers on Point-to-MultiPoint subnets. On these subnets, the job of the Hello protocol is simply to detect active OSPF neighbors and to detect when communication between neighbors is bidirectional. A router on a Point-to-MultiPoint subnet periodically sends OSPF hellos to all other router on the subnet with which the router can converse directly. On a Point-to-MultiPoint subnet, each router becomes adjacent to all other routers with which it can communicate directly, performing initial database synchronization though Database Exchange, then participating in reliable flooding with its neighbors. For example, in Figure 16, each link is an OSPF adjacency. In order to flood an LSA from router A to router D, the LSA first goes to router B (or one of several alternative paths).

A router on a Point-to-MultiPoint subnet includes the following links in its router-LSA: a point-to-point connection for each of its neighbors on the Point-to-MultiPoint subnet and a single stub (static) network connection to its own IP interface address. For example, router D in Figure 16 would include four point-to-point in its router-LSA (one each to router B, C, E, and F) and a single stub network connection for its address of 10.6.6.4. Note, then, that to route from 10.6.6.4 to 10.6.6.1, the OSPF routing calculation will calculate a next hop of 10.6.6.2. Even to go between two routers on the same IP subnet, an intermediate router must be traversed. What the Point-to-MultiPoint model gains in autoconfiguration and robustness, it loses in efficiency. The closer the underlying physical subnet comes to providing full-mesh connectivity, the less efficient Point-to-MultiPoint becomes. However, when implementing the OSPF protocol in networks like packet radio or wireless LAN, due to the line-of-sight problems the Point-to-MultiPoint model will be the best and sometimes the only option. One main feature of these types of networks is the limited bandwidth available. To improve the efficiency on low-bandwidth networks the OSPF protocol offers some extra features with the OSPF Demand Circuit extensions [16]. When enabled, the OSPF Demand Circuit extensions will reduce the routing protocol traffic to a minimum, hereby making more of the link bandwidth available for user-data traffic.

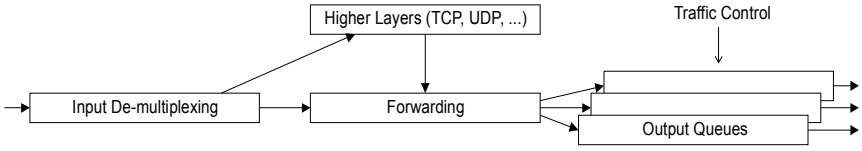
6 Prototype implementation

This section describes the prototype implementation of the Power Awareness Routing protocol that is studied in this thesis. This prototype implementation uses the Linux implementation environment, and it enhances an existing Zebra OSPF implementation environment. The prototype will be run inside multiple virtual Linux machines (created with the User-Mode Linux capability).

6.1 The Linux Implementation Environment

Linux is an open source operating system [22], which has a complete UNIX-like operating system with support for a number of advanced networking features. Linux supports good developing tools, like compilers, editors, debugger and more. Furthermore the ZEBRA implementation environment used to implement the OSPF protocol is supported by Linux. For these reasons, Linux is a good candidate for the prototype implementation of the Power Awareness Routing protocol.

Figure 17 briefly shows the network design in Linux. In particular, this figure shows how the kernel processes incoming packets and how packets from higher layers are processed. The packets arrive at the input de-multiplexer, which examines them to determine if the packets are destined for the local host. If that is the case, they are sent to the higher layers, or if not, they are sent to the forwarding block. Here routing and forwarding plays a crucial role. The forwarding functionality selects the correct outgoing interface for each packet using a routing table. Afterwards, the packet is enqueued on a queue associated with the interface before transmitted on the physical line for the interface.

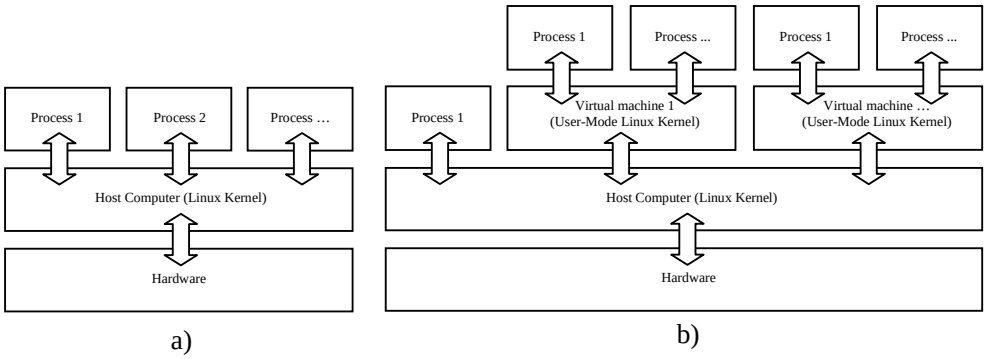


- Figure 17: Overview of networking in Linux

6.2 User-Mode Linux

User-Mode Linux (UML) [23] is an open source solution that can be used to create multiple virtual machines on a single computer. User-Mode Linux allows to run multiple instances of Linux on the same system at the same time. User-Mode Linux is currently limited to Linux. User-Mode Linux is designed for a variety of purposes, such as kernel debugging, testing applications, virtual networking, etc. Disk storage for the virtual machine is entirely contained inside a single file on the physical machine. Access to hardware can be assigned in each virtual machine. With properly limited access, nothing what is done on the virtual machine can change or damage the real computer, or its software.

Normally, the Linux Kernel communicates directly with the hardware (video card, keyboard, hard drives, etc), and any program process what is running asks the kernel to operate the hardware, see figure 18a. In a User-Mode Linux kernel (i.e. a virtual machine) this works different; instead of communicating directly with the hardware, it communicates with the main Linux kernel on the host computer, like any other program. Programs can then run inside User-Mode Linux as if they were running under a normal kernel, see figure 18b.



- Figure 18: a) Normal Linux Kernel operation;
  b) Virtual machine (with User-Mode Linux Kernel) operation.

In the virtual Linux machine (i.e. the User-mode Linux Kernel) all the devices are virtual, being constructed from software resources provided by the host. These include every type of device that would be expected to be present on a typical physical machine:

- Consoles and serial lines may be attached to a variety of devices on the host, including pseudo-terminals, virtual consoles, file descriptors, and xterms.

- Block devices can be associated with anything on the host that resembles a normal file, including files and device nodes such as CD-ROMs, floppies, disk partitions, and whole disks. These normally contain a filesystem image or a swap signature and are mounted or used as swap by the virtual machine.

- Network devices can be attached to most types of software network interfaces on the host, such as TUN/TAP, Ethertap, and slip devices. There are also two mechanisms for exchanging Ethernet frames directly between virtual machines without going through the host's networking system - one involving a central daemon acting as an Ethernet switch and the other using a multicast network. This daemon and the multicast network provide a completely virtual network to other virtual machines. This virtual network is completely disconnected from the physical network unless one of the virtual machines on it is acting as a gateway.

6.3 GNU ZEBRA

6.3.1 About Zebra

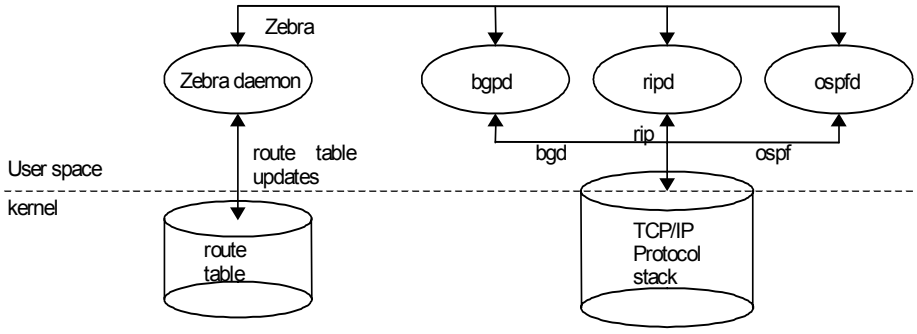
Zebra [24] is a free routing software package that provides TCP/IP based routing services with routing protocols support such as such as RIPv1, RIPv2, RIPng, OSPFv2, OSPFv3, BGP-4, and BGP-4+. In addition to traditional IPv4 routing protocols, Zebra also supports IPv6 routing protocols. Zebra uses an advanced software architecture to provide a high quality, multi-server routing engine. The GNU Zebra uses a process for each supported IP routing protocol. Zebra has an interactive user interface for each supported routing protocol and supports common client commands.

Zebra is an official gnu software and distributed under the gnu General Public License [25]. The Zebra project started out as the brainchild of Kunihiro Ishiguro and was intended for consumption by the open source software community. Zebra however grew to be popular and now also a commercial version of Zebra is supported which is

distributed by IP Infusion Incorporation under the name of “ZebOS” [26]. At the moment Zebra is still under development but nearing completion and the release of version 1.0 is expected in the near future. The latest release is version 0.94, and this version is used for the implementation of the power awareness routing scheme.

Zebra runs on several platforms, namely GNU/Linux 2.2.X and 2.4.X, FreeBSD 4.X and 5.X, NetBSD 1.6.X, OpenBSD 3.X and SUN Solaris. IPv6 support is for FreeBSD, NetBSD, OpenBSD and GNU/Linux. Within the Zebra environment, a multi-home computer (a host with multiple interfaces) can easily be configured as a router that runs multiple routing protocols .

The Zebra system architecture can be seen Figure 19. The different routing protocol machineries are implemented in protocol-specific routing daemons. The *ripd* daemon handles the RIP protocol, see Table 10, while *ospfd* is a daemon which supports the OSPF version 2. The *bgpd* supports the BGP-4 protocol. The Zebra kernel routing manager is used to manage the protocol routing specific daemons and the routing table.



- Figure 19: Zebra System Architecture

Each routing protocol daemon uses the basic socket API to receive and transmit protocol specific PDUs (e.g. OSPF PDUs).

This multi-process architecture brings extensibility, modularity and maintainability. At the same time it also brings many configuration files and terminal interfaces. Each daemon has its own configuration file and terminal interface. Before starting the daemons, possible network interfaces and static routes must be configured in the zebra configuration file. For instance the configuration of an OSPF network must be done in the *ospfd* configuration file.

Daemon name	Protocol function
Bgpd	Manages BGP-4 and BGP-4+ protocol
Ripd	Manages RIPv1, v2 protocol
Ripngd	Manages RIPng protocol
Ospfd	Manages OSPFv2 protocol
Ospf6d	Manages OSPFv3 protocol
Zebra	For Kernel routing table update and routing information redistribution between above protocols

- Table 10: List of protocol daemons

6.3.2 OSPF metric cost assignment and LSA implementation

An overview of the OSPF protocol is given in chapter 5. From all features that are supported by OSPF, the feature of metric cost assignment and the Link State Advertisements (LSA) feature are associated with the power awareness routing implementation. Therefore, we will only focus on these two OSPF features.

With the power awareness routing implementation it is needed to assign a power parameter as metric cost for the interface state. Based on these power aware states, the routing tables are build on each router.

The OSPF LSA implementation uses an update mechanism to maintain the routing tables by the Link State Update and Link State Acknowledgment packets, see section 5.2. Each Link State Update packet carries a set of new link state advertisements (LSAs) which describe the state of the interfaces. A single Link State Update packet may contain the LSAs of several routers.

In RFC 2328 [5] an OSPF router periodically advertises its link state (in the Zebra implementation, the default is every half hours). Link states are also advertised when a router's state changes. A router's adjacencies are reflected in the contents of its LSAs. This relationship between adjacencies and link state allows the protocol to detect dead routers in a timely fashion. LSAs are flooded throughout the area. The flooding algorithm is reliable, ensuring that all routers in an area have exactly the same link-state database. This database consists of the collection of LSAs originated by each router belonging to the area. From this database each router calculates a shortest-path tree, with itself as root. This shortest-path tree in turn yields a routing table for the protocol.

As described in section 5.2 and Table 8, the OSPF protocol supports different types of LSA's, e.g., Router link, Network link, Summary link to network, Summary link to Autonomous System (AS) boundary router and External link.

For the power aware routing implementation described in this thesis, only the Router link LSA (Router-LSA) type is used. Therefore, this section describes only the Zebra functions used for the implementation of the Router-LSA.

6.3.3 Zebra OSPF daemon implementation

The OSPF daemon of Zebra is called *ospfd* . Zebra is written in C and its source code can be found in the subdirectory: *./ospfd* of Zebra's source directory. Table 11 gives a short description for each source file of *ospfd*.

Zebra Threads

After the daemons are started and have performed their initial initializations like installation of command nodes and command elements, they are able to execute the functions using a thread mechanism. The types of threads used in Zebra are different than the normal operating system supported threads. Zebra defines a thread as a structure and performs its own scheduling of threads. There is one master thread that maintains the list of threads to be executed. There are 3 possibilities for a thread to be scheduled:

1) timer expiration;
2) I/O event (read or write, not both at once);
3) as an event (to decouple threads).

ospf_abr.c	ABR related functions
ospf_asbr.c	some ASBR functions, database for external information
ospf_ase.c	Algorithm for AS-external route calculation
ospf_dump.c	log file output
ospf_flood.c	LSA flooding
ospf_ia.c	inter-area routing
ospf_interface.c	Interface related functions
ospf_ism.c	Interface state machine
ospf_lsa.c	most LSA related functions: look up, generate, originate, refresh, age, and flush router, network, AS-external, and summary LSAs
ospf_lsdb.c	link-state database
ospf_main.c	daemon main routine, initialization
ospf_neighbor.c	neighbor related data, link-state retransmission and request lists
ospf_network.c	socket calls to join and quit multicast groups
ospf_nsm.c	neighbor state machine
ospf_packet.c	Receiving, queuing, and sending OSPF packets, various timers
ospf_route.c	routing table
ospf_routemap.c	Functions for exchange of routing information (route map) with other Zebra components
ospf_snmp.c	SNMP support
ospf_spf.c	shortest path algorithm
ospf_zebra.c	interface (API) between OSPFD and the Zebra main daemon
ospfd.c	several functions concerning areas, virtual links, vty commands

- Table 11: OSPFD source files

The implementation of the Router-LSAs in ospfd

Router-LSAs are originated by OSPF routers for each area they belong to. Router-LSAs are flooded throughout a single area only. They contain information about the router and its interfaces, including the connected subnets.

The Zebra OSPF functions, and their interaction, used to implement the Router-LSA feature are shown in Figure 20. We can separate the different functions used in the Router-LSA implementation into different groups, namely:

- *Updating router status*: create a new router-LSA when the status of a router is changed, this can be an internal router, an area border router (ABR) or an autonomous system border routers (ASBR);

- *Updating link status*: detect and notify the situation when the status or metric costs of a router's interface is changed;

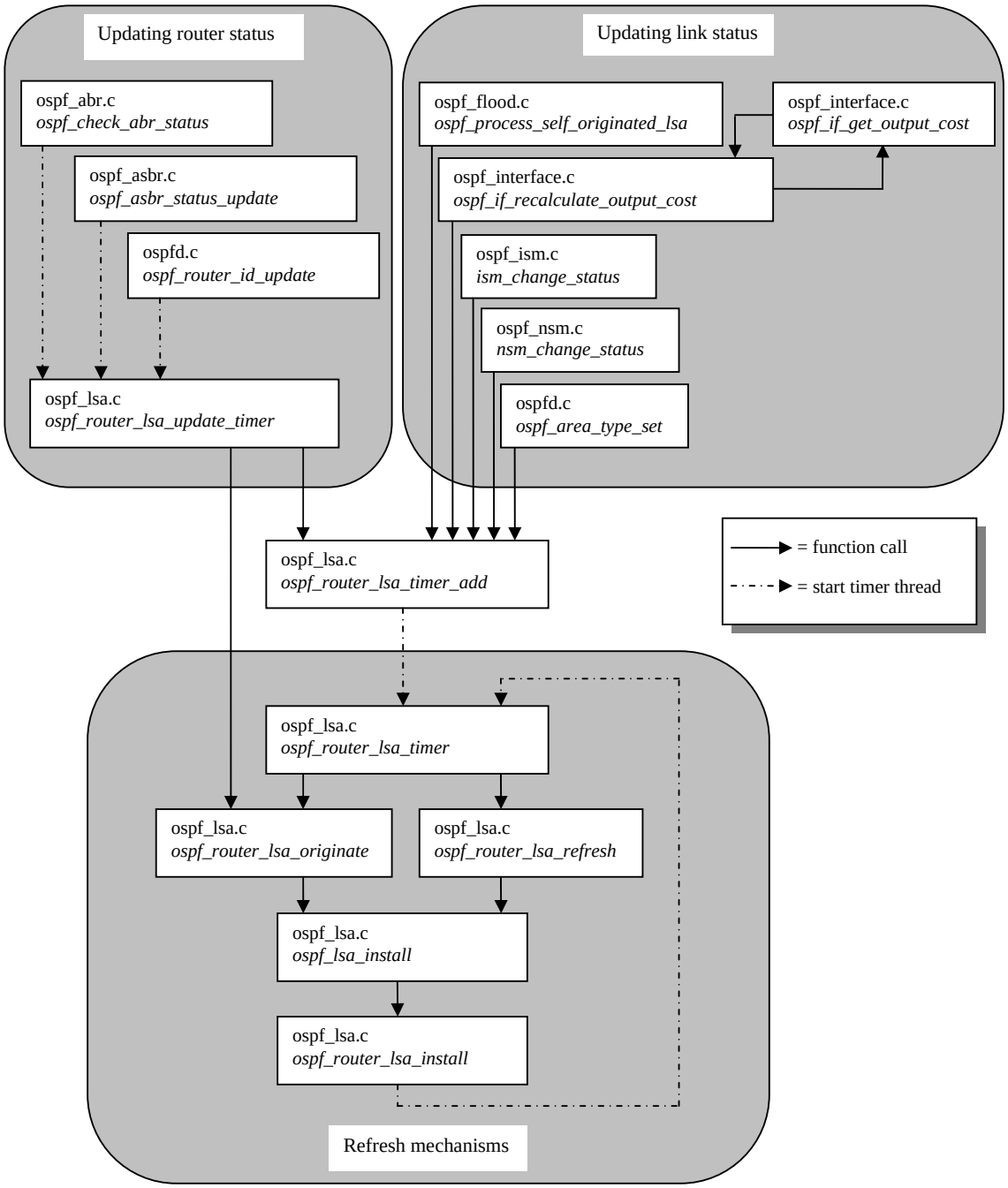
- *Refresh mechanisms*: Implements the LSA refreshment mechanism. Link states are soft states, i.e. they have to be refreshed periodically. When the duration of a LSA reaches the value MaxAge without being refreshed, the LSA is removed from the link-state database and flushed from the routing domain. It is evident that a LSA has to be refreshed by its originating router, which, before this, it has to check if the information is still valid.

The following functions can be grouped in the *Updating router status* part:

- ospf_abr.c*: ospf_check_abr_status()*: Checks and updates the status of area border routers;

- ospf_asbr.c: *ospf_asbr_status_update()*: Checks and updates the status of autonomous system border routers (ASBRs);

- ospfd.c: *ospf_router_id_update()*: Checks and updates the router status;
- ospf_lsa.c: *ospf_router_lsa_update_timer()*: used when router related information has changed.



- Figure 20: Origination and refreshing of router-LSAs (figure based on Figure 1 from [27])

The following functions can be grouped in the *Updating link status* part:

- ospf_flood.c:*ospf_process_self_originated_lsa():* used when a self-originated router-LSA has been received;
- ospf_interface.c:*ospf_if_recalculate_output_cost():* used when the cost associated with a router's interface has changed;
- ospf_interface.c:*ospf_if_get_output_cost():* used when the cost associated with a router's interface has changed, then this function gets the new value of the metric cost;
- ospf_ism.c:*ism_change_status()*: used when the status of the interface state machine has changed;
- ospf_nsm.c:*nsm_change_status()*: used when the status of the neighbor state machine has changed;
- ospfd.c:*ospf_area_type_set()*: used when the area type is set (default, stub, NSSA).

The following functions can be grouped in the *Refresh mechanisms* part:

- ospf_lsa.c:*ospf_router_lsa_timer()*: used to trigger the *ospf_router_lsa _originate();*
- ospf_lsa.c:*ospf_router_lsa_originate()*: used to originate new router-LSAs;
- ospf_lsa.c:*ospf_router_lsa_refresh()*: used to originate the refresh of router-LSAs;
- ospf_lsa.c:*ospf_lsa_install()*: used to trigger the *ospf_router_lsa_install()*; function;
- ospf_lsa.c:*ospf_router_lsa_install()*: used after having originated or refreshed the LSA (ensures periodical refreshing).

The function ospf_lsa.c*:ospf_router_lsa_originate()* originates new router-LSAs, it is triggered by the timer thread ospf_lsa.c*:ospf_router_lsa_timer()*.This timer is activated by the function ospf_lsa.c:*ospf_router_lsa_timer_add()*, which first, if needed, it stops an already running timer. This function is only called if router-LSA related information has changed.

6.4 Power Awareness Routing Implementation

The power awareness routing implementation enhances the Zebra OSPF routing software by introducing a power aware metric cost assignment and by adapting the LSA process to support power awareness.

6.4.1 Setting metric costs in Zebra

Usually, in the Zebra software environment, the metric cost of a OSPF interface link is set manually. This can be accomplished in two ways. The first method sets the metric cost indirectly, where first the bandwidth value for the interface is set and subsequently the OSPF metric cost (see chapter 5), is calculated. The bandwidth value for the interface in the zebra daemon configuration is set using the following commands:

Interface bandwidth <1-10000000>;
Interface Command no bandwidth <1-10000000>

The second method sets the metric cost directly. The following command can be used for this purpose:

ip ospf cost <1-65535> Interface Command
no ip ospf cost Interface Command

This metric cost value is set on the router-LSA's metric field and it is used for the shortest path three calculation. If no bandwidth or OSPF metric cost are specified then a default metric cost of 10 will be used for all interfaces.

6.4.2 Power awareness routing metric

In power awareness routing, different parameters can be used for the link costs, depending on the desired routing scheme. Power aware metrics can be used to minimize the variance in each computer power level, to minimize the ratio between cost/packet, and to maximize the battery lifetime of a wireless node.

In this thesis, we focus on the residual energy (power level) of the wireless nodes in a system. To maximize the lifetime of the wireless network (defined as the time to the earliest time a message cannot be sent) we want to avoid the use of wireless nodes with small residual energy, since we would like to maximize the minimum lifetime of all wireless nodes. This scheme will result in, that nodes with high residual energy capacity participate in the routing process more often than the nodes with low residual capacity. Translated to metric costs, this will mean, that when a wireless node has plenty of energy, the metric costs must be set to a low value. When a wireless node has a small residual energy level the metric costs must be set to a high value, making this wireless node less favorable for routing packets.

The residual energy of a wireless node can be defined in percents, with a fully charged battery represented by a power level of 100%. A completely discharged battery, can be represented by a power level of 0%. There are different ways on selecting the corresponding metric costs, provided that the chosen metric value lies in the interval: $1 \leq \text{cost} \leq 65535$. For simplicity, in this prototype implementation, the minimum metric cost is defined as being equal to 1, with a power level at 100%. The maximum metric cost is then defined as being equal to 100, with a power level at 1% (at 0% there is no power left for routing). Note that in this situation, every 1% drop in the power level corresponds to an increase of a value equal to 1 in the metric costs.

6.4.3 Determining the battery status

To determine the battery (or power) status of a wireless node, a method is required to collect the information related to the battery status of the wireless node. Today most laptop computers are equipped with an Advanced Power Management control program (*APM*) BIOS. In Linux, the *APM* can be used to communicate with the Advanced Power Management daemon (*APMD*), on collecting the power status of the wireless node, or to place the system into a *suspend* or *stand-by state*. Depending on the used operating system and on the used flags, the *APM* can provide the battery status information either in percents of remaining power level or in the estimated battery lifetime. In appendix B a possible method of using *APM* in Zebra to update the metric cost on the battery status is given. This is provided by modifying the function ospf_interface.c:*ospf_if_get_output_cost()*, see section 6.3.3 and Figure 20.

However, in this thesis, a test bed is used, where a wired desktop computer (see chapter 7) runs the Zebra routing software. This desktop computer is not supporting the *APM* BIOS. Therefore, in the test bed, we cannot use *APM* for collection of the battery status information. To emulate the functionality of *APM* and the lifetime cycle of a battery an another method is needed.

First, by using real battery lifetime curves, a file is created with a list of timestamps (from 0 to max_timestamp), and with their corresponding power level (in %). A special program has been written named "*power*" (see chapter 7 and appendix C for the

source code), to create this file according to a predetermined model for the lifetime cycle of a battery, such as the one given in [28]. This file is named "*powerdata.txt*" and must be located in the configuration directory of Zebra (default:./usr/local/etc/).

Next, to emulate *APM* in the Zebra software environment, every time that the OSPF process is started the *start* timestamp is determined in the initialization process of the power awareness routing, i.e. in ospf_interface.c:*power_init().* After this, every time a function is called to read the power status of the battery the *current* timestamp is determined. Then the difference between the *start* timestamp and the *current* timestamp is calculated. This resulting time difference is looked up in the file "*powerdata.txt*". and the found corresponding power entry becomes the new power level.

6.4.4 Power awareness routing ospfd implementation

The power aware enhancements of *ospfd* can be seen in Figure 21. The functions that are modified or are new have a thicker outline. The source code of these functions can be found in appendix D.

To keep the routing tables of all OSPF routers up-to-date, every time the metric cost is changed a router-LSA is created and flooded to all the routers in the same area. The function ospf_interface.c:*ospf_if_recalculate_output_cost()* is used to redefine the metric cost and to create a router-LSA.

Two schemes are implemented to trigger an update of the metric costs. This is configured in the *power.conf* file (created by the program "*power*"), located in the zebra configuration directory (./usr/local/etc).

The first scheme (called "*Time interval*") is based on the time function ospf_interface.c:*time_interval_mode()*. A timer thread is set with a predefined time period. When the timer expires the function ospf_interface.c: *ospf_if_recalculate_output_cost()* is activated to recalculate the new metric cost. Then a new timer thread to call ospf_interface.c:*time_interval_mode()* is set to repeat the same process. In this manner the metric cost can be updated at certain time intervals.

In the second scheme (called "*Power interval*"), the metric cost and the creation of router-LSA is based on the power level of the battery (in function ospf_interface.c:*power_interval_mode()*). A timer thread is set up to check the power level at certain time periods. If the power level has reached a certain predefined level the function ospf_interface.c: *ospf_if_recalculate_output_cost()* is activated and a Link State Update is created. If the power level reached 0% or the maximal timestamp is reached the ospfd program will be exited and the routing process will be stopped.

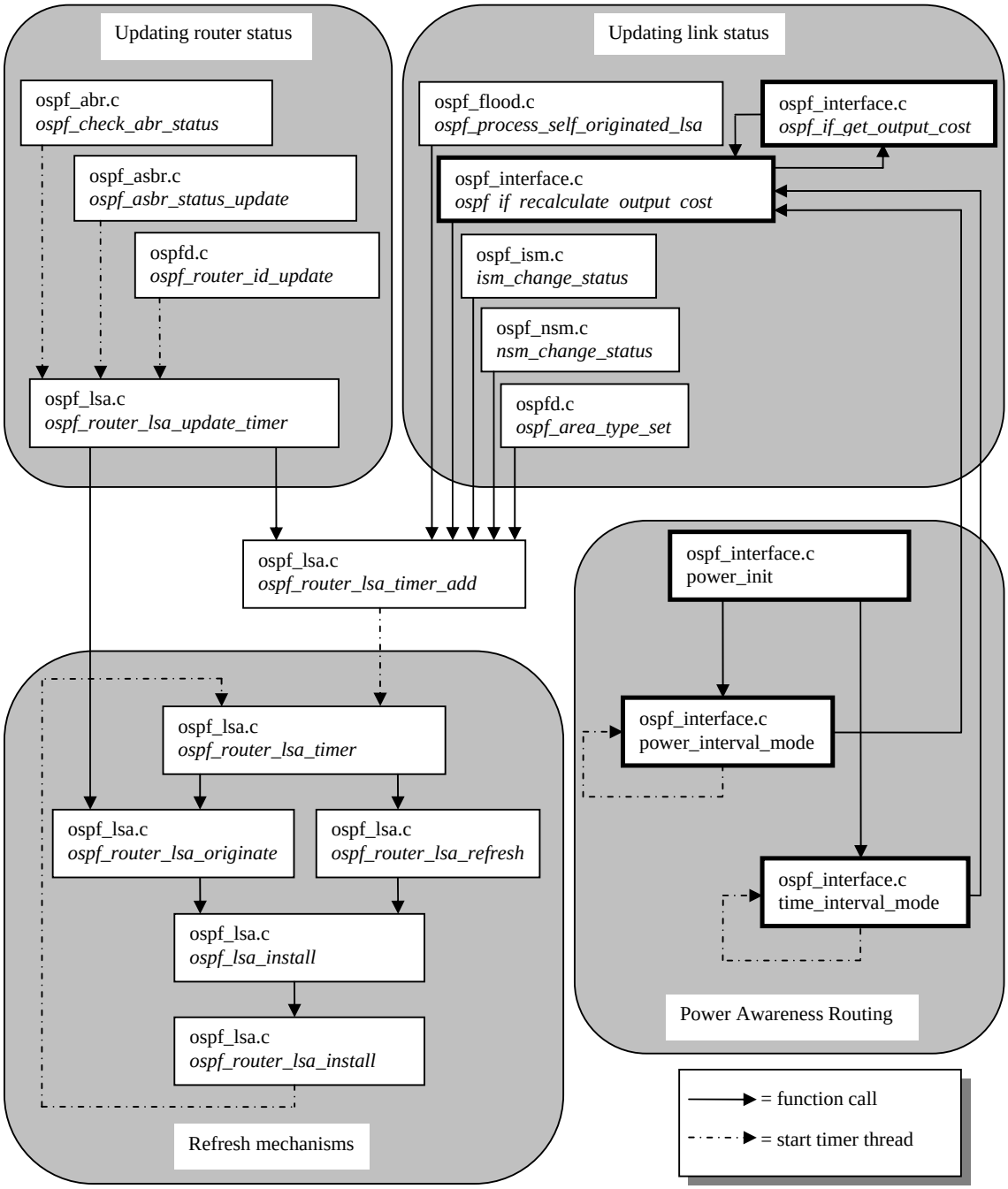
The modified functions are:

- ospf_interface.c:*ospf_if_recalculate_output_cost():* used to recalculate the metric cost (by calling the function ospf_interface.c:*ospf_if_get_output_cost()*) and creates a new router-LSA;
- ospf_interface.c:*ospf_if_get_output_cost():* used to determine the new metric costs, called by ospf_interface.c: *ospf_if_recalculate_output_cost().*

The following functions are completely new:

- ospf_interface.c:*power_init()*: used to initializate the power awareness routing, e.g. determine the used Link State Update scheme (out of the *power.conf* file), sets up the link cost, and timer interval, and start the first power awareness timer thread;

- ospf_interface.c:*time_interval_mode()*: used to call ospf_interface.c: *ospf_if_recalculate_output_cost(...)* to update the (power awareness) metric cost of the router's interfaces and schedules a next timer thread to ospf_interface.c: *time_interval_mode()*.
- ospf_interface.c:*power_interval_mode()*: when this function is called, first the power level will be checked. Depending on the used Link State Update scheme, at certain predefined power levels the metric cost of the router's interfaces will be updated by calling ospf_interface.c:*ospf_if_recalculate_output_cost(...)*. Finally a new timer thread to ospf_interface.c:*power_interval_mode()* will be scheduled.



- Figure 21: Origination and refreshing of router-LSAs using power awareness

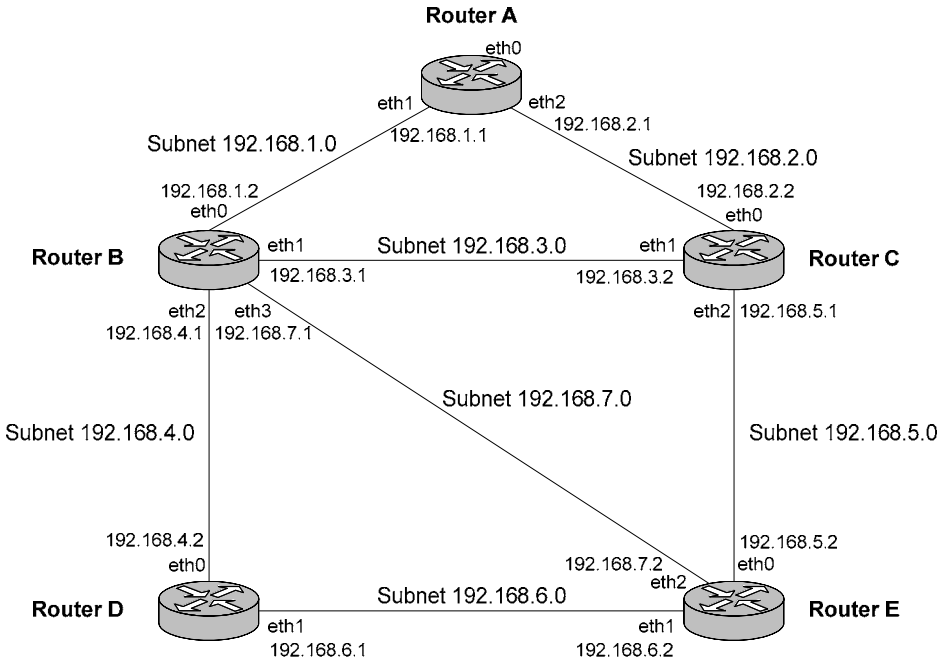
7 The experiments

The previous chapter described the prototype implementation of the power awareness routing scheme. To perform experiments on the implemented power awareness routing prototype a test bed is required. The specifications discussed in chapter 4 should, if possible, be incorporated into the test bed. Hereby making the test bed comparable to a multihop ad hoc network. In this chapter the design and configuration of the test bed will be discussed. Furthermore, the design choices and assumptions will be motivated and finally the various experiments will be described.

7.1 Test bed configuration

To test and accomplish experiments on the implemented power awareness routing prototype, a test bed is required. A wireless ad hoc network composed of several wireless nodes (laptop computers) would be ideal for the test bed. However, there were not enough laptop computers available for the test bed. Therefore another solution had to be found. The solution was found in *User-Mode Linux* (UML) (see section 6.2). UML is an open source solution that can be used to run multiple Linux kernels (virtual machines) at the same time on a single Linux host computer. The disk storage for the virtual machine is entirely contained inside a single file on the host computer.

In each virtual machine virtual network devices can be configured. By using a central daemon acting as an ethernet switch, ethernet frames can directly be exchanged between virtual machines without going through the host's networking system. This provides a completely virtual network to other virtual machines. This network is completely disconnected from the physical network, unless one of the virtual machines is acting as a gateway. The virtual network behaves similar as a real network and therefore the test bed configuration can easily be implemented into a real network without affecting its operation.



- Figure 22: Network configuration

The network configuration which is used in the experiments is shown in Figure 22. The host computer used in two experiments was a Pentium 2, 500 MHz with 256 Mb memory (see sections 7.5.2 and 7.5.3), and in one experiment a Pentium 4, 2,8 Ghz with 512 Mb memory computer was used as host (see section 7.5.1). The operating system on the host computers was Mandrake Linux 9.2. On the host computer are five virtual machines created. The virtual machines are running a Mandrake Linux 8.2 kernel. The file system of this kernel (*root_fs.md-8.2-full.pristine.20020324.bz2*) can be downloaded from the UML homepage [23] and is well tested and relative bug-free, and can easily be adapted for our purposes. The number of virtual machines (i.e. nodes) that can be created is limited by the available internal memory of the host machine. Each virtual machine need at least 32 Mb memory. Five virtual machines are enough for the different experiments, and can still run on the Pentium 2 machine with only 256 Mb internal memory.

In each virtual machine are several virtual interfaces (eth0, eth1, etc.) configured, and these are interconnected according figure 22 with the UML-switch daemon (see appendix E for the start-up scripts and configuration).

The zebra routing software (version 0.94) is installed in each virtual machine, so that each node can act as a router. In the experiments the behaviour of the Link State Update mechanism is studied, therefore the possibility to monitor the Link State Update creation of each separate node is necessary. To accomplish this every link connects only two nodes and the links are configured in separate IP subnets all within a single OSPF area. The subnets in OSPF are configured as broadcast subnets, although OSPF Point-to-MultiPoint subnets would probably be a better choice (no extra control traffic for Designated Router (DR) and Backup Designated Router (BDR) selection), Point-to-MultiPoint is in this version of zebra only partial implemented. Appendix F shows the configuration files of the zebra and ospf daemon of each node (*zebra.conf* and *ospfd.conf*). In total seven links and subnets are configured.

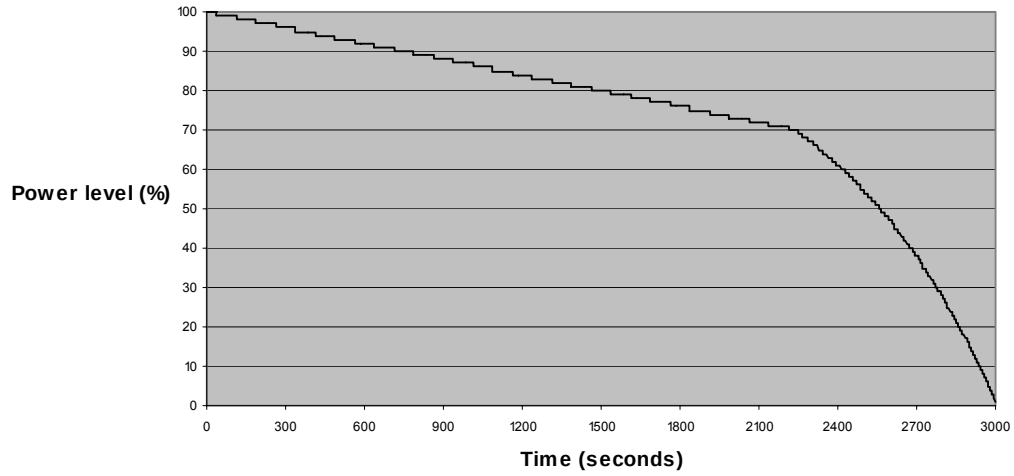
7.2 Battery lifetime model

The battery lifetime can be described using different models, see section 6.6.3. The lifetime cycle of a battery depends on many factors, for instance the battery capacity, the type of battery, the battery usage, the age of the battery. All these factors are affecting the battery lifetime cycle and therefore, the lifetime cycle is different for every battery type. Hence manufacturers of batteries only give specifications of a battery in terms of the minimum battery lifetime.

The model that is used in the experiments is derived from a discrete-time model for batteries described in an article by Benini [28]. In this article a model is presented for Lithium-Ion batteries. Today most notebook and laptop computers use Li-Ion batteries as their power supply. The article shows a graph with the lifetime cycle of a Li-Ion battery with nominal capacity of 0.5 Ah. The maximal lifetime is in this case just below 30000 seconds (about 8 hours). This graph is used as reference for the battery lifetime cycle used in the experiments.

Deducing from the graph we can say that in the first part of the lifetime cycle (roughly until the power level drops below 70%), the power level drops linearly. After 70% the power level drops exponentially to 0%. A program has been written ("*power*"), see appendix C for the source code, to create a file with timestamps and their corresponding power levels that follow this model.

Each single experiment, when performed in real-time, would take about 30000 seconds (see [28]). Therefore, the maximal lifetime of the battery, in the experiments, is scaled to 3000 seconds instead of the 30000 seconds, to shorten the duration of the experiments. Figure 23 shows the battery lifetime cycle used in most of the performed experiments.



- Figure 23: The battery lifetime cycle used in the experiments

7.3 Link State Update schemes

The Link State Updates are responsible for maintaining and updating the link costs, which are changing as a result of the power awareness routing process. Due to this fact the network will experience a load overhead. In order to measure this network load overhead we use the number of the generated Link State Updates per time unit as a metric in the experiments. Furthermore, for the same reason the bandwidth required by the generated Link State Updates is also used as a metric in these experiments.

In OSPF each node periodically broadcasts the link costs to its outgoing links to the other nodes. The other nodes, after receiving this information, can update their view of the network topology and apply a shortest path algorithm to choose the path to each destination. Normally, in a fixed network the link cost rarely change and Link State Updates are only sent in case of link state changes and topology changes (mostly caused due failures) or due to the refresh mechanism. The link states are refreshed periodically (in the Zebra software environment, the default is every half hour) to keep the routing tables consistent. In the power awareness routing scheme the link costs are changing continuously due to the power consumption of the node. To continuous broadcast Link State Updates is undesirable, because this would lead to a massive increase of control packets, hereby making the routing protocol very ineffective. The choice of when to update the link costs will largely determine the effectiveness of the routing protocol scheme.

For the different experiments we will examine two different Link State Update schemes. These schemes are:

1. The *Power-awareness LSA update scheme*: similar to OSPF, this scheme periodically refreshes the link states with the power awareness metric costs. The length of refresh period must be based on the envelope of the battery lifetime curve and the maximum lifetime of the battery. Making the refresh period to long, the link costs will not always be up-to-date and making it too short, unnecessary control packets will be send, hereby increasing the network load overhead. In most of the experiments (in the cases with a maximum lifetime of 3000 seconds) the refresh time is set equal to 50 seconds.

2. The *Enhanced Power-awareness LSA update scheme*: this scheme uses information that is associated with the battery lifetime model. If we examine the envelope of the battery lifetime cycle curve we can see that the power level is changing rapidly only when the value of the power level drops below 70% of its maximum. In the first part of the curve (i.e., power level higher than 70% of its maximum), the power drops linearly and slowly. Therefore, the generation of the Link State Updates is not frequently required and the need of keeping the other nodes informed with the actual link costs is less important. Therefore, less Link State Updates can be sent, hereby reducing the network load overhead. In this part of the battery lifetime curve, two Link State Updates are generated. One at approximately 100% of the energy level, and the other one at about 85% of the energy level. When the power level reaches 70% or lower the link states will be periodically refreshed (similar to the *Power-awareness LSA update scheme*), with a refresh time set equal to 50 seconds.

If we want to use the described battery model and both Link State Update schemes in the experiments, a number of assumptions have to be made, namely:

- All nodes have the same power consumption characteristics;
- The traffic and processing load is balanced on each node;
- The power consumption on each node is uniform and follows the battery life-time of a Li-Ion battery with nominal capacity of 0.5 Ah (see section 7.4)
- No Link State Updates due to mobility of nodes are considered in the performed experiments. Note that in reality the number of the generated Link State Updates due to mobility can be significant.
- Links state updates are not only created by the power awareness routing process, but also by the refresh mechanisms of OSPF.

7.4 Procedure followed in the experiments

In the experiments (see section 7.5) the numbers of Link State Update packets have to be measured. In all of the experiments described in section 7.5 the network configuration discussed in section 7.1 is used. To measure and extract the right information a number of tools are used and developed.

The program “power” (see appendix C for the source code) is written to configure the battery lifetime cycle (see section 7.2) of the nodes and the used Link State Update scheme (see section 7.3). Running the program, first the maximum battery lifetime must be entered. Then the desired Link State Update scheme must be chosen (with the following three choices: 1. the *Power-awareness LSA update scheme*, *2.* the *Enhanced Power-awareness LSA update scheme*, and 3. the standard Zebra OSPF implementation (no power awareness routing)), and finally the refresh time of the Link State Update scheme must be entered. The program will then create two files in the default configuration directory of zebra (./usr/local/etc). The first file is “*powerdata.txt*”, this file contains a list with timestamps and their corresponding power levels that follow the battery lifetime model described in section 7.2. The second file is “*power.conf*”, which contains the configuration of the used Link State Update scheme.

The *tcpdump* tool [29] is used in the experiment to capture data traffic on an interface. *Tcpdump* is the most used network sniffer/analyzer for Linux/UNIX. It is a powerful tool that allows sniffing network packets and enables one to make some statistical analysis out of those dumps. With *tcpdump* one can precisely monitor all the network traffic. The information that is needed in the experiments is the packet header (to identify the packet as a Link State Update), the timestamp, the packet length and the original broadcaster.

To obtain this information the following command options are used for *tcpdump*:

```
tcpdump -a -v -s 96 -q -tt ip -i eth1 >output.log
```

options: -a: Attempts to convert network and broadcast addresses to names.
-v: More verbose output. More information is printed.
-s 96: Sets larger snapshots length so no packet-information is missed.
-q: Quiet output. Print less protocol information so output lines are shorter.
-tt: Print an unformatted timestamp on each dump line.
ip: Capture IP packets.
-i eth1: Listen on interface eth1.

The output of *tcpdump* is redirected to a file (in this case output.log) to save the information. Below some lines of a possible output of the *tcpdump* tool can be seen:

```
1087851614.860843 192.168.1.2 > 224.0.0.5:  OSPFv2-hello 44: rtrid 192.168.3.1 backbone E mask 255.255.255.0 int 10 pri 1 dead 40 nbrs [tos 0xc0]  [ttl 1] (id 64186, len 64)
1087851615.880739 192.168.1.1 > 224.0.0.5:  OSPFv2-hello 48: backbone E mask 255.255.255.0 int 10 pri 1 dead 40 nbrs 192.168.3.1 [tos 0xc0]  [ttl 1] (id 29765, len 68)
1087851616.899593 192.168.1.2 > 224.0.0.5:  OSPFv2-hello 48: rtrid 192.168.3.1 backbone E mask 255.255.255.0 int 10 pri 1 dead 40 nbrs 192.168.1.1 [tos 0xc0]  [ttl 1] (id 64190, len 68)
1087851625.900369 192.168.1.1 > 224.0.0.5:  OSPFv2-hello 48: backbone E mask 255.255.255.0 int 10 pri 1 dead 40 nbrs 192.168.3.1 [tos 0xc0]  [ttl 1] (id 29767, len 68)
1087851626.920517 192.168.1.2 > 224.0.0.5:  OSPFv2-hello 48: rtrid 192.168.3.1 backbone E mask 255.255.255.0 int 10 pri 1 dead 40 nbrs 192.168.1.1 [tos 0xc0]  [ttl 1] (id 64194, len 68)
1087851635.919551 192.168.1.1 > 224.0.0.5:  OSPFv2-hello 48: backbone E mask 255.255.255.0 int 10 pri 1 dead 40 nbrs 192.168.3.1 [tos 0xc0]  [ttl 1] (id 29769, len 68)
1087851636.939836 192.168.1.2 > 224.0.0.5:  OSPFv2-hello 48: rtrid 192.168.3.1 backbone E mask 255.255.255.0 int 10 pri 1 dead 40 nbrs 192.168.1.1 [tos 0xc0]  [ttl 1] (id 64198, len 68)
1087851645.940191 192.168.1.1 > 224.0.0.5:  OSPFv2-hello 48: backbone E mask 255.255.255.0 int 10 pri 1 dead 40 nbrs 192.168.3.1 [tos 0xc0]  [ttl 1] (id 29771, len 68)
1087851647.000153 192.168.1.2 > 224.0.0.5:  OSPFv2-hello 48: rtrid 192.168.3.1 backbone E mask 255.255.255.0 int 10 pri 1 dead 40 nbrs 192.168.1.1 [tos 0xc0]  [ttl 1] (id 64202, len 68)
1087851650.887403 192.168.1.2.netbios-ns > 192.168.1.255.netbios-ns:  [udp sum ok] udp 50 (DF) (ttl 64, id 0, len 78)
1087851650.893289 192.168.1.1.netbios-ns > 192.168.1.2.netbios-ns:  udp 62 (DF) (ttl 64, id 0, len 90)
1087851652.589947 192.168.1.1 > 192.168.1.2:  OSPFv2-dd 32: backbone E I/M/MS S 40D74C85 [tos 0xc0]  [ttl 1] (id 30052, len 52)
1087851653.872367 192.168.1.2 > 192.168.1.1:  OSPFv2-dd 32: rtrid 192.168.3.1 backbone E I/M/MS S 40D74C85 [tos 0xc0]  [ttl 1] (id 11758, len 52)
1087851653.877244 192.168.1.1 > 192.168.1.2:  OSPFv2-dd 92: backbone E M S 40D74C85 { E S 80000002 age 1 rtr 192.168.1.1  } { E [tos 0xc0]  [ttl 1] (id 30053, len 112)
1087851653.890743 192.168.1.2 > 192.168.1.1:  OSPFv2-dd 52: rtrid 192.168.3.1 backbone E M/MS S 40D74C86 { E S 80000002 age 0 rtr 192.168.3.1  } [tos 0xc0]  [ttl 1] (id 11759, len 72)
1087851653.893604 192.168.1.1 > 192.168.1.2:  OSPFv2-dd 32: backbone E S 40D74C86 [tos 0xc0]  [ttl 1] (id 30054, len 52)
1087851653.895441 192.168.1.2 > 192.168.1.1:  OSPFv2-dd 32: rtrid 192.168.3.1 backbone E MS S 40D74C87 [tos 0xc0]  [ttl 1] (id 11760, len 52)
1087851653.899327 192.168.1.1 > 192.168.1.2:  OSPFv2-dd 32: backbone E S 40D74C87 [tos 0xc0]  [ttl 1] (id 30055, len 52)
1087851653.899378 192.168.1.1 > 192.168.1.2:  OSPFv2-ls_req 36: backbone { rtr 192.168.3.1  } [tos 0xc0]  [ttl 1] (id 30056, len 56)
1087851653.902565 192.168.1.2 > 192.168.1.1:  OSPFv2-ls_req 60: rtrid 192.168.3.1 backbone { rtr 192.168.1.1  } { rtr 192.168.5.1  } { net dr 192.168.5.1 if 192.168.2.2 } [tos 0xc0]  [ttl 1] (id 11761, len 80)
1087851653.907056 192.168.1.1 > 224.0.0.6:  OSPFv2-ls_upd 168: backbone { E S 80000002 age 2 rtr 192.168.1.1  } [|ospf] [tos 0xc0]  [ttl 1] (id 52726, len 188)
1087851653.938763 192.168.1.2 > 224.0.0.5:  OSPFv2-ls_upd 100: rtrid 192.168.3.1 backbone { E S 80000002 age 1 rtr 192.168.3.1  } [|ospf] [tos 0xc0]  [ttl 1] (id 64205, len 120)
1087851653.948714 192.168.1.2 > 224.0.0.5:  OSPFv2-ls_upd 60: rtrid 192.168.3.1 backbone { E S 80000001 age 1 net dr 192.168.3.1 if 192.168.1.2 mask 255.255.255.0 rtrs 192.168.1.1 192.168.3.1 } [tos 0xc0]  [ttl 1] (id 64206, len 80)
1087851654.053404 192.168.1.2 > 224.0.0.5:  OSPFv2-ls_upd 76: rtrid 192.168.3.1 backbone { E S 80000002 age 2 rtr 192.168.6.1  } [|ospf] [tos 0xc0]  [ttl 1] (id 64207, len 96)
1087851654.061724 192.168.1.2 > 224.0.0.5:  OSPFv2-ls_upd 60: rtrid 192.168.3.1 backbone { E S 80000001 age 2 net dr 192.168.6.1 if 192.168.4.2 mask 255.255.255.0 rtrs 192.168.3.1 192.168.6.1 } [tos 0xc0]  [ttl 1] (id 64208, len 80)
1087851654.620491 192.168.1.1 > 224.0.0.6:  OSPFv2-ls_ack 104: backbone { E S 80000002 age 1 rtr 192.168.3.1  } { E S 80000001 age 1 net dr 192.168.3.1 if 192.168.1.2 } [tos 0xc0]  [ttl 1] (id 52731, len 124)
1087851654.827753 192.168.1.1 > 224.0.0.6:  OSPFv2-ls_upd 88: backbone { E S 80000002 age 3 rtr 192.168.6.2  } [|ospf] [tos 0xc0]  [ttl 1] (id 52732, len 108)
1087851654.881636 192.168.1.1 > 224.0.0.6:  OSPFv2-ls_upd 124: backbone { E S 80000001 age 3 net dr 192.168.6.2 if 192.168.5.2 mask 255.255.255.0 rtrs 192.168.5.1 192.168.6.2 } [|ospf] [tos 0xc0]  [ttl 1] (id 52733, len 144)
1087851654.923818 192.168.1.2 > 224.0.0.5:  OSPFv2-ls_ack 164: rtrid 192.168.3.1 backbone { E S 80000002 age 2 rtr 192.168.1.1  } { E S 80000002 age 2 rtr 192.168.5.1  } [tos 0xc0]  [ttl 1] (id 64209, len 184)
1087851655.960304 192.168.1.1 > 224.0.0.5:  OSPFv2-hello 48: backbone E mask 255.255.255.0 int 10 pri 1 dead 40 dr 192.168.1.2 bdr 192.168.1.2 nbrs 192.168.3.1 [tos 0xc0]  [ttl 1] (id 29774, len 68)
1087851657.020241 192.168.1.2 > 224.0.0.5:  OSPFv2-hello 48: rtrid 192.168.3.1 backbone E mask 255.255.255.0 int 10 pri 1 dead 40 dr 192.168.1.2 bdr 192.168.1.1 nbrs 192.168.1.1 [tos 0xc0]  [ttl 1] (id 64211, len 68)
```

To extract the information of the output of *tcpdump*, a special program is written. The program "*countlsa*" (see appendix G for the source code) extracts the following information and saves this information (in this order) into a different file:

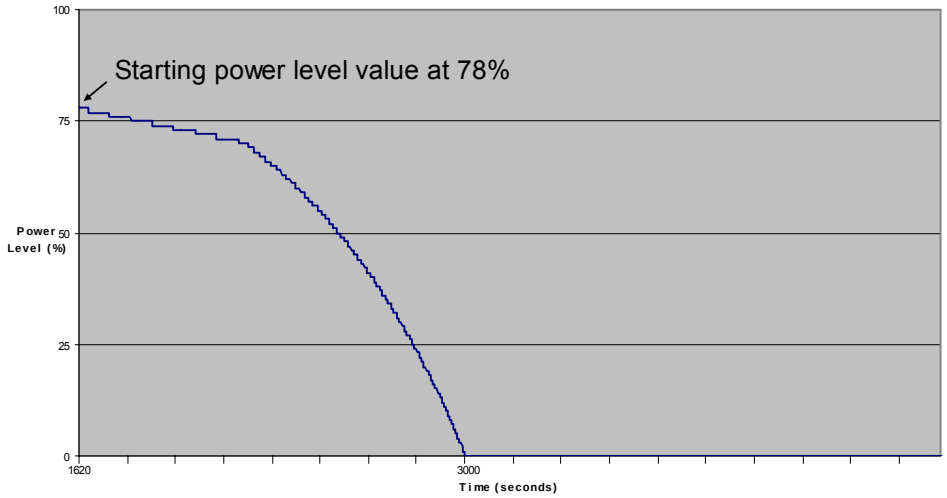
1. the relative timestamp (measured from the first timestamp);
2. a counter for the number of Link State Updates;
3. the packet length (in bytes);
4. the cumulative packet length (in bits).

This may result in the following output:

```
39.046213 1 188 1504
39.077920 2 120 2464
39.087871 3 80 3104
39.192561 4 96 3872
39.200881 5 80 4512
39.966910 6 108 5376
40.020793 7 144 6528
43.197106 8 96 7296
43.307431 9 80 7936
43.307467 10 108 8800
44.245758 11 96 9568
44.253338 12 96 10336
44.296104 13 120 11296
44.300222 14 120 12256
45.033203 15 108 13120
45.041403 16 108 13984
50.022800 17 120 14944
```

Important to note is that no distinction can be made between Link State Update packets generated by the existing OSPF operation and Link State Update packets generated by the power awareness implementation.

In some of the experiments it is necessary that the starting power level value of the battery of a node is randomly chosen. Therefore a special program "*randomize*" (see appendix H for the source code) is written that generate a uniformly distributed pseudo-random starting level value for the battery lifetime model by adapting the "powerdata.txt" file. The graph of the battery lifetime model for a node with a starting power level value of 100% is shown in figure 23 (see section 7.2). An example of a graph of the battery lifetime model for a node, after applying the program "*randomize*", is shown in figure 24. The starting power level value of this node is 78%.



- Figure 24: Battery lifetime model with a starting power level value of 78%

To measure the Link State Update packets in the different experiments the following procedure has been followed:

1. Configure on every node, by applying the program "*power*" the battery lifetime cycle and the used Link State Update scheme;

2. If necessary, apply the program "*randomize*" to generate a random starting power level value for the battery lifetime model;

3. Start the zebra daemon in every node (with the command: zebra –d);

4. Start *tcpdump* to monitor and capture the network traffic on the specified interface;

5. Start the OSPF daemon simultaneous on all nodes (with the command ospfd –d);

6. When the power level value on all of the nodes have reached 0% and the routing process has stopped, stop tcpdump and use the program "*countlsa*" to extract the results of the experiments.

7.5 The Experiments

This section describes the different experiments which are performed on the power awareness routing prototype. In all of the experiments the network configuration shown in figure 22 (see section 7.1) is used.

7.5.1 Real model vs. scaled model experiment

The discrete-time model for batteries in the article by Benini [28] has a maximum lifetime of about 30000 seconds. The experiments when performed in real-time, i.e. if the battery lifetime model of Benini is used in the experiments, the time needed to perform the different experiments will be quite large, i.e. 30000 seconds per experiment. Therefore we need to shorten the duration of each experiment. This can be done by scaling down the battery lifetime model used in the performed experiments by a factor of 10.

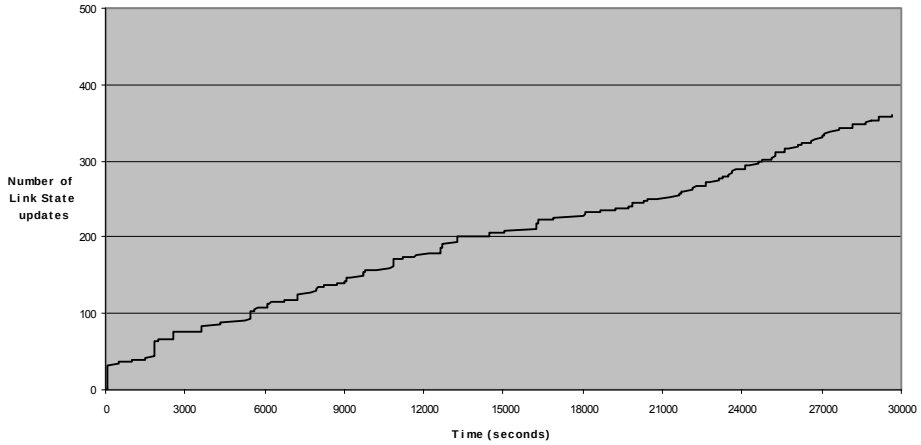
The objective of this experiment is to see what the consequences are of scaling down the maximum battery lifetime in the experiments. From now on, we refer to the battery lifetime model of Benini (with the maximum lifetime of 30000 seconds) as the *real model*, and to the scaled down battery lifetime model (with a maximum lifetime of 3000 seconds) as the *scaled model*. In this experiment a Pentium 4 2,8 GHz computer is used as host computer. All network traffic is measured from interface eth1 of Node A. No network traffic is generated except data traffic generated by the routing protocol. The Link State Updates are measured using the procedure described in section 7.4. Three of the nodes (C, D and E) have a different starting power level, node C starts at a power level of 32%, node D at 77% and node E at 96%. Node A and B start at 100%, to keep the interface up during the entire experiment. These power level values are the same in all four experiments; only the time scale is different.

There are four experiments performed, for both time scales in combination with the two proposed schemes (the Power-awareness LSA update scheme and the Enhanced Power-awareness LSA update scheme) of section 7.5. In these experiments, the calculation of the length of the LSA refresh period must be based on the envelope of the battery lifetime curve and the maximum lifetime of the battery. Making the refresh period to long, the link costs will not always be up-to-date and making it too short, unnecessary control packets will be sent, hereby increasing the network load overhead. In the experiments where the real model is used, the refresh period is chosen to be 500 seconds, while in the experiments where the scaled model is used, the refresh period is chosen to be 50 seconds (i.e., scaled down 10 times).

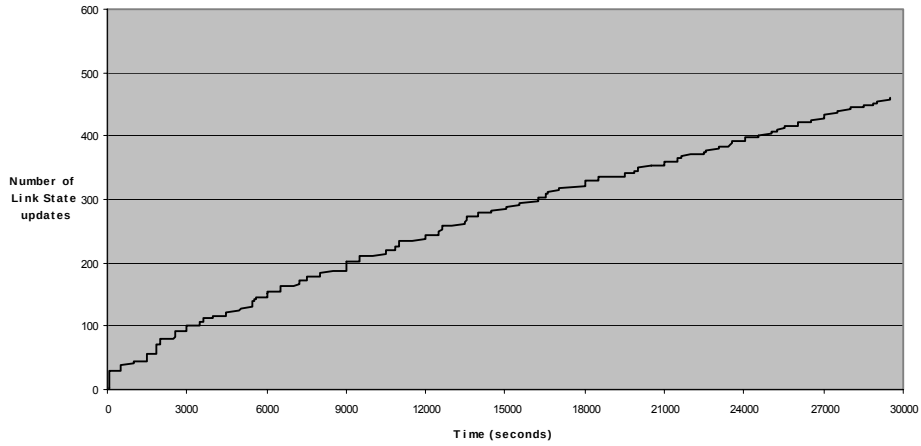
First, the experiment is performed using the *real model*. Figure 25 shows the results when the *Power-awareness LSA update scheme* is used and Figure 26 shows the results when the *Enhanced Power-awareness LSA update scheme* is used.

Figure 27 and 28 are showing the results of the experiments using the *scaled model*. Figure 27 shows the results when the *Power-awareness LSA update scheme* is used and Figure 28 when the *Enhanced Power-awareness LSA update scheme* is used. Comparing the graphs of the *scaled model* with the graphs of the *real model*, can be seen that the graphs are following a similar curve, especially in the experiment using the *Power-awareness LSA update scheme*. Only the number of generated Link State Updates is different. In the *Power-awareness LSA update scheme* the maximum number of generated Link State Updates is 459 packets vs. 194 packets (a difference of 265 packets), and in the *Enhanced Power-awareness LSA update scheme* the maximum number of generated Link State Updates is 361 packets vs. 107 packets (a difference of 254 packets). This maximum difference in Link State Updates between the *scaled model* and *real model* can be explained by the duration time of the experiments. The *real model* experiment is performed over a time period of more than

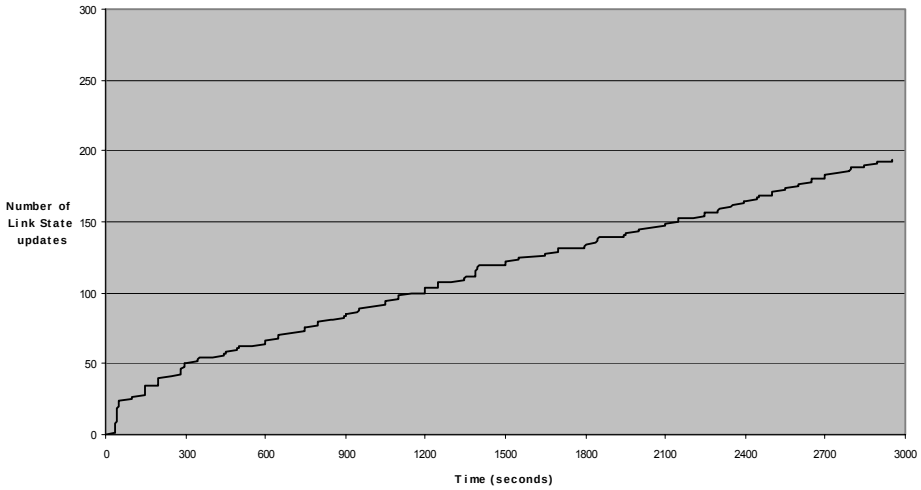
8 hours, instead experiments using the *scaled model* are performed over a time period of 50 minutes.



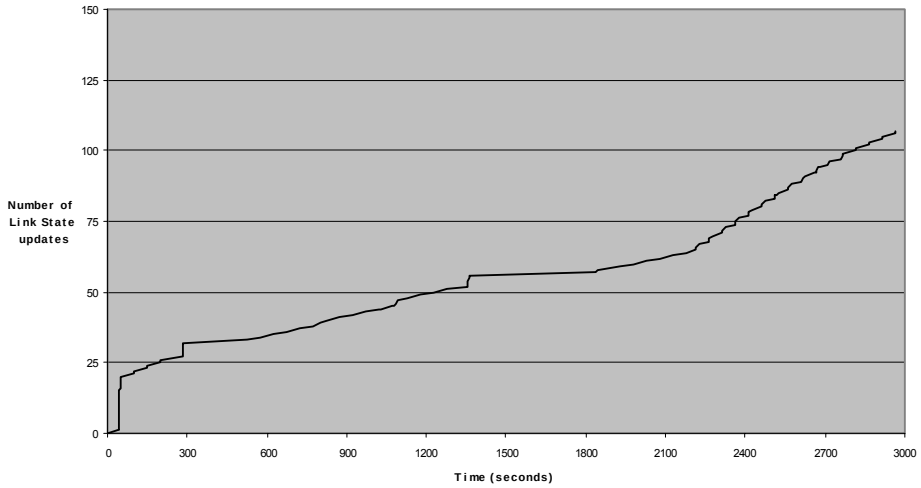
- Figure 25: Link State Updates using the *real model* (max. battery lifetime = 30000 seconds) and the *Power-awareness LSA update scheme*



- Figure 26: Link State Updates using the *real model* (max. battery lifetime = 30000 seconds) and the *Enhanced Power-awareness LSA update scheme*

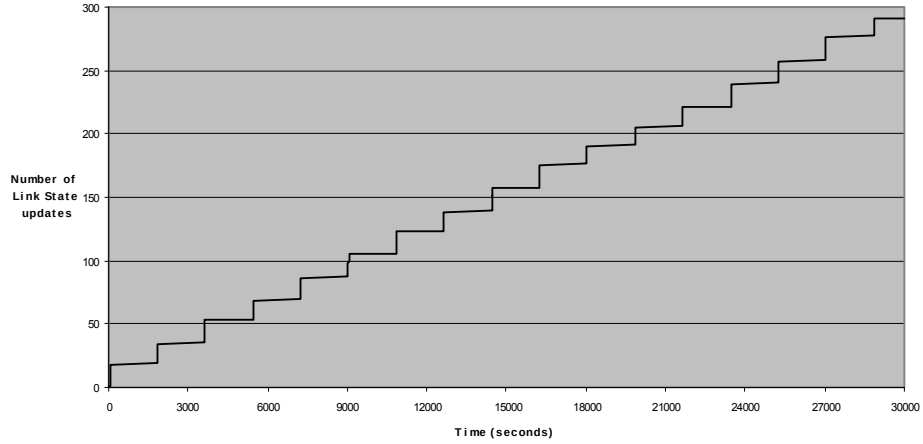


- Figure 27: Link State Updates using the *scaled model* (max. battery lifetime = 3000 seconds) and the *Power-awareness LSA update scheme*



- Figure 28: Link State Updates using the *scaled model* (max. battery lifetime = 3000 s) and the *Enhanced Power-awareness LSA update scheme*

This means that for the duration of the *real model* experiments the standardized refresh mechanisms of OSPF are generating more extra Link State Update packets, in addition to the Link State Updates generated by the power awareness schemes, than in the *scaled model*, were the duration of the experiments is much shorter. To verify these results an extra experiment is performed. In this experiment, using the same configuration as in the previous experiments, but using the OSPF routing protocol without power awareness routing (thus the standard Zebra OSPF implementation) the number of generated Link State Updates are measured. The results are shown in Figure 29, where the number of Link State Update packets generated by the standardized refresh mechanisms of OSPF is shown.



- Figure 29: Link State Updates in the standardized OSPF operation (no power awareness routing)

The graph of Figure 29 shows that approximately every 1800 seconds 15 to 20 Link State Updates are generated by the standardized refresh mechanisms of OSPF (this is in accordance with the configuration of the Zebra-OSPF implementation where the default Link State refresh time is set to 1800 seconds). In Figure 29 can also be seen that in the period from 0 to 3000 seconds (the time period of the *scaled model*) a maximum of 34 Link State Updates is generated. At 30000 seconds (the maximum battery lifetime of the *real model*) the maximum generated Link State Updates is 291. Hence, the maximum difference in Link State Updates generated by the refresh mechanisms between the *scaled model* and the *real model* is 291 – 34 = 257 Link

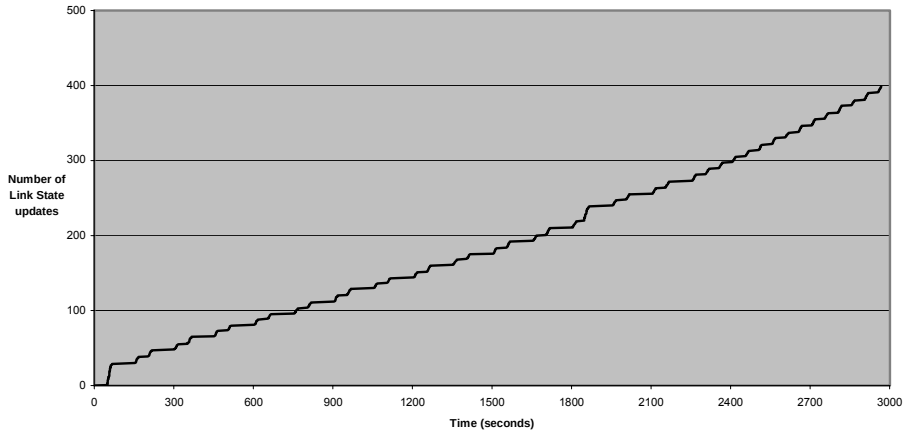
State Updates. This result is similar to the results of the experiments with the power awareness routing Link State Update schemes (the *Power-awareness LSA update scheme* had a maximum difference of 265 Link State Updates, and the *Enhanced Power-awareness LSA update scheme* had a difference of 254 Link State Updates).

These results indicate that the behavior of the refresh mechanisms, responsible for the extra generated Link State Updates, is similar in both Link State Update schemes. From this it can be concluded that if the lifetime of the battery model is scaled down, like in most of the experiments, one should take into account, that when translating the results of the experiments into reality values, the number of Link State Updates generated would be higher in reality than in the performed experiments. This is due to the additional Link State Updates generated by the standardized refresh mechanisms of OSPF.

7.5.2 Worst case experiment

In this experiment we want to examine a worst case scenario. The host computer is a Pentium 2 500 MHz machine. All five nodes will start with a fully charged battery (power level = 100%). The maximum battery lifetime is set on 3000 s. With *tcpdump* we listen at interface *eth0* of Node B to intercept the traffic. The network will be completely silent (i.e. no data traffic is generated, only traffic associated to the OSPF routing protocol). Only the Link State Updates are monitored and measured.

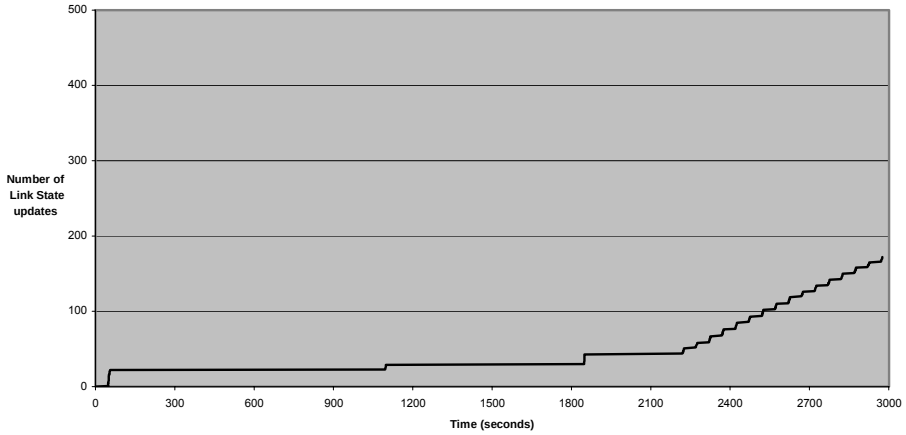
Figure 30 shows the number of Link State Updates vs. time units, measured using the *Power-awareness LSA update scheme*. From this figure can be seen that the number of the Link State Updates increases linearly with the time. This is a realistic observation, due to the fact that every node broadcasts a Link State Update every 50 seconds, resulting in a linear increase of the number of Link State Updates with the time. In total 399 Link State Updates are generated. The Link State Update scheme would generate about (3000/50) * 5 = 300 Link State Updates. The rest of the Link State Updates, i.e., 99 Link State Updates, are generated by other routing protocol processes, for instance the process used during the initialization of the routing protocol or the process that retransmits Link State Updates when some router didn't receive Link State Update acknowledgements on time. After about 1800 seconds in the experiments the standardized refresh mechanisms of OSPF are also generating extra Link State Updates (about 15 to 20), see section 7.5.1.



- Figure 30: Link State Updates using the *Power-awareness LSA update scheme*

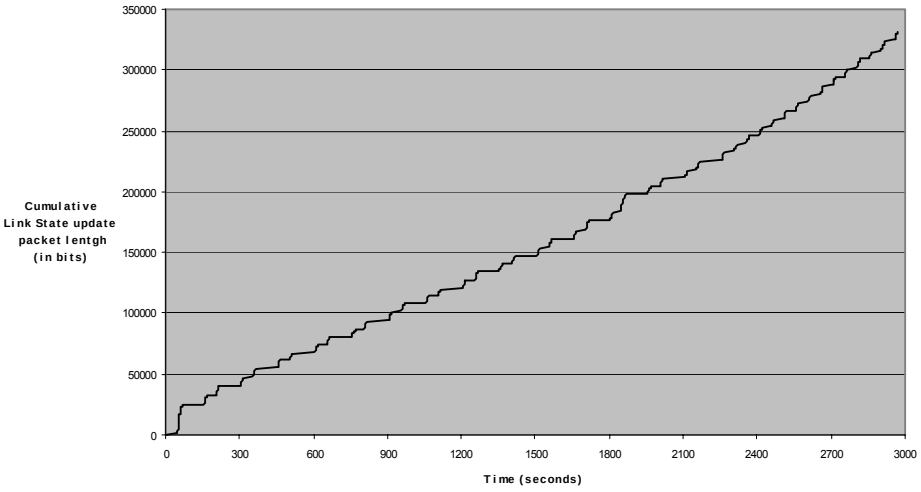
Figure 31 shows the results when the *Enhanced Power-awareness LSA update scheme* is used in the experiment. The path followed by this curve is quite different.

We see, as expected, that until the timestamp at 2220 seconds only a few Link State Updates have been created. Only after 2220 seconds (when the power level drops below 70%) Link State Updates are regular generated in the same way as in the *Power-awareness LSA update scheme*. This shows that in the *Enhanced Power-awareness LSA update scheme* the number of generated Link State Updates (maximal 172) is much lower than in the *Power-awareness LSA update scheme*, creating much less overhead on the network.



- Figure 31: Link State Updates using the *Enhanced Power-awareness LSA update scheme*

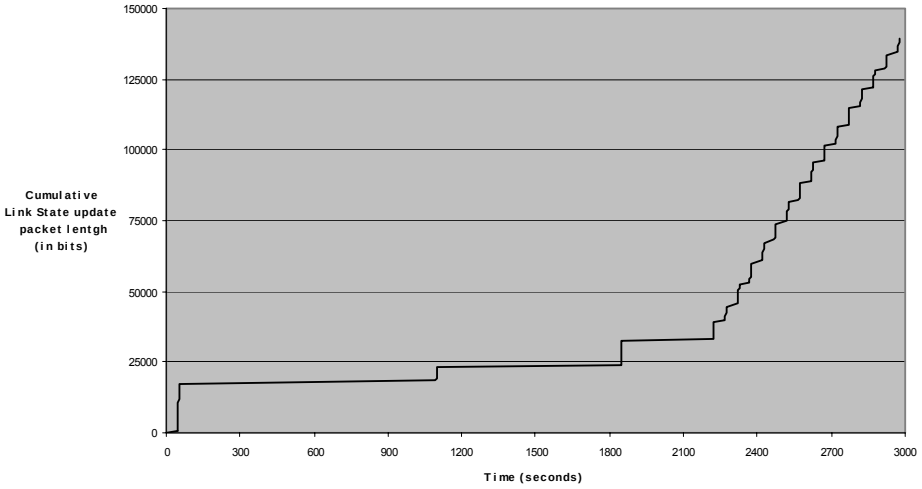
Figure 32 shows the cumulative packet length (in bits) of all of the generated Link State Update packets with the *Power-awareness LSA update scheme*. In the graph can be seen that there is little variation in the length of the Link State Update packets and the path of the graph corresponds with the path of Figure 30.



- Figure 32: Cumulative packet length of Link State Update packets using the *Power-awareness LSA update scheme*

Figure 33 shows the cumulative packet length (in bits) of all of the generated Link State Update packets with the *Enhanced Power-awareness LSA update scheme*. Similar to Figure 32 this graph follows a similar curve as the graph of Link State Updates with the *Enhanced Power-awareness LSA update scheme* (Figure 31). Comparing figure 32 and 33 shows that the *Power-awareness LSA update scheme*

generates much more data, thus requires more bandwidth, than the *Enhanced Power-awareness LSA update scheme*. Dividing the total amount of generated bits with the total number of Link State Updates packets results, with the *Power-awareness LSA update scheme* in an average packet length of 331040 / 399 = 829,67 bits/packet, and with the *Enhanced Power-awareness LSA update scheme* in an average packet length of 139456 / 172 = 810,79 bits/packet. This shows that the choice of Link State Update schemes does not affect the size of the generated Link State Update packets. Important to note is that in this experiment the Link State Acknowledgment packets, which follow every Link State Update, are not monitored.



- Figure 33: Cumulative packet length of Link State Update packets using the *Enhanced Power-awareness LSA update scheme*

7.5.3 Mean value experiments

To compare the two Link State Update schemes of section 7.3 we want to determine the mean value of the number of generated Link State Updates of both of the Link State Update schemes. In the experiments, needed to determine the mean value of the number of generated Link State Updates, a Pentium 2 500 MHz machine is used as host computer. The experiments need to be repeated a number of times to provide sufficient samples (at least 31 samples are required) for the calculation of the confidence interval [30].

The confidence interval of the calculated mean values is calculated using the following equation:

$$(\overline{x} - z_{1-\alpha/2}\ s/\sqrt{n}\ ,\quad \overline{x} + z_{1-\alpha/2}\ s/\sqrt{n}\ )$$

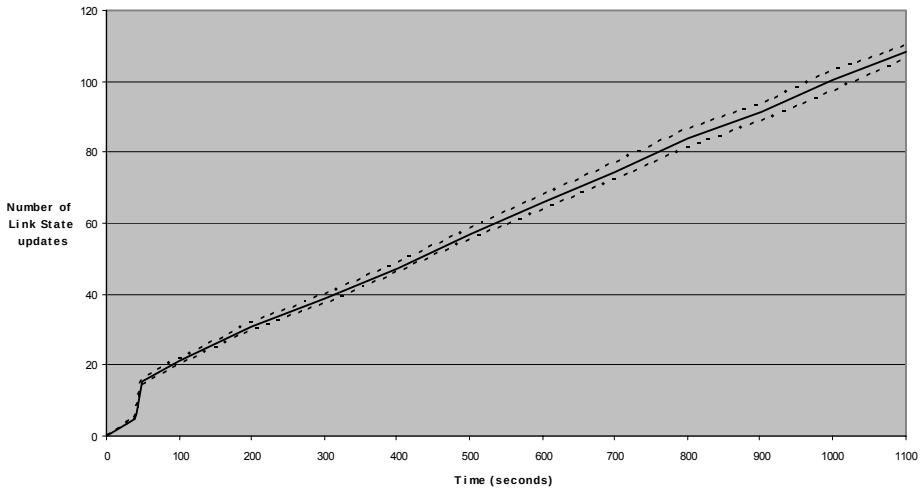
- $\overline{x}$ is the sample mean, the mean of the generated Link State Updates of all experiments at one time interval;
- s is the sample standard deviation of all the generated Link State Updates of all experiments at one time interval;
- n is the number of samples (the number of experiments);
- $z_{1-\alpha/2}$ is the (1 – α/2)- quantile of a unit normal variate. The value of $z_{1-\alpha/2}$ is 1,645, with a 90% confidence interval (value taken from Table A.2 in the Appendix of [30]).

In the experiments the procedure described in section 7.4 is followed. Each node will have a uniformly distributed random starting level value for the battery lifetime, determined by the program "*randomize*" (see section 7.4), except for node B which is used as point of measure and starts in each experiment with a power level value of 100%. The network traffic is intercepted at interface *eth0* of Node B using *tcpdump*. No other network traffic is generated except the traffic generated by the routing protocol.

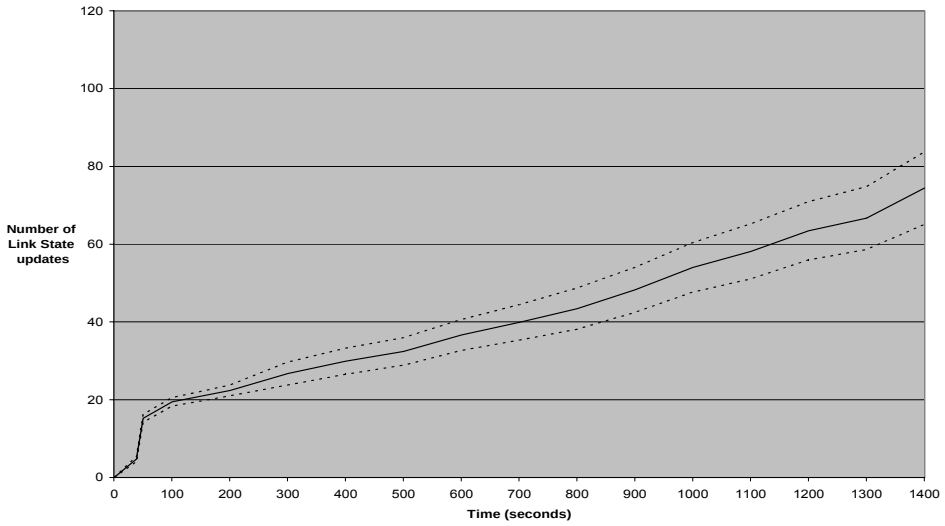
The experiment is repeated a number of times to provide the necessary samples. In each experiment the starting level value for the battery lifetime for node A, C, D and E will be random determined (uniformly distributed in the range between 0 and 100%). The results of these experiments are used to calculate the mean values of the generated Link State Updates at 100 seconds time intervals. Note that in order to cover the initialization period of the routing protocol, the number of generated Link State Updates are also monitored at 39, 40 and 50 seconds.

The mean and its confidence intervals for the Link State Update schemes are depicted in Figure 34 for the *Power-awareness LSA update scheme* and in Figure 35 for the *Enhanced Power-awareness LSA update scheme*. The confidence intervals are forming the two bounds for the mean value (shown as the black line), a lower bound (shown as the lower dotted line) and an upper bound (shown as the upper dotted line). We can state with 90% confidence that the mean of the number of generated Link State Updates at any given time lies in the area between the lower and upper bound.

The node B is used as measurement point and therefore its starting power level value is set to 100%, meaning that the power level of this node will not reach 0% before an experiment is completed. Due to the fact that the starting power level value of each other node is chosen based on a uniformly distributed fashion, the power level on such a node may reach the 0% power level before an experiment is completed.



- Figure 34: Mean value experiment using the *Power-awareness LSA update scheme*



- Figure 35: Mean value experiment using the *Enhanced Power-awareness LSA update scheme*

This situation can also occur for node A, which is connected to the same link used by node B as measurement point. This means that when the power level value of node A has reached 0%, the link goes down and it cannot longer be used for measurements. For each of the two Link State Update schemes about 80 experiments are performed. Due to the above mentioned reasons, none of these experiments could be performed up to the maximum experiment time, i.e., 3000 seconds. For the scenario where the *Power-awareness LSA update scheme* is used, the experiments could be run up to 1100 seconds, see Figure 34, while for the scenario where the *Enhanced Power-awareness LSA update scheme* is used, see Figure 35, the experiments could be run up to 1400 seconds.

Due to the long duration of each experiment and due to the lack of available time, the number of experiments could not be further increased. However, based on the shown results we can conclude that the mean number of generated Link State Updates due to power awareness, when the *Enhanced Power-awareness LSA update scheme* is used, is much lower than the mean number of generated Link State Updates due to power awareness, when the *Power-awareness LSA update scheme* is used. In particular, when the experiment time is higher than 100 seconds, the slope of the linear curve shown in Figure 34 (for the *Power-awareness LSA update scheme)* is at least more than twice steeper than the slope of the linear curve shown in Figure 35 (for the *Enhanced Power-awareness LSA update scheme).*

8 Conclusions and recommendations

In this thesis a number of subjects are studied in relation to power awareness routing. Furthermore, this thesis describes the design and implementation of a power awareness routing prototype for mobile ad hoc networks.

The mobile ad hoc network technology has many advantages due to the lack of need of an existing network infrastructure or centralized administration. One drawback of mobile ad hoc networks is that the wireless devices usually are powered by batteries which limit the lifetime of the devices and ultimately the lifetime of the entire network. The development of power conservation schemes (to reduce the energy consumption of wireless devices), and the improvement of batteries (increasing the battery lifetime) will extend the applicability of ad hoc networks in the future. One solution for power conservation in mobile ad hoc networks is power awareness routing.

At the moment the leading technology in mobile ad hoc networks is the IEEE 802.11 standard. Although IEEE 802.11 supports one form of power conservation mode, by setting wireless devices into Awake of Doze mode during the idle time, however no services are implemented to reduce the power consumption during communication. Furthermore IEEE 802.11 does not support multihop communication and unidirectional links, which are favourable in ad hoc networks. To implement an efficient power awareness routing scheme, multihop communication and unidirectional links support is needed.

For an implementation of power awareness routing a number of design aspects of routing protocols are favourable, making some ad hoc routing protocols more suitable for the implementation of a power awareness routing prototype. The following design aspects are favourable for our power awareness routing prototype:

- A table-driven protocol: It is favourable that information about the network topology is maintained in all routers, and that any change in network topology is quickly forwarded to the other nodes. Furthermore the global overview of network topology allows to predict future network behavior and this can be used to determine future quality-of-service (QoS).
- Link-state routing: For the prototype, the ability to calculate the optimum route in an ad hoc network based on power awareness parameters is needed. This requires that the metric cost should be freely chosen.
- A flat network architecture: Assume that the functionality of all nodes is equal, to ensure that no nodes will attract more network traffic than others, so that their battery is not depleted faster.
- Full-topology routing: A full and recent overview of the routing topology of the entire network is required, to prevent that routing decisions are based on outdated information.
- Support for unidirectional links: This is a desirable feature; it makes the routing protocol suitable for more complex wireless network environments.

Three routing protocol are meeting these criteria, namely TBRPF-FT, FSR and the conventional link state protocol OSPF. Of these three the TBRPF-FT is the best option for implementing power awareness routing in a mobile ad hoc network. However for practical reasons, power awareness routing is implemented in this thesis using the OSPF protocol.

A significant work was needed before the power awareness routing scheme could be implemented into an existing Zebra OSPF implementation. The lack of documentation and pseudo code for the C source code of Zebra lead to a complicated implementation that took a long time to be realized.

In the power awareness routing prototype the power level value of the battery is used as the metric cost. The power awareness routing is implemented in a node such that at certain times the power level of the battery is monitored and if necessary a Link State Update is generated to the other nodes in the network to update their routing tables. The battery lifetime cycle is simulated by using a file with timestamps and corresponding power level values according to a predetermined model for the lifetime cycle of a battery. This battery lifetime model is based on a discrete-time model for batteries, with the maximum lifetime scaled down (with a factor of 10). Two different Link State Update schemes are implemented into the power awareness routing prototype. The *Power-awareness LSA update scheme*; in which periodically the power awareness metric cost is updated and a Link State Update is broadcasted, and the *Enhanced Power-awareness LSA update scheme*; in which the battery lifetime cycle is divided into two areas (one area where the battery power level value changes relative slow and only a few Link State Updates are broadcasted, and an area where the battery power level value changes rapidly and the Link State Updates are periodically broadcasted).

With the User-Mode Linux solution a virtual network has been build on a single host computer. This virtual network is used to perform three types of experiments on the power awareness routing prototype. The *real model* vs. *scaled model* experiment shows that scaling down the maximal lifetime of the battery has a limited effect on the number of Link State Update generated by the power awareness routing implementation. However, the duration of the experiment did have an effect on the number of Link State Updates that are generated by the standardized refresh mechanisms of OSPF. This shows that the periodic broadcast of Link State Updates of OSPF create a considerably overhead in control traffic, making it less favourable for mobile ad hoc protocols. The ad hoc protocol TBRPF broadcasts, unlike OSPF, only Link State Updates when changes in topology are reported. This minimizes the overhead and making TBRPF more efficient in mobile ad hoc networks than OSPF.

The results of the worst case experiment show that if the battery lifetime cycle of a device is known in advance or can be predicted, and when this information is then used to determine the optimal moments to broadcast a Link State Update, the amount of control traffic on the network can be significant reduced. The results of the mean value experiments show that repeating the experiments a number of times, little variations in the results of the experiments are observed, thus making a single experiment reasonably reliable.

The different experiments have shown that the power awareness routing implementation in zebra is working correctly and can be implemented into a Linux environment. Moreover the experiments have shown that when the battery lifetime models are known, the operation of the power awareness routing scheme can be influenced such that the generated overall network overhead is significantly reduced. Furthermore it shows that the User-Mode Linux can be used to create virtual networks that emulate real network scenarios, where network experiments can efficiently be performed.

This thesis can be seen as a basic study on power awareness routing. For future work on this subject, the following recommendations are made:

- study and implement different algorithms for creating Link State Updates using the power awareness routing prototype;
- study battery lifetime models of wireless devices, and investigate the possibility of using this information online (in real-time) to predict future topology changes in the network;
- implement a ad hoc routing protocol (e.g. TBRPF) into the power awareness routing prototype, and compare it with OSPF;
- implement the power awareness routing prototype into a real mobile ad hoc network.

References

[1] Bluetooth, http://www.bluetooth.com/index.asp

[2] International Standard ISO/IEC 8802-11, “Part 11: Wireless LAN Medium Access (MAC) and Physical Layer (PHY) Specifications”, Published by the IEEE, 1999 Edition

[3] HIPERLAN, http://portal.etsi.org/bran/kta/Hiperlan/hiperlan2.asp

[4] Malkin, " RIP Version 2", RFC2453, November 1998

[5] Moy, “OSPF Version 2”, RFC2328, April 1998

[6] Perkins and Bhagwat, "Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers", Proceedings of ACM SIGCOMM'94, London, Sep. 1994

[7] Murthy and Garcia-Luna-Aceves, "An Efficient Routing Protocol for Wireless Networks", ACM Mobile Networks and App. J., Special Issue on Routing in Mobile Communication Networks, Oct. 1996, pp. 183-97.

[8] Tsu-Wei Chen and Mario Gerla, "Global State Routing: A New Routing Scheme for Ad-hoc Wireless Networks", Proc. IEEE ICC'98, 5 pages

[9] Jacquet, Muhlethaler, Qayyum, Laouiti, Viennot and Clausen, “Optimized Link State Routing Protocol, IEFT Internet draft, draft-ietf-manet-olsr-04.txt, 2 March 2001

[10] Bellur, Ogier, and Templin, “Topology Broadcast Based on Reverse-Path Forwarding (TBRPF)”, IEFT Internet draft, draft-ietf-manet-tbrpf-01.txt, 2 March 2001

[11] Pei, Gerla, and Chen, “Fisheye State Routing Protocol (FSR) for Ad Hoc Networks”, IEFT Internet draft, draft-ietf-manet-fsr-01.txt, 17 November 2000

[12] Perkins and Royer, “Ad Hoc On Demand Distance Vector (AODV) Routing”, IEFT Internet draft, draft-ieft-manet-aodv-02.txt, Nov 1998 (Work in progress)

[13] Johnson, Maltz and Hu, “The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks”, IEFT Internet draft, draft-ietf-manet-dsr-05.txt, 2 March 2001 (Work in progress)

[14] Park, Corson, “Temporally-ordered routing algorithm (TORA)”, version 1 Internet Draft, draft-ietf-manet-tora-spec- 03.txt, June 2001

[15] Toh, “Associativity-Based Routing for Ad hoc Networks”, Wireless Pers. Commun., vol 4, no. 2, March 1997, pp 1-36

[16] Lee and Gerla, “Split Multipath Routing with Maximally Disjoint Paths in Ad Hoc Networks”, 2000

[17] Narayanaswamy, Kawadia, Sreenivas, Kumar, “Power Control in Ad-Hoc Networks: Theory, Architecture, Algorithm and Implementation of the COMPOW Protocol”, 2002

[18] Lindgren, Schelén, “Infrastructured ad hoc networks”, 2002

[19] Misra, Banerjee, “Maximizing Network Lifetime for Reliable Routing in Wireless Environments”, 2002

[20] Gomez, Campbell, Naghshineh, Bisdikian, “PARO: A Power-Aware Routing Optimization Scheme for Mobile Ad hoc Networks”, Internet draft: draft-gomez-paro-manet-00.txt, February 2001

[21] Xu, Heidemann, Estrin, “Adaptive Energy-Conserving Routing for Multihop Ad Hoc Networks”, 2000

[22] The Linux Documentation Project, http://www.tldp.org

[23] The User-mode Linux Kernel Home Page, http://user-mode-linux.sourceforge.net/

[24] GNU Zebra, http://www.zebra.org

[25] GNU Operating System - Free Software Foundation, http://www.gnu.org/

[26] IP Infusion, http://www.ipinfusion.com/

[27] G. Munz, “OSPF Link-State Database in GNU Zebra”, Student’s project, Ecole Nationale des telecommunications Department Informatique et Reseaux, 2002.

[28] Benini, Castelliz, Maciiz, Maciiz, Poncinoz, Scarsiz, “A Discrete-Time Battery Model for High-Level Power Estimation”

[29] Tcpdump, http://www.tcpdump.org/

[30] Jain, “The art of computer system performance analysis, Techniques for experimental design, measurement, simulation, and modeling”, Wiley, 1991

Appendix

Appendix A: Ad hoc protocol list

Appendix B: APM power awareness routing implementation

Appendix C: Source code of power program

Appendix D: Source code power awareness routing implementation

Appendix E: User-mode linux script files

Appendix F: Zebra/OSPFd configuration files

Appendix G: Source code of countlsa program

Appendix H: Source code of randomize program

Appendix A: Ad hoc protocol list

- ***ABR*** (Associativity Based Routing protocol) - C.-K. TOH ASSOCIATIVITY-BASED LONG-LIVED ROUTING (ABR) PROTOCOL , Internet Draft.

- ***AODV*** (Ad hoc On Demand Distance Vector routing protocol) - C. PERKINS, E.ROYER AND S. DAS Ad hoc On-demand Distance Vector (AODV) Routing, Internet Draft, draft-ietf-manet-aodv-11.txt, work in progress, Aug 2002.

- ***CBRP*** (Cluster Based Routing Protocol) - M. JIANG, J. LI, Y. C. TAY Cluster Based Routing Protocol (CBRP) Functional Specification Internet Draft, draft-ietf-manet-cbrp.txt, work in progress, June 1999.

- ***CEDAR*** (Core Extraction Distributed Ad hoc Routing) - RAGHUPATHY SIVAKUMAR, PRASUN SINHA, VADUVUR BHARGHAVAN Core Extraction Distributed Ad hoc Routing (CEDAR) Specification, Internet Draft, draft-ietf-manet-cedar-spec-00.txt

- ***CGSR*** (Clusterhead Gateway Switch Routing protocol) - Clusterhead Gateway Switch Routing protocol (CGSR) S. Murthy and J.J. Garcia-Luna-Aceves, "An Efficient Routing Protocol for Wireless Networks", ACM Mobile Networks and App. J., Special Issue on Routing in Mobile Communication Networks, Oct. 1996, pp. 183-97.

- ***DBF*** (Distributed Bellman-Ford routing protocol) - SHREE MURTHY, J.J. GARCIA-LUNA-AVECES Distributed Bellman-Ford routing protocol (DBF), A Routing Protocol for Packet Radio Networks, Proc. ACM International Conference on Mobile Computing and Networking, pp. 86-95, November, 1995.

- ***DDR*** (Distributed Dynamic Routing Algorithm) - NAVID NIKAEIN, HOUDA LABIOD, CHRISTIAN BONNET Distributed Dynamic Routing Algorithm (DDR) for Mobile Ad Hoc Networks, in proceedings of the MobiHOC 2000 : First Annual Workshop on Mobile Ad Hoc Networking & Computing.

- ***DREAM*** (Distance Routing Effect Algorithm for Mobility) - S. BASAGNI, I. CHLAMTAC, V. R. SYROTIUK, B. A. WOODWARD A Distance Routing Effect Algorithm for Mobility (DREAM) In Proc. ACM/IEEE Mobicom, pages 76-84, October 1998.

- ***DSDV*** (Higly Dynamic Destination-Sequenced Distance Vector routing protocol) - C. E. PERKINS, P. BHAGWAT Higly Dynamic Destination-Sequenced Distance Vector (DTDV) for Mobile Computers Proc. of the SIGCOMM 1994 Conference on Communications Architectures, Protocols and Applications, Aug 1994, pp 234-244.

- ***DSR*** (Dynamic Source Routing protocol) - D. JOHNSON, D. MALTZ, Y-C. HU AND J. JETCHEVA: The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks, Internet Draft, draft-ietf-manet-dsr-05.txt, work in progress, June 2001.

- ***FORP*** (Flow Oriented Routing Protocol)

- ***FSR*** (Fisheye State Routing protocol) - MARIO GERLA, GUANGYU PEI, XIAOYAN HONG, TSU-WEI CHEN Fisheye State Routing Protocol (FSR) for Ad Hoc Networks Internet Draft, draft-ietf-manet-fsr-00.txt, work in progress, June 2001.

- ***GLS(Grid)*** (Geographic Location Service) - JINYANG LI, JOHN JANOTTI, DOUGLAS S. J. DE COUTU, DAVID R. KARGER, ROBERT MORRIS A Scalable Location Service for Geographic Ad Hoc Routing M.I.T. Laboratory for Computer Science

- ***GPSAL*** (GPS Ant-Like Routing Algorithm) - Daniel Câmara, Antonio Alfredo F. Loureiro, A Novel Routing Algorithm for Hoc Networks, Baltzer Journal of Telecommunications Systems, 18:1-3, 85-100, Kluwer Academic Publishers, 2001.

- ***GSR*** (Global State Routing protocol) - Global State Routing protocol (GSR) [Iwata99] A. Iwata, C.-C. Chiang, G. Pei, M. Gerla, and T.-W. Chen, "Scalable Routing Strategies for Ad Hoc Wireless Networks" IEEE Journal on Selected Areas in Communications, Special Issue on Ad-Hoc Networks, Aug. 1999, pp.1369-79.

- ***ISAIAH*** (Infra-Structure Aodv for Infrastructured Ad Hoc networks) - ANDERS LINDGREN AND OLOV SCHELÉN Infrastructured ad hoc networks In Proceedings of the 2002 International Conference on Parallel Processing Workshops (International Workshop on Ad Hoc Networking (IWAHN 2002)). pages 64-70. August 2002.

- ***LANMAR*** (Landmark Routing Protocol for Large Scale Networks) - MARIO GERLA, XIAOYAN HONG, LI MA, GUANGYU PEI Landmark Routing Protocol (LANMAR) Internet Draft, draft-ietf-manet- lanmar-01.txt, work in progress, June 2001.

- ***LAR*** (Location-Aided Routing protocol) - Y.-B. KO, V. N. H. Location-Aided Routing in mobile Ad hoc networks In Proc. ACM/IEEE Mobicom, pages 66-75, October 1998.

- ***LMR*** (Lightweight Mobile Routing protocol) - M.S. CORSON AND A. EPHREMIDES Lightweight Mobile Routing protocol (LMR) ,A distributed routing algorithm for mobile wireless networks, Wireless Networks 1 (1995).

- ***OLSR*** (Optimized Link State Routing Protocol) - PHILIPPE JACQUET, PAUL MUHLETHALER, AMIR QAYYUM, ANIS LAOUITI, LAURENT VIENNOT, THOMAS CLAUSEN Optimized Link State Routing Protocol Internet Draft, draft-ietf-manet-olsr-04.txt, work in progress, June 2001.

- ***PAMAS*** (PAMAS-Power Aware Multi Access Protocol with Signaling Ad Hoc Networks) - S. SINGH, C.S. RAGHAVENDRA PAMAS & PAMAS-Power Aware Multi Access Protocol with Signaling Ad Hoc Networks.

- ***PARO*** (Power-Aware Routing Optimization Protocol) - J. GOMEZ, A. T. CAMPBELL, M. NAGHSHINEH, C. BISDIKIAN, T.J. WATSON POWER-AWARE ROUTING OPTIMIZATION PROTOCOL (PARO) Internet Draft, draft-gomez-paro-manet-00.txt, work in progress, June 2001.

- ***SSR*** (Signal Stability Routing protocol) - R. DUBE, C. D. RAIS, K. WANG, AND S. K. TRIPATHI Signal Stability based adaptive routing (SSR alt SSA) for ad hoc mobile networks, IEEE Personal Communication, Feb. 1997.

- ***STAR*** (Source Tree Adaptive routing protocol) - J.J. GARCIA-LUNA, M. SPOHN Source Tree Adaptive Routing Internet Draft, draft-ietf-manet-star-00.txt, work in progress, October 1999.

- ***TBRPF*** (Topology Broadcast based on Reverse-Path Forwarding routing protocol) - BHARGAV BELLUR, RICHARD G. OGIER, FRED L. TEMPLIN Topology Broadcast Based on Reverse-Path Forwarding (TBRPF) Internet Draft, draft-ietf-manet-tbrpf-01.txt, work in progress, June 2001.

- ***TORA*** (Temporally-Ordered Routing Algorithm routing protocol) - V. PARK, S. CORSON TEMPORALLY-ORDERED ROUTING ALGORITHM (TORA) VERSION 1 Internet Draft, draft-ietf-manet-tora-spec- 03.txt, work in progress, June 2001.

- ***WRP*** (Wireless Routing Protocol) - Wireless Routing Protocol (WRP) [Chen98] Tsu-Wei Chen and Mario Gerla, "Global State Routing: A New Routing Scheme for Ad-hoc Wireless Networks" Proc. IEEE ICC'98, 5 pages.

- ***ZHLS*** (Zone-Based Hierarchical Link State Routing) - JOA NG, I-TAI LU Zone-Based Hierarchical Link State Routing (ZHLS). An abstract routing protocol and medium access protocol for mobile ad hoc networks Submitted for partial fulfillment of the requirements for the degree of doctor of philosophy (Electrical engineering) in January 1999.

- ***ZRP*** (Zone Routing Protocol protocol) - ZYGMUNT J. HAAS, MARC R. PEARLMAN, PRINCE SAMAR THE BORDERCAST RESOLUTION PROTOCOL (BRP) Internet Draft, draft- ietf-manet-zone-zrp-04.txt, work in progress, July 2002.

Appendix B: APM power awareness routing implementation

Modification of function ospf_interface.c: ospf_if_get_output_cost() to determine the power status using the Advanced Power Management control program (*APM*). The apm command options are for netBSD, for Linux these are slightly different.

```
ospf_if_get_output_cost (struct ospf_interface *oi)
 {
  /* If all else fails, use default OSPF cost */
  u_int32_t cost,i;
  u_int32_t bw, refbw;

  FILE *p;

  bw = oi->ifp->bandwidth ? oi->ifp->bandwidth : OSPF_DEFAULT_BANDWIDTH;
  refbw = ospf_top ? ospf_top->ref_bandwidth : OSPF_DEFAULT_REF_BANDWIDTH;

  /* A specifed ip ospf cost overrides a calculated or power awareness one. */
  if (OSPF_IF_PARAM_CONFIGURED (IF_DEF_PARAMS (oi->ifp), output_cost_cmd) ||
     OSPF_IF_PARAM_CONFIGURED (oi->params, output_cost_cmd))
   cost = OSPF_IF_PARAM (oi, output_cost_cmd);

   /* See if a cost can be calculated from the power awareness routing process */

   else    {
     /* Power awareness routing implementation */
     /* Get powerstatus (using apm) or else calculate cost from the zebra processes interface bandwidth field */
                if ((p = popen("apm -l", "r")) == NULL)
                cost = (u_int32_t) ((double)refbw / (double)bw + (double)0.5);
                else {
                        fscanf(p,"%u", &i);
                        cost = (101 -i);
                       }
                if (cost < 1) cost = 1;
                else if (cost > 65535) cost = 65535;
                pclose(p);
                }
  return cost;
 }
```

Appendix C: Source code of power program

This program is used to create the battery lifetime model (in file “powerdata.txt”) in the default zebra configuration directory (./usr/local/etc). Further it sets the used Link State Update scheme and time intervals in file “power.conf”, also in the default zebra configuration directory (./usr/local/etc).

```
#include <stdio.h>
#include <time.h>
#include <math.h>

void setup_power (void);
void power_init(void);

int power_mode, time_interval, power_interval;
struct Powerstatus	{
				int		timestamp;
				double	power;
			};

struct Powerstatus entry;

main()
{
		setup_power();
		power_init();
}

/* Setup the file with the power metric costs */
void setup_power (void)
{
		int		i,choice;
		double maxtime,max,x1,x2;
		FILE *Out;

		printf("Create file with powerstatus to simulate APM function.\n\n");
		printf("Enter maximum lifetime of battery (in seconds): ");
		scanf("%lf", &maxtime);

		if ((Out = fopen("/usr/local/usr/powerdata.txt", "w")) == NULL) fprintf(stderr, "File could not be opened.\n");
		else
			{
			for (i=0;i<=maxtime; ++i) {
				entry.timestamp=i;
				max=maxtime/0.8537;
				x1 = 101-(exp((i*4.605170186)/maxtime));
				x2 = 100-((40*i)/maxtime);
				if (x2>70) entry.power= x2;
				else entry.power = x1;
				fprintf(Out,"%d %0.0f\n",entry.timestamp, entry.power);
					}
			}
		fclose(Out);
		break;
return;
}

/* Which Power awareness routing must be used? */
void power_init(void)
{
  FILE *confout;

question:
		printf("\nWhich OSPF Link State Update scheme should be used?");
		printf("\n1. Power-awareness LSA update scheme.");
		printf("\n2. Enhanced Power-awareness LSA update scheme.");
		printf("\n3. Standard Zebra OSPF implementation (no power awareness routing).");
		printf("\nEnter your choice (1,2 or 3): ");
		scanf("%d", &power_mode);
```

```
        switch (power_mode) {

        case (1):   printf("\nGive interval (in seconds) for power metric updates: ");
                    scanf("%d", &time_interval);

                    /* save setting in configuration file power.conf */
                    if ((confout = fopen("/usr/local/usr/power.conf", "w")) == NULL)
                            fprintf(stderr, "File could not be opened.\n");
                    else fprintf(confout,"%d %d\n",power_mode, time_interval);

                    fclose(confout);
                    break;

        case (2):   printf("\nGive interval (in seconds) for power metric updates: ");
                    scanf("%d", &power_interval);

                    /* save setting in configuration file */
                    if ((confout = fopen("/usr/local/usr/power.conf", "w")) == NULL)
                              fprintf(stderr, "File could not be opened.\n");
                    else fprintf(confout,"%d %d\n",power_mode, power_interval);

                    fclose(confout);
                    break;

        case (3):   /* save setting in configuration file */
                    if ((confout = fopen("/usr/local/usr/power.conf", "w")) == NULL)
                            fprintf(stderr, "File could not be opened.\n");
                    else fprintf(confout,"%d %d\n",power_mode, 0);

                    fclose(confout);
                    break;

        default:    printf("\nIncorrect value!\n");
                    goto question;
        }
return;
}
```

Appendix D: Source code power awareness routing implementation

```
#define POWER_CHECK_INTERVAL 10 /* Set interval to check power status in power_interval_mode */
#define MAX_TIMESTAMP 3000 /*Set maximimal timestamp */

 double timekeeper = 0;
 time_t start_timestamp, measure_timestamp;
 int power_mode = 0;
 int power_interval;

/* Modified functions */
int ospf_if_get_output_cost (struct ospf_interface *oi)
{
  /* If all else fails, use default OSPF cost */
  u_int32_t cost = 1;
  u_int32_t bw, refbw;
  FILE *filein;
  struct Powerstatus      {
                                        double      timestamp;
                                        double      power;      };
  struct Powerstatus entry;

  bw = oi->ifp->bandwidth ? oi->ifp->bandwidth : OSPF_DEFAULT_BANDWIDTH;
  refbw = oi->ospf->ref_bandwidth;

  /* Poweraware mode */
  if (power_mode == 1 || power_mode == 2) {
  /* Poweraware mode with time interval */
     if (power_mode == 1) {
                measure_timestamp = time(NULL);
                timekeeper = difftime(measure_timestamp,start_timestamp);
                if ((filein = fopen("/usr/local/etc/powerdata.txt", "r")) == NULL)  printf("File with powerstatus could not be opened.\n");
                else {
                        fscanf(filein, "%lf%lf", &entry.timestamp, &entry.power);
                        while (!feof(filein) && timekeeper > 0) {
                                fscanf(filein, "%lf%lf", &entry.timestamp, &entry.power);
                                if (entry.timestamp >= timekeeper) break;
                                }
                        cost = (101 - entry.power);
                        /* if last timestamp set powerlevel at 0% */
                        if (entry.timestamp == MAX_TIMESTAMP) entry.power = 0;
                        /* if powerlevel is at 0% exit OSPF */
                        if (entry.power == 0) {
                                zlog_info ("Power Awareness Routing: Power level = 0; routing is stopped");
                                exit(EXIT_SUCCESS);
                                }
                        fclose(filein);
                  }
               if (cost < 1)  cost = 1;
               else if (cost > 65535)     cost = 65535;
               }

  /* Poweraware mode with power interval */
     if (power_mode == 2) {
                  if ((filein = fopen("/usr/local/etc/powerdata.txt", "r")) == NULL) printf("File with powerstatus could not be opened.\n");
                  else {
                        fscanf(filein, "%lf%lf", &entry.timestamp, &entry.power);
                        while (!feof(filein) && timekeeper > 0) {
                                        fscanf(filein, "%lf%lf", &entry.timestamp, &entry.power);
                                        if (entry.timestamp==timekeeper) break;
                                        }
                        cost = (101 - entry.power);

                        /* if last timestamp set powerlevel at 0% */
                        if (entry.timestamp == MAX_TIMESTAMP) entry.power = 0;
                        /* if powerlevel is at 0% exit OSPF */
                        if (entry.power == 0) {
                                zlog_info ("Power Awareness Routing: Power level = 0; routing is stopped");
                                exit(EXIT_SUCCESS);
                                }
                        fclose(filein);
                        }
                        if (cost < 1)  cost = 1;
                        else if (cost > 65535) cost = 65535;
                        }
```

```
                /* Normal OSPF (no powerawareness mode) */
                else {
                        if (cost < 1)  cost = 1;
                        else if (cost > 65535) cost = 65535;
                    }
                }
                else {
                 /* A specifed ip ospf cost overrides a calculated one. */
                        if (OSPF_IF_PARAM_CONFIGURED (IF_DEF_PARAMS (oi->ifp), output_cost_cmd) ||
                         OSPF_IF_PARAM_CONFIGURED (oi->params, output_cost_cmd))
                        cost = OSPF_IF_PARAM (oi, output_cost_cmd);
                        /* See if a cost can be calculated from the zebra processes interface bandwidth field. */
                        else            {
                                        cost = (u_int32_t) ((double)refbw / (double)bw + (double)0.5);
                                        if (cost < 1)  cost = 1;
                                        else if (cost > 65535) cost = 65535;
                                }
                }
  return cost;
}

void ospf_if_recalculate_output_cost (struct interface *ifp)
{
  u_int32_t newcost;
  struct route_node *rn;

  for (rn = route_top (IF_OIFS (ifp)); rn; rn = route_next (rn))
    {
      struct ospf_interface *oi;
      if ( (oi = rn->info) == NULL)
                continue;
      newcost = ospf_if_get_output_cost (oi);

      /* Is actual output cost changed? */
    if (oi->output_cost != newcost)
                {
                  oi->output_cost = newcost;
                  zlog_info ("Power Awareness Routing: updating metric cost");
                  printf("New cost = %d.\n", newcost);
                  ospf_router_lsa_timer_add (oi->area);
                }
    }
}

/* New functions */
/* Initialization of Power awareness routing  */

void power_init (void)
{
                FILE *confinp;
                listnode node;

  start_timestamp = time(NULL);
  /*  Open configuration file power.conf to check mode  */
  if ((confinp = fopen("/usr/local/etc/power.conf", "r")) == NULL)   printf("Configuration file power.conf could not be opened.\n");
  else fscanf(confinp, "%d %d", &power_mode, &power_interval);

   /* Poweraware mode with time interval */
  if (power_mode == 1) time_interval_mode();

  /* Poweraware mode with power interval */
  if (power_mode == 2) {
                        measure_timestamp = time(NULL);
                        timekeeper = difftime(measure_timestamp,start_timestamp);
                        for (node = listhead (om->iflist); node; nextnode (node))
                                ospf_if_recalculate_output_cost (getdata (node));
                        power_interval_mode();
                        }
  fclose(confinp);
  return;
}
```

```
/* Power awareness routing with time interval */

void time_interval_mode(void)
{
   listnode node;

        zlog_info ("Power Awareness Routing: checking powerstatus");

        for (node = listhead (om->iflist); node; nextnode (node))
                ospf_if_recalculate_output_cost (getdata (node));
        thread_add_timer (master, time_interval_mode, NULL, power_interval);
        return;
}

/* Power awareness routing with power interval */

void power_interval_mode(void)
{
        struct Powerstatus      {
                                double      timestamp;
                                double      power;
                                        };
        struct Powerstatus entry;
        FILE *in;
        listnode node;

        measure_timestamp = time(NULL);
        timekeeper = difftime(measure_timestamp,start_timestamp);

        zlog_info ("Power Awareness Routing: checking powerstatus");

        if ((in = fopen("/usr/local/etc/powerdata.txt", "r")) == NULL)
                printf("File with powerstatus could not be opened.\n");
        else {
                fscanf(in, "%lf%lf", &entry.timestamp, &entry.power);
                while (!feof(in) && timekeeper > 0) {
                                        fscanf(in, "%lf%lf", &entry.timestamp, &entry.power);
                                        if (entry.timestamp == timekeeper) break;
                                        }
                if (entry.power == 85) {
                                for (node = listhead (om->iflist); node; nextnode (node))
                                        ospf_if_recalculate_output_cost (getdata (node));
                                                                        }
                if (entry.power <= 70)  {
                                for (node = listhead (om->iflist); node; nextnode (node))
                                        ospf_if_recalculate_output_cost (getdata (node));
                                thread_add_timer (master, power_interval_mode, NULL, power_interval);
                                }
                else thread_add_timer (master, power_interval_mode, NULL, POWER_CHECK_INTERVAL);
                fclose(in);
                }
        return;
}
```

Appendix E: User-mode linux script files

Start-script for setting up tap devices and UML-switch (run at host computer):

```
#! /bin/bash
#Run as Root

#Clean up system
ifconfig tap0 down
tunctl -d tap0
killall uml_switch
rm -f 1400?

echo starting routing daemons
uml_switch -unix 14000 14001 -hub &  # connects Nodes A and B
uml_switch -unix 14002 14003 -hub &  # connects Nodes A and C
uml_switch -unix 14004 14005 -hub &  # connects Nodes B and C
uml_switch -unix 14006 14007 -hub &  # connects Nodes B and D
uml_switch -unix 14008 14009 -hub &  # connects Nodes C and E
uml_switch -unix 14010 14011 -hub &  # connects Nodes D and E
uml_switch -unix 14012 14013 -hub &  # connects Nodes B and E

#Load tun module
modprobe tun
#Check rights on /dev/net/tun; if necessary repair link

#Setup tap devices
tunctl -u johnny -t tap0
#Configure tap devices
ifconfig tap0 hw ether ff:0:0:0:0:1
ifconfig tap0 192.168.0.1 netmask 255.255.255.0 up
```

Script for starting Node A:

```
#!/bin/sh
echo starting Node A
linux mem=32M ubd0=fs_node_a,root_fs.md-8.2-full.pristine.20020324 ubd1=swap_ap eth0=tuntap,
tap0,ff:0:0:0:0:1,198.168.0.1 eth1=daemon,ff:0:0:0:1:1,unix,14000,14001 eth2=daemon, ff:0:0:0:1:2,
unix,14002,14003
```

Script for starting Node B:

```
#!/bin/sh
echo starting Node B
linux mem=32M ubd0=fs_node_b,root_fs.md-8.2-full.pristine.20020324 ubd1=swap_b eth0=daemon,
ff:0:0:0:2:0,unix,14000,14001 eth1=daemon,ff:0:0:0:2:1,unix,14004,14005 eth2=daemon,ff:0:0:0:2:2,
unix,14006,14007 eth3=daemon,ff:0:0:0:2:3,unix,14012,14013
```

Script for starting Node C:

```
#!/bin/sh
echo starting Node C
linux mem=32M ubd0=fs_node_c,root_fs.md-8.2-full.pristine.20020324 ubd1=swap_c eth0=daemon,
ff:0:0:0:3:0,unix,14002,14003 eth1=daemon,ff:0:0:0:3:1,unix,14004,14005 eth2=daemon,ff:0:0:0:3:2,
unix,14008,14009
```

Script for starting Node D:

```
#!/bin/sh
echo starting Node D
linux mem=32M ubd0=fs_node_d,root_fs.md-8.2-full.pristine.20020324 ubd1=swap_d eth0=daemon,
ff:0:0:0:4:0,unix,14006,14007 eth1=daemon,ff:0:0:0:4:1,unix,14010,14011
```

Script for starting Node E:

```
#!/bin/sh
echo starting Node E
linux mem=32M ubd0=fs_node_e,root_fs.md-8.2-full.pristine.20020324 ubd1=swap_a eth0=daemon,
ff:0:0:0:5:0,unix,14008,14009 eth1=daemon,ff:0:0:0:5:1,unix,14010,14011 eth2=daemon,ff:0:0:0:5:2,
unix,14012,14013
```

Start script for setting up network devices in node A (run in virtual machine):

```
#!/bin/sh
#Run as Root in virtual machine Node A

#Clean up system
ifconfig eth0 down
ifconfig eth1 down
ifconfig eth2 down
ifconfig eth0 up
ifconfig eth1 up
ifconfig eth2 up

echo Configure network devices
ifconfig eth0 hw ether ff:0:0:0:1:0
ifconfig eth0 192.168.0.1 netmask 255.255.255.0
ifconfig eth1 hw ether ff:0:0:0:1:1
ifconfig eth2 hw ether ff:0:0:0:1:2

#echo Mount Host file system
mount none /mnt/host -t hostfs

echo Start Zebra daemon
zebra -d

echo Start Ospf daemon
ospfd -d
```

Start script for setting up network devices in node B (run in virtual machine):

```
#!/bin/sh
#Run as Root in virtual machine Node B

#Clean up system
ifconfig eth0 down
ifconfig eth1 down
ifconfig eth2 down
ifconfig eth3 down
ifconfig eth0 up
ifconfig eth1 up
ifconfig eth2 up
ifconfig eth3 up
```

```
echo Configure network devices
ifconfig eth0 hw ether ff:0:0:0:2:0
ifconfig eth1 hw ether ff:0:0:0:2:1
ifconfig eth2 hw ether ff:0:0:0:2:2
ifconfig eth3 hw ether ff:0:0:0:2:3

#ifconfig eth0 192.168.1.2 255.255.255.0 up
#ifconfig eth1 192.168.3.1 255.255.255.0 up
#ifconfig eth2 192.168.4.1 255.255.255.0 up

echo Mount Host file system
mount none /mnt/host -t hostfs

echo Start Zebra daemon
zebra -d

echo Start Ospf daemon
ospfd –d
```

Start script for setting up network devices in node C (run in virtual machine):

```
#!/bin/sh
#Run as Root in virtual machine Node C

#Clean up system
ifconfig eth0 down
ifconfig eth1 down
ifconfig eth2 down
ifconfig eth0 up
ifconfig eth1 up
ifconfig eth2 up

echo Configure network devices
ifconfig eth0 hw ether ff:0:0:0:3:0
ifconfig eth1 hw ether ff:0:0:0:3:1
ifconfig eth2 hw ether ff:0:0:0:3:2

echo Mount Host file system
mount none /mnt/host -t hostfs

echo Start Zebra daemon
zebra -d

echo Start Ospf daemon
ospfd –d
```

Start script for setting up network devices in node D (run in virtual machine):

```
#!/bin/sh
#Run as Root in virtual machine Node D

#Clean up system
ifconfig eth0 down
ifconfig eth1 down
ifconfig eth0 up
ifconfig eth1 up

echo Configure network devices
ifconfig eth0 hw ether ff:0:0:0:4:0
ifconfig eth1 hw ether ff:0:0:0:4:1
```

```
echo Mount Host file system
mount none /mnt/host -t hostfs

echo Start Zebra daemon
zebra -d

echo Start Ospf daemon
ospfd –d
```

Start script for setting up network devices in node E (run in virtual machine):

```
#!/bin/sh
#Run as Root in virtual machine Node E

#Clean up system
ifconfig eth0 down
ifconfig eth1 down
ifconfig eth2 down
ifconfig eth0 up
ifconfig eth1 up
ifconfig eth2 up

echo Configure network devices
ifconfig eth0 hw ether ff:0:0:0:5:0
ifconfig eth1 hw ether ff:0:0:0:5:1
ifconfig eth2 hw ether ff:0:0:0:5:2

echo Mount Host file system
mount none /mnt/host -t hostfs

echo Start Zebra daemon
zebra -d

echo Start Ospf daemon
ospfd -d
```

Appendix F: Zebra/OSPFd configuration files

Configuration files for Zebra (zebra.conf) and OSPFd (ospfd.conf).

Node A:

zebra.conf:

```
!
! zebra configuration file
!
!
hostname Node_A
password zebra
enable password zebra
!
!log stout
!
! Interface's description.
!
interface eth0
ip address 192.168.0.1/24
interface eth1
ip address 192.168.1.1/24
interface eth2
ip address 192.168.2.1/24
!
! Static default route.
!
ip route 224.0.0.5/32 127.0.0.1
ip route 224.0.0.6/32 127.0.0.1
ip route 224.0.0.9/32 127.0.0.1
log file /usr/local/etc/zebra.log
```

ospfd.conf:

```
!
! ospfd configuration file
!
hostname ospfd
password zebra
!enable password please-set-at-here
!
! Router
router ospf
ospf router-id 192.168.1.1
network 192.168.1.0/24 area 0
network 192.168.2.0/24 area 0
!
! Interface
!
interface eth1
!
interface eth2
!
log stdout
log file /usr/local/etc/ospf.log
```

Node B:

zebra.conf:

```
!
! zebra configuration file
!
hostname Node_B
password zebra
enable password zebra
!
! Interface's description.
!
interface eth0
ip address 192.168.1.2/24
!
interface eth1
ip address 192.168.3.1/24
!
interface eth2
ip address 192.168.4.1/24
!
interface eth3
ip address 192.168.7.1/24
!
! Static default route.
!
ip route 224.0.0.5/32 127.0.0.1
ip route 224.0.0.6/32 127.0.0.1
ip route 224.0.0.9/32 127.0.0.1
log file /usr/local/etc/zebra.log
```

ospfd.conf:

```
!
! ospfd configuration file
!
!
hostname ospfd
password zebra
!
! Router
router ospf
ospf router-id 192.168.3.1
network 192.168.1.0/24 area 0
network 192.168.3.0/24 area 0
network 192.168.4.0/24 area 0
network 192.168.7.0/24 area 0
!
interface eth0
!
interface eth1
!
interface eth2
!
interface eth3
!
log stdout
log file /usr/local/etc/ospf.log
```

Node C:

zebra.conf:

```
!
! zebra configuration file
!
hostname Node_C
password zebra
enable password zebra
!
! Interface's description.
!
interface eth0
ip address 192.168.2.2/24
interface eth1
ip address 192.168.3.2/24
interface eth2
ip address 192.168.5.1/24
!
! Static default route.
!
ip route 224.0.0.5/32 127.0.0.1
ip route 224.0.0.6/32 127.0.0.1
ip route 224.0.0.9/32 127.0.0.1
log file /usr/local/etc/zebra.log
```

ospfd.conf:

```
!
! ospfd configuration file
!
!
hostname ospfd
password zebra
!
! Router
router ospf
ospf router-id 192.168.5.1
network 192.168.2.0/24 area 0
network 192.168.3.0/24 area 0
network 192.168.5.0/24 area 0
!
! Interface
!
interface eth0
!
interface eth1
!
interface eth2
!
log stdout
log file /usr/local/etc/ospf.log
```

Node D:

zebra.conf:

```
!
! zebra configuration file
!
!
hostname Node_D
password zebra
enable password zebra
!
! Interface's description.
!
interface eth0
ip address 192.168.4.2/24
!
interface eth1
ip address 192.168.6.1/24
!
! Static default route.
!
ip route 224.0.0.5/32 127.0.0.1
ip route 224.0.0.6/32 127.0.0.1
ip route 224.0.0.9/32 127.0.0.1
log file /usr/local/etc/zebra.log
```

ospfd.conf:

```
!
! ospfd configuration file
!
!
hostname ospfd
password zebra
!enable password please-set-at-here
!
! Router
router ospf
ospf router-id 192.168.6.1
network 192.168.4.0/24 area 0
network 192.168.6.0/24 area 0
!
! Interface
!
interface eth0
!
interface eth1
!
log stdout
log file /usr/local/etc/ospf.log
```

Node E:

zebra.conf:

```
!
! zebra configuration file
!
!
hostname Node_E
password zebra
enable password zebra
!
!log stout
!
! Interface's description.
!
interface eth0
ip address 192.168.5.2/24
interface eth1
ip address 192.168.6.2/24
interface eth2
ip address 192.168.7.2/24
!
! Static default route.
!
ip route 224.0.0.5/32 127.0.0.1
ip route 224.0.0.6/32 127.0.0.1
ip route 224.0.0.9/32 127.0.0.1
log file /usr/local/etc/zebra.log
```

ospfd.conf:

```
!
! ospfd configuration file
!
!
hostname ospfd
password zebra
!enable password please-set-at-here
!
! Router
router ospf
ospf router-id 192.168.6.2
network 192.168.5.0/24 area 0
network 192.168.6.0/24 area 0
network 192.168.7.0/24 area 0
!
! Interface
!
interface eth0
!
interface eth1
!
log stdout
log file /usr/local/etc/ospf.log
```

Appendix G: Source code of countlsa program

Program to extract information out of tcpdump output files.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX_STRINGLENGTH 350

void write_LSA_info(void);

FILE *inFilePtr, *outFilePtr;
char *inputstring[MAX_STRINGLENGTH];
char timestampstring[100];
double starttimestamp, timestamp, relative_timestamp;
char *lenptr, *len_string, *lenstr = "len", dummy;
int count_LSA=0, count_LSA_upd=0, count_LSA_ack=0;
int LSA_length;
long count_LSA_length=0;

main(int argc, char *argv[])
{
                        if (argc != 3) {        printf("Usage: countlsa infile outfile\n");
                                                exit(1);
                                        }
                        if ((inFilePtr = fopen(argv[1], "r")) != NULL) {
                                if ((outFilePtr = fopen(argv[2], "w")) != NULL) {
                                /* read line out of file into a string */
                                fgets(inputstring, MAX_STRINGLENGTH, inFilePtr);
                                /* set begintimestamp */
                                strncpy(timestampstring, inputstring, 17);
                                starttimestamp = atof(timestampstring);
                                while (!feof(inFilePtr)) {
                                        if ((strstr(inputstring, "OSPFv2-ls_upd")!= NULL) write_LSA_info();
                                fgets(inputstring, MAX_STRINGLENGTH, inFilePtr);
                                                }
                                }
                                else printf("File \"%s\" could not be opened\n", argv[2]);
                        }
                        else printf("File \"%s\" could not be opened\n", argv[1]);
        fclose(inFilePtr);
        fclose(outFilePtr);
        return 0;
}

void write_LSA_info(void)
{
        /* count Link State Update */
        ++count_LSA;
        /* get timestamp */
        strncpy(timestampstring, inputstring, 17);
        timestamp = atof(timestampstring);
        /* calculate relative timestamp */
        relative_timestamp = timestamp - starttimestamp;
        /* get packet length */
        len_string = strstr(inputstring, lenstr);
        lenptr = strtok(strstr(inputstring, lenstr), ")");
        sscanf(lenptr, "%s%d", &dummy, &LSA_length);
        count_LSA_length = count_LSA_length + (LSA_length*8);
        /* write to file */
        fprintf(outFilePtr,"%lf %d %d %d\n", relative_timestamp, count_LSA, LSA_length, count_LSA_length);

        return;
}
```

Appendix H: Source code of randomize program

Program used to randomize (uniformly distributed) the starting power level of the nodes.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

main()
{
        struct Powerstatus      {
                                int             timestamp;
                                int             power; };
        struct Powerstatus entry;
        FILE *in, *Out;
        int i=0, x;
        int *index1, *index2;
        int maxtime;
        double drandom;
        int random;

        /* Determine max timestamp */
        if ((in = fopen("/usr/local/etc/powerdata.org", "r")) == NULL)
                                printf("File with powerstatus could not be opened.\n");
        else            {
                                fscanf(in, "%d%d", &entry.timestamp, &entry.power);
                                while (!feof(in)) fscanf(in, "%d%d", &entry.timestamp, &entry.power);
                        }
        fclose(in);
        maxtime = entry.timestamp;
        printf("maxtime = %d\n", maxtime);
        index1 = calloc(maxtime,sizeof(double));
        index2 = calloc(maxtime,sizeof(double));

        /* Read powerstatus in */
        if ((in = fopen("/usr/local/etc/powerdata.txt", "r")) == NULL)
                                printf("File with powerstatus could not be opened.\n");
        else {
                                fscanf(in, "%d%d", &entry.timestamp, &entry.power);
                                while (!feof(in)){
                                        index1[i] = entry.power;
                                        fscanf(in, "%d%d", &entry.timestamp, &entry.power);
                                        i = ++i;
                                }
                         }
        fclose(in);
        /* Randomize (uniformly distributed)  the powerindex */
        srand48(time(NULL));
        drandom = drand48()*maxtime;
        random = drandom;
        printf("Random shift = %d\n", random);
        for (i=0;i<=(maxtime);++i)
                        {
                        x = i + random;
                        if (x >= maxtime) index2[x - maxtime]  = index1[i];
                        if (x == maxtime) index2[maxtime]  = 0;
                        else index2[x] = 0;
                        }

        /* Write to new file */
        if ((Out = fopen("/usr/local/etc/powerdata.txt", "w")) == NULL)
                                fprintf(stderr, "File could not be opened.\n");
        else                            {
                                        for (i=0;i<=maxtime; ++i) {
                                                entry.timestamp=i;
                                                entry.power = index2[i];
                                                fprintf(Out,"%d %d\n",entry.timestamp, entry.power);
                                                }
                                        }
        fclose(Out);
        free(index1);
        free(index2);
        return 0;
        }
```