A Security Architecture For IPv6 Enabled Wireless Medical Sensor Networks

by

M. Abdul Alim

B.Sc., University of Rajshahi (Bangladesh), 1996

M.Sc., University of Rajshahi (Bangladesh), 1997

Thesis Submitted In Partial Fulfillment Of
The Requirements For The Degree Of
Master of Science
in
Mathematical, Computer, and Physical Sciences
(Computer Science)

The University of Northern British Columbia

July 2007

Your file Votre référence
ISBN: 978-0-494-48828-7
Our file Notre référence
ISBN: 978-0-494-48828-7

Abstract

We present the design of an IPv6 enabled wireless sensor network based on the IEEE 802.15.4 standard for medical monitoring. We design a routing mechanism for efficient flooding, a hop-by-hop error recovery and congestion control mechanism for reliable packet delivery and a lightweight security architecture for the medical monitoring system. We extend the widely used Extensible Authentication Protocol (EAP) to employ the Generalized Pre-shared Key (GPSK) authentication method with some optimizations for securing the system. We use the 3-party EAP model with the Personal Area Network Coordinator (PAN coordinator) of IEEE 802.15.4 standard as the EAP authenticator for authenticating sensor nodes within the radio range of the PAN coordinator. In order to use EAP authentication for a sensor node several hops away from the PAN coordinator, we define a new role (relay authenticator) for its coordinator which tunnels EAP messages to the PAN coordinator securely. We define EAP message encapsulation for IEEE 802.15.4 networks and a key hierarchy for the security architecture. We have simulated the system and shown that EAP based authentication is feasible in wireless sensor networks.

Contents

List of Tables

List of Figures

Acknowledgments

I would like to give my deepest thanks to my supervisors Prof. Behcet Sarikaya and Prof. Liang Chen, who influenced me most during my years at the University of Northern British Columbia. Throughout my two year's studying and researching, they gave me effective guidance with their knowledgeable suggestions. From them, I learnt how to develop new ideas from the papers, how to analyze and simulate network protocols, and how to write a well structured paper. I feel grateful to them for any future success I may have.

I also want to thank to Dr. Siamak Rezaei and Dr. Moustafa Mohamed for their suggestions and comments on this thesis. These valuable comments helped a lot to complete this thesis.

Finally I will give my deep appreciation and thanks to my parents Maser Ali and Abiron Nesa and to my wife Melava Sarmin, for sharing their unconditional love with me and giving me the courage for pursuing my research goals at UNBC.

Chapter 1

Introduction

Wireless sensor networking (WSN) is one of the most exciting and challenging research areas of recent time. A number of independent low-cost, low-power, battery-operated nodes, communicating wirelessly over limited frequency and bandwidth constitutes a wireless sensor network [1]. These sensor nodes run TinyOS [2], an event-driven operating system for networked applications in wireless embedded systems. The deployment of wireless sensor networks is becoming more common in a wide variety of applications for collecting and disseminating sensitive and important information due to their low cost, deployment flexibility and mobility support. Some of the most common applications include environmental and habitats monitoring, medical monitoring, disaster management, infrastructure and seismic monitoring, traffic control, industrial plant monitoring etc. Fig. 1.1 shows a typical sensor network deployment architecture for monitoring applications such as patient monitoring. The increasing number of applications and resource constraints of WSNs emerge a number of research issues such as power management, network discovery, control and routing, reliable transport, integration to the Internet, collaborative signal and information processing, tasking and querying, and security are all currently under research.

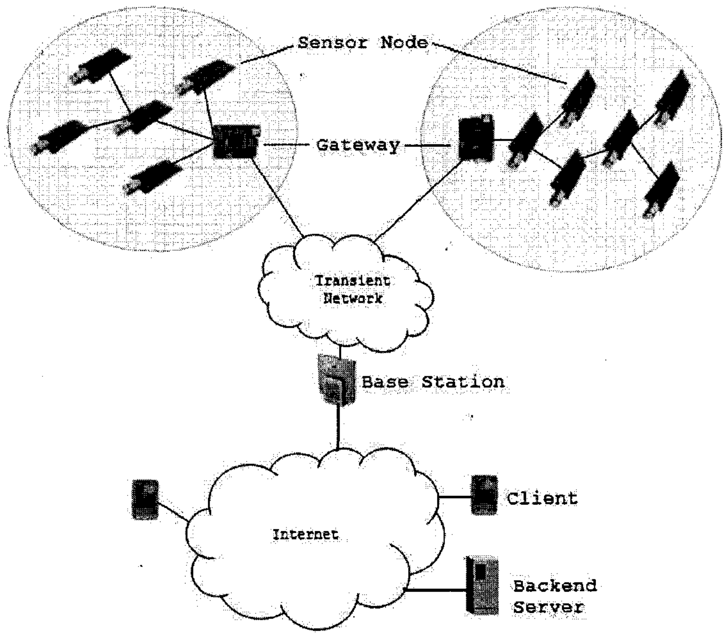


Figure 1.1: Typical Sensor Network Architecture

1.1 WSN for Medical Monitoring

Medical monitoring using wireless sensor network has the potential to offer better quality of healthcare services to an increased number of patients [3] [4]. Wireless patient monitoring involves measurement and digitization of vital signs (e.g. BP, EKG, respiration rate, pulse, and oxygen saturation), transmission of packets over wireless networks, and delivery of complete information to one of more healthcare professionals. For the last few years, a number of research projects has been exploring wireless sensor networks for medical services like continuous patient monitoring and providing emergency medical care in disasters. For example, the wireless sensor based patient monitoring system, Ayushman [5], collects real-time health information in diverse scenarios, from home based monitoring to disaster relief. CodeBlue [6] extends wireless sensor networks applications for emergency medical services in disasters. In [7], we have designed a wireless medical sensor network for recognizing mental states based on Electroencephalograms (EEG) recordings.

Wireless EEGs consist of EEG sensors, electrodes, digitizing equipment with wireless interface providing connection to a personal computer or to a Personal Digital Assistant (PDA). EEG sensors are connected to well-defined parts of the human scalp (see Fig. 1.2) to collect brain signals. Brain signals from each sensor (channel) are digitized at 250 Hz with 16 bits implying a data rate of 4 kbps per channel. In a recent clinical trial, it has been proved that four-channel wireless EEG quality is acceptable and feasible to perform in the hospital emergency rooms [8]. The IEEE 802.15.4 standard is designed for applications that require low data rate, low power consumption, low cost, self-organization, and flexible topologies which makes it most suitable communications protocol for use in wireless sensor networking [9]. The IEEE 802.15.4 radio supports a minimum data rate of 20 kbps and a maximum data rate of 250 kbps and can be used to transport EEG data.

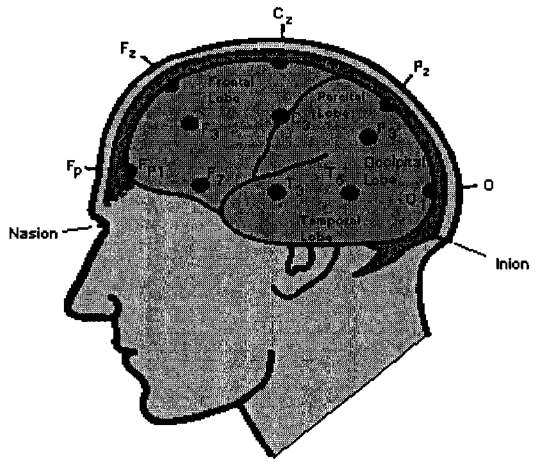


Figure 1.2: International 10-20 System of EEG Electrode Placement

In [7], we have designed a combined software and hardware platform for medical sensor networks like the CodeBlue [10]. We have designed EEG sensors with Tmote Sky [11] motes, which employ the IEEE 802.15.4 radio and 12-bit analog to digital conversion hardware. EEG data, gathered and posted online by Z. Keirn [12], have been stored in Tmote's flash memory to be used as an EEG sensor. The system consists of a number of EEG sensor nodes, a base station, and a backend server. We have used a Stargate [13] single-board microcomputer with a Tmote as the base station; Stargates have the processing power and memory capabilities of a typical

PDA.

A spanning tree-based multi-hop routing protocol is used to disseminate queries to sensor nodes and sensor readings to the base station. A reliable data transport protocol is used to disseminate queries to sensor nodes and sensor readings to the base station reliably [14]. The system uses a simple query processor similar to the CodeBlue's CBQ, that takes queries of the form $\langle S, C, p \rangle$, where S is the set of subjects that should report data for this query, C is the count of the total number of samples and can be left unspecified to obtain the samples continuously and p is the filter predicate. We have added additional queries to the system by incorporating machine learning module for mental state recognition. The query predicates can specify the mental task being performed by a subject, for example $\langle S, , p = T \rangle$, where T is a mental state. For this type of query, the query processor sends a query to the motes for subject S and the data received from motes are then forwarded to the machine learning subsystem for recognition.

EEG signals are usually contaminated by eye movements, eye blinks, and muscular activities and therefore, we need to remove these artifacts from samples collected by electrodes before analyzing EEG signals. We have used the Independent Component Analysis (ICA) [15] in order to remove artifacts from the EEG data collected from sensors. The ICA method is based on the assumptions that the time series recorded on the scalp are spatially stable mixtures of the activities of temporally independent cerebral and artifactual sources, that the summation of potentials arising from different parts of the brain, scalp, and body is linear at the electrodes, and that propagation delays from the sources to the electrodes are negligible. We have extracted frequency components (i.e. features) from the clean EEG data after removing artifacts.

Two-second long segments of the time signal which overlap one second have used to compute the power spectrum with a short time Fourier transformation. This gives one feature per second and frequency band of 1/2 Hz. We have used a modified Mel Frequency Cepstral Coefficient (MFCC) [16], which has linear frequency spacing below 80Hz and logarithmic spacing above 80Hz. Feature values for different frequency bands might have different ranges and they might fluctuate differently. However, large fluctuations do not necessarily mean a large importance for classification [17] and therefore a simple normalization technique is used on training and test data sets. On training and test data sets, mean and variance are calculated for each electrode and each frequency band, which are then used for mean subtraction and variance normalization.

We have implemented a Bayesian network [18] machine learning algorithm to recognize mental states from EEG features, which calculates the probabilities of variations in EEG signals by using mixture of Gaussian family of distributions. The classifier calculates the likelihood of each input signal to categorize it into a class of mental state, where EEG features are the evidence and the calculated probabilities form the hypothesis for classification, for further information see [7].

1.2 Sensor Network Hardware

Wireless sensor devices called *motes*, consist of a Central Processing Unit (CPU), low-power radio, a power source, optional sensing elements and a modest amount of local storage in several cubic inch package. For example, the Berkeley Mica2 motes use an 8MHz 8-bit Atmel ATMEGA128L CPU with 128 KB of instruction memory, 4 KB of random access memory (RAM) for data, and 512 KB of flash memory [19]. Mica2 offers a low-power single-chip radio from Chipcon (CC1000), that operates at 433 MHz or 916 MHz and delivers up to 76.8 kbps application bandwidth with an indoor range of approximately 20 to 30 meters. Recently, Moteiv has brought the next-generation mote platform, Tmote Sky, for extremely low-power and high data-rate sensor network applications [11]. Tmote Sky uses 8MHz Texas Instruments 16-bit reduced instruction set computer (RISC) processor (MSP430) with on-chip 10 KB of RAM, 48 KB of flash memory, the IEEE 802.15.4 radio, and an integrated on-board antenna providing 250 kbps data rate at 2.4 GHz with up to 125 meter range.

The resource scarcity of wireless sensor networks requires minimizing resource usage while providing better quality services. Communications infrastructure plays an important role in the success of wireless sensor networks because it affects the network architecture, data rates, network size, span, power management and security protocols. Bluetooth [20] (IEEE 802.15.1) is the first well-known standard designed low data rate applications. Bluetooth provides efficient channel hopping and high throughputs but requires expensive synchronization. During *piconet* formation, the master node must be in high-power scanning mode and Bluetooth radio consumes hundreds of milliwatts of power [20]. The complexity of Bluetooth makes it expensive and requires relatively high operating power and therefore unsuitable for use with low-cost and low-power wireless sensor networking applications.

1.2.1 The IEEE 802.15.4 Standard

The IEEE 802.15.4 Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPAN) [9] defines wireless personal area networks (WPANs) for small, inexpensive, power-efficient, short-range wireless devices. The main objectives of an LR-WPAN are ease of installation without or little infrastructure, reliable data transfer, short-range operation, and extremely low cost and reasonable battery life, whilst maintaining a simple and flexible protocol. The IEEE 802.15.4 supports simple devices that consume minimal power and typically operate in the Personal Operating Space (POS) of 10 meters or less. LR-WPAN supports one-hop *star topology* and multi-hop *peer-to-peer topology* (see Fig. 1.3) to meet the requirements of different applications.

In the star topology, the communication is established between devices and a single central controller, called the PAN coordinator. The PAN coordinator is the primary controller of the network. The peer-to-peer topology also has a PAN coordinator, however devices may communicate with any other device as long as they are in range of one another. Peer-to-peer topology allows ad-hoc, self-organizing and self-healing network topology such as mesh networking topology and requires mesh routing functionalities to forward messages from one device to another inside the PAN.

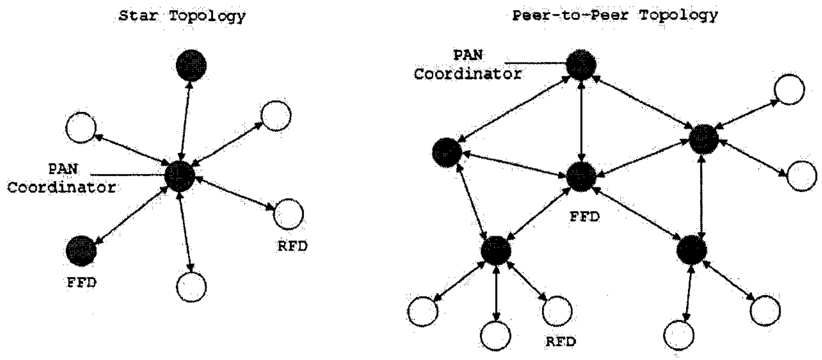


Figure 1.3: IEEE 802.15.4 Star and Peer-to-Peer Topology

The IEEE 802.15.4 defines two types of devices that may participate in an LR-WPAN: *full functionality devices* (FFDs) and *reduced functionality devices* (RFDs). FFDs can communicate with any other neighboring node in the PAN and may become a network coordinator, whereas RFDs can communicate only with its coordinator and therefore must associate with a coordinator in both topologies.

Each time a coordinator wakes up, it transmits a network beacon frame to inform other nodes that it is awake, the super-frame duration, and the next time a beacon will be sent, known as the beacon duration. After receiving a beacon from a coordinator, other devices may synchronize to the schedule of the coordinator and request to associate. Following notification from the coordinator that the association was successful, associated nodes may send data to the coordinator after its beacon. In the other direction, coordinators advertise data they have for associated nodes in the payload of their beacon message. Nodes may then request data from the coordinator, who then sends the data. In the peer-to-peer topology, FFDs must listen to beacons from other nodes to communicate with them. A device in an LR-WPAN can use either a 64-bit IEEE extended address (EUI-64) or a 16-bit short address assigned during the association procedure.

The peer-to-peer topology allows to form a larger network and a special case of

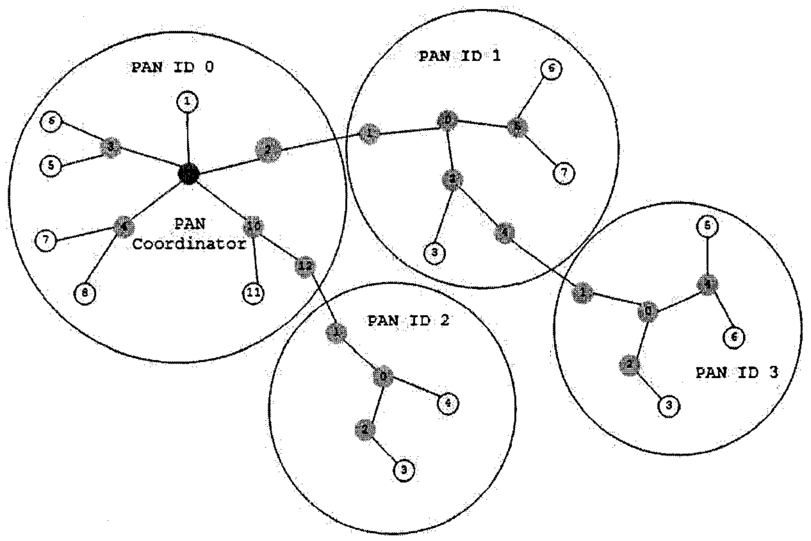


Figure 1.4: IEEE 802.15.4 Cluster Tree Topology

it is a *cluster tree* network where non-leaf nodes are FFDs. An RFD connects to a cluster tree network as a leaf device at the end of a branch, because RFDs do not allow other devices to associate. Any of the FFDs may act as a coordinator and provide synchronization services to other devices or other coordinators. Only one of these coordinators can be the overall PAN coordinator, which may have greater computational resources than any other device(s) in the PAN. If the PAN coordinator permits the device to join, it adds the new device as a child device in its neighbor list. Then the newly joined device adds the PAN coordinator as its parent in its neighbor list and begin transmitting periodic beacons; other candidate devices may then join the network at that device. The simplest form of a cluster tree network is a single cluster network, but larger networks are possible by forming a mesh of multiple neighboring clusters (see Fig. 1.4). In cluster tree networks, each PAN is identified by using a 16-bit PAN ID.

The IEEE 802.15.4 Security

The IEEE 802.15.4 standard specifies a number of security suites based on symmetric-key cryptography to provide data confidentiality, data authenticity, and replay protection at the link layer [9]. An application needs to set its security requirements by setting the appropriate parameters into the radio stack. If the application does not set any parameters then, by default, there is no security enabled.

Security Level	**Description**
None	No Security
MIC-32	4-octet Message Integrity Code (MIC)
MIC-64	8-octet Message Integrity Code
MIC-128	16-octet Message Integrity Code
ENC	Encryption Only
ENC-MIC-32	Encryption with 4-octet MIC
ENC-MIC-64	Encryption with 8-octet MIC
ENC-MIC-128	Encryption with 16-octet MIC

Table 1.1: IEEE 802.15.4 MAC Security

The IEEE 802.15.4 defines 8 different levels of security to provide varying levels of data authenticity and for optional data confidentiality on a frame-by-frame basis. The first of these provides no security, the next three are for message integrity protection, followed by encryption only, and finally the last three are for encryption and authentication (see Table 1.1). All of the above security alternatives but the first always provide replay protection. However, it does not define authentication and key exchange mechanism due to the large number of targeted application areas.

1.3 Contributions

This thesis makes a number of contributions toward the integration of IPv6 into low-power sensor devices and securing wireless medical sensor networks:

- Developed an efficient broadcast tree construction protocol and adapted on-demand routing protocols to minimize the number of retransmissions required for flooding a packet inside the WPAN. A hop-by-hop error recovery mechanism is employed for reliable frame delivery inside the WPAN to detect packet loss as early as possible and hence to minimize the communications overhead.

- Designed a security architecture for wireless medical sensor networks on top of IEEE 802.15.4 security primitives by defining authentication and key management mechanisms. We also define the key hierarchy for the proposed security architecture.

- Developed an authentication framework for wireless sensor networks to protect unauthorized devices to access the network. We extend the 3-party EAP authentication for multi-hop WPAN architecture and optimize the GPSK EAP method for low-power wireless sensor networks.

1.4 Outline

The remainder of this thesis is organized into 5 chapters exploring the integration of IPv6 in low-power wireless networks, efficient routing protocols for the wireless mesh, and optimal security solution for wireless sensor networks.

Chapter 2 discusses the integration of IPv6 into low-power wireless sensor networks. The subjects covered in this chapter include required IPv6 functionalities and challenges in implementing these features in low-power wireless sensor networks. Then how these features can be incorporated into low-power wireless networks are discussed in detail.

Chapter 3 presents mesh routing and reliable data transport protocols for low-power wireless networks. First, we introduce the optimized broadcast tree construction algorithm followed by on-demand mesh routing protocol on top of the broadcast tree. We also present some simulation results for evaluating the performance of our protocols. Second, we present an efficient implementation of reliable data transport protocol for wireless sensor networks.

In Chapter 4, we discuss security architectures for wireless sensor networks. First, we present the security threats, requirements, and challenges of implementing security solution for low-power wireless sensor networks. Then we discuss the security primitives that can be implemented on resource-constrained sensor devices and security services available in IEEE 802.15.4 devices. Finally, we present an overview of the security architecture that has been proposed for low-power wireless sensor networks.

In Chapter 5, we present an authentication and key management framework for low-power wireless sensor networks. First, we present the architectural components of the security framework, an EAP encapsulation for IEEE 802.15.4 networks, and a key hierarchy. Then we discuss the authentication and key management procedure for devices joining the network and finally, we present simulation results for performance evaluation.

Finally, Chapter 6 summarizes the accomplishments of this study and relates them to the wider field. Some improvements are suggested as future avenues of research.

Chapter 2

Integrating IPv6 into Wireless Sensor Networks

The increasing applications of wireless sensor networks necessitates the connectivity of WSNs to the Internet. Although it is possible to connect WSNs to the Internet by using some sort of proxies (e.g. the *serial forwarder* of TinyOS [2]), but they do not provide seamless access to sensors and have the problem of single point of failure. In addition, proxies violate the end-to-end communication paradigm of the Internet, accompanied by problems similar to Network Address Translation (NAT). Furthermore deploying a non-IP communication approach means the need for developing new control and communication protocols that were already standardized for IP communication [21].

Many sensor network applications require seamless access to the sensors and pervasive monitoring capabilities. However, sensor networking has raised diverse challenges to researchers due to its wireless nature, node density, limited resources (e.g. battery power, processing capacity, and storage memory), low reliability of the nodes, and mobile distributed architecture different from those of classical mobile ad-hoc networks (MANET) [22]. Zuniga et al., [23] points that the implementation of IP stack in WSNs could not be viable due to computational limitations and low memory resources of sensor nodes. In contrast, the author in [24] proves the opposite by constructing a model that permits full TCP/IP stack, designated as μIP, to be executed in 8-bit micro-controllers. In [21], authors investigated the required IPv6 functions and protocols for IP-enabled sensor networks and designed a system for disaster management.

The Internet Engineering Task Force (IETF) has formed a working group, IPv6 over Low power WPAN (6LoWPAN) [25], to identify challenges and goals of incorporating IPv6 into LR-WPAN [26] and to define IPv6 functionalities for LR-WPAN [25] [27]. A number of companies (e.g. Sensinode Ltd. [28] and Arch Rock Corp. [29]) already manufacturing sensor devices with IPv6 support. In [30], David E. Culler presents the technological advancement toward implementation of IPv6 in low-power wireless devices of all kinds and their connectivity to the Internet like Wi-Fi.

2.1 Requirements and Challenges

Wireless sensor networks commonly use the IEEE 802.15.4 PHY and MAC layer protocols and therefore IPv6 operation has to be specified for it. The IEEE 802.15.4 MAC layer leaves only 81 octets after link-layer security for upper layer, which is far below the minimum length of 1280 octets of an IPv6 packet[31]. Furthermore the IPv6 header without additional extension headers is of 40 octets and the user datagram protocol (UDP) uses 8-octet and the transmission control protocol (TCP) uses 20-octet header. Therefore, a *fragmentation and reassembly adaptation layer* must be provided at the layer below the IP layer. It is also important to use *header compression* for efficient transmission of IPv6 packets over IEEE 802.15.4 networks.

To increase the size of the sensor network, sensor networks are expected to form a multi-hop ad hoc network in which IP packets need to be routed to their destinations. Furthermore, sensor devices could be mobile, that is, nodes may enter or leave the network at any time or just move around. Therefore, an ad hoc on-demand routing protocol is required to create a virtual single-link broadcast network below the IPv6 layer, which is important for applying standard IPv6 functionalities like neighbor discovery and stateless address autoconfiguration.

To communicate with other devices using IP stack, a sensor node needs to be configured with an IP address. Since sensor networks may consist of hundreds of nodes and they are deployed in an ad hoc fashion, stateless address autoconfiguration [32] is necessary to reduce the configuration overhead on hosts.

Integrating wireless sensor networks into the Internet requires extra attention to security mechanisms including confidentiality, integrity protection, authentication, authorization and access control, intrusion detection, denial of service prevention, etc. The Internet Security protocol (IPSec) [33] is mandatory to run IPv6, however power constraints and limited processing capability of sensor devices make IPSec computation intensive and the Internet Key Exchange protocol (IKE) [34] communication intensive. Thus IP enabled sensor networks need to define efficient keying methods to reduce communications overhead by minimizing per packet overhead and of course the number of signaling messages.

2.2 IPv6 Adaptation for IEEE 802.15.4 Networks

The IETF 6LoWPAN working group defines the LoWPAN encapsulation of IPv6 packets for the IEEE 802.15.4 MAC protocol data unit. IPv6 packets to be transported over IEEE 802.15.4 networks are prefixed by an encapsulation header stack. Each header in the header stack contains a header type (e.g. Mesh, Broadcast, Fragmentation) followed zero or more header fields. The *dispatch header* identifies a LoWPAN packet and the *Dispatch* field is used to differentiate subsequent fields. The *mesh header* is used by the mesh routing protocol for forwarding a frame toward the destination in the PAN. The *fragmentation header* is used for transporting large IP packets using fragmentation and reassembly mechanism.

2.2.1 Fragmentation and Reassembly

If an IPv6 packet does not fit within a single IEEE 802.15.4 MAC frame, the 6LoWPAN adaptation layer breaks the packet into link fragments and uses the fragmentation header. The first fragment contains the fragment header with the first three bits set to 110 followed by the *datagram size* (11 bits) followed by the *datagram tag* (16 bits). The subsequent fragments contains fragmentation header with the first three bits set to 111 followed by the *datagram size*, followed by the *datagram tag* followed by *datagram offset* (8 bits). The *datagram tag* and the *datagram size* must be the same for all fragments of an IPv6 packet. Fig. 2.1 shows the fragmentation header formats.

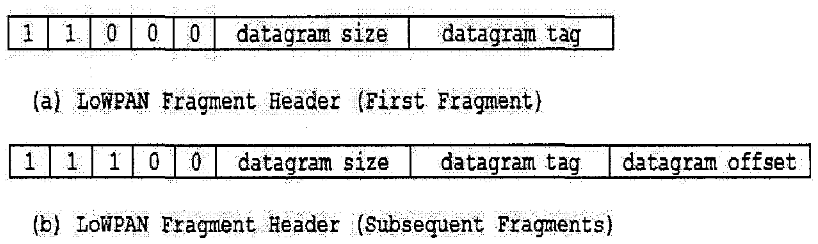

1 1 0 0 0 datagram size datagram tag
(a) LoWPAN Fragment Header (First Fragment)
1 1 1 0 0 datagram size datagram tag datagram offset
(b) LoWPAN Fragment Header (Subsequent Fragments)

Figure 2.1: LoWPAN IP Fragmentation

A receiver node uses the source address, the destination address, the datagram size and the datagram tag to identify all the fragments that belong to a given IPv6 packet. When a node receives the first fragment, it starts the reassembly process and starts reassembly timer. The receiver buffers received fragments and constructs the IPv6 packet by concatenating fragments based on the *datagram offset* field for each fragment. The size of the reassembled packet must be equal to the *datagram size.* If any fragment overlaps with any other fragment, or the size of the reassembled packet does not comply with the *datagram size*, or reassembly timer expires, the receiver discards the packet.

2.2.2 Header Compression

Communication protocols use packet headers to help applications to communicate over large distances connected by multiple links or hops in the network. However, as long as the applications are communicating most of this information carried in packet headers remains the same or changes in specific patterns. By observing the fields that remain constant or change in specific patterns it is possible either not to send them in each packet or to represent them in a smaller number of bits, which is described as *header compression* (HC). The process of header compression uses the concept of flow context, which is a collection of information about field values and change patterns of field values in the packet header. This context is formed on the compressor and the decompressor side for each packet flow. The first few packets of a newly identified flow are used to build the context on both sides and are sent without compression.

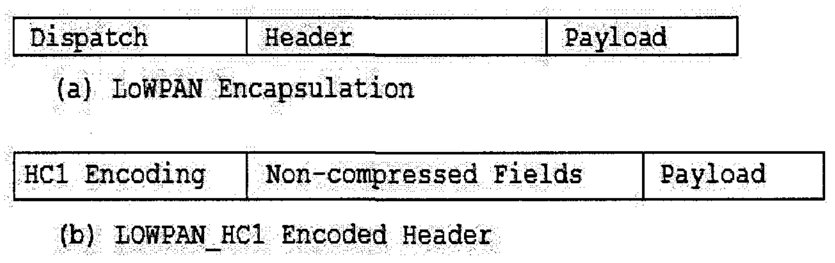


Figure 2.2: LoWPAN Header Compression

The IEEE 802.15.4 association procedure gathers some state information and can be used to compress the following common IPv6 header fields onset: version, link-local IPv6 source and destination addresses, the lower order 64 bits of IPv6 address which can be generated from the link layer addresses, the packet length which can be computed either from the *Frame Length* field of IEEE 802.15.4 MAC header or from the *datagram size* field of the fragment header, *Traffic Class* and *Flow Label* which are zero, and *Next Header* which is either User Datagram Protocol (UDP), Internet Control and Management Protocol (ICMP) or Transmission Control Protocol (TCP). Only the *Hop Limit* field of the IPv6 header needs to be sent always in full. Thus common IPv6 header can be compressed to 2 octets, 1 octet for the *HC1 Encoding* and 1 octet for the *Hop Limit*, instead of 40 octets. Such a packet is compressible via the LOWPAN_HC1 format by using a header dispatch type of *LOWPAN_HC1* followed by a dispatch header encoded using the *HC1 Encoding* as shown in Fig. 2.2. This header may be preceded by a fragmentation header, which may be preceded by a mesh header. Table **2.1** presents the *Dispatch* values defined by the 6LoWPAN adaptation layer.

Dispatch	**Dispatch Header**
NALP	Not a LoWPAN frame
IPv6	Non-compressed IPv6 header
LOWPAN_HC1	HC1 encoded IPv6 header
LOWPAN_BC0	LoWPAN broadcast header
ESC	More Dispatch

Table 2.1: LoWPAN Dispatch Values

The *HC1 Encoding* uses 1 octet for encoding the IPv6 header. The IPv6 address fields are encoded by *HC1 Encoding* with 4 bits, source address with bits 0 and 1 and destination address with bits 2 and 3, and are interpreted as shown in Table 2.2, where PI means prefix carried in-line, PC means prefix compressed, II means interface identifier (IID) carried in-line, and IC means IID compressed.

Traffic Class and *Flow Label* are encoded using bit 4; a 0 indicates that they are not compressed and a 1 indicates that both of them are zero. The *Next Header* is encoded using bits 5 and 6 as shown in Table **2.3.** A 0 for the bit 7 of the *HC1 Encoding* indicates no more header compression bits and a 1 indicates that the *HC1 Encoding* is immediately followed by more header compression bits according to *HC2 Encoding* format, for more information see [25]. Non-compressed IPv6 header fields can be sent instead of the entire IPv6 header, which offer different degrees of header

bits 0-1/2-3	**Encoding**	
00	PI	II
01	PI	IC
10	PC	II
11	PC	IC

Table 2.2: LoWPAN HC1 Encoding for Address Fields

bits 5-6	**Meaning**
00	Non-compressed
01	UDP
10	ICMP
11	TCP

Table 2.3: HC1 Encoding for Next Header

compression.

2.2.3 Mesh Routing

Wireless sensor networks are usually multi-hop networks, therefore IP packets (e.g. for neighbor discovery) to be delivered inside the WPAN need to use mesh routing. However, the IEEE 802.15.4 specification [9] does not define such capability. The IETF 6LoWPAN working group defines mesh delivery header (see Fig. 2.3) to enable mesh routing in IEEE 802.15.4 networks, so that FFDs can employ some mesh routing protocol to populate their routing tables.

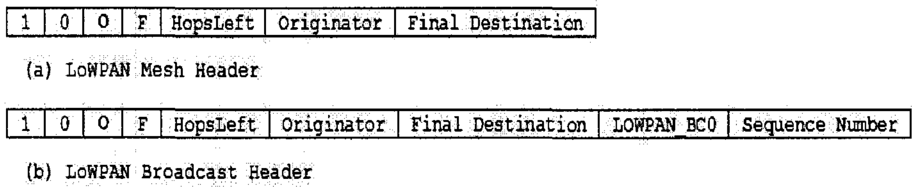


Figure 2.3: LoWPAN Mesh Header Formats

The mesh type is indicated by the first two bits (10) of the header and two bit fields, O and F, are used to indicate address types for the originator node and the final destination node. A 0 for O or F indicates that the address is an IEEE 64-bit extended address (EUI-64) and a 1 indicates that the address is a 16-bit short address. The $HopsLeft$ is a 4-bit field to be used as hops count for route discovery or data forwarding. Each node decrements the $HopsLeft$ before sending a mesh frame towards its next hop and if $HopsLeft$ reaches zero, the frame is discarded. The *Originator* is the link-layer address of the originator of the frame and the *Final Destination* is the link-layer address of the intended destination of the frame.

When a node has a packet to send to a destination that is not in its radio range, it includes a mesh delivery header with the *Originator* set to its own link-layer address, and the *Final Destination* set to the packet's ultimate destination address. It sets the source address in the IEEE 802.15.4 header to its own link-layer address, puts the forwarder's link-layer address in the IEEE 802.15.4 header's destination address field and transmits the packet. When a node receives a mesh frame, it looks at the mesh header's *Final Destination* field to determine the actual destination. If the node itself is the ultimate destination, it consumes the packet otherwise it decrements the $HopsLeft$ by one, and if the result is zero, discards the packet. If $HopsLeft$ is greater than zero, the node employs mesh routing to determine the next hop towards the ultimate destination and puts that address in the destination address field of the IEEE 802.15.4 header. Finally, the node changes the source address in the IEEE 802.15.4 header to its own link-layer address and transmits the packet.

For IPv6 broadcast and multicast packets, a broadcast header is added immediately following the mesh header. The broadcast header consists of a $LOWPAN\_BC0$ dispatch followed by an 8-bit *Sequence Number* field which is used to detect duplicate packets (see Fig. 2.3).

2.3 Network Architecture and Neighbor Discovery

As discussed in Chapter 1, IEEE 802.15.4 networks support star, peer-to-peer and cluster-tree topologies. In all deployment architectures, the WPAN is controlled by a special node called *PAN coordinator*. Mesh routing in peer-to-peer and cluster-tree topologies make all nodes on-link neighbors to the PAN coordinator virtually. We assume that a WPAN maps to a specific IPv6 link and that the PAN coordinator acts as the default IPv6 router for all nodes of the WPAN.

The IEEE 802.15.4 standard defines two addressing modes, the IEEE 64-bit extended address (EUI-64) and 16-bit short address for WPAN devices. A 16-bit short address can be used after a successful association with a coordinator and is unique within the WPAN. An EUI-64 address can be mapped to an IPv6 address trivially, since these addresses never change for a given node. To use a 16-bit short address as a link-layer address and to generate Interface ID (IID) from it, IPv6 needs to take extra caution for stateless address autoconfiguration (SAA), neighbor unreachability detection (NUD) and duplicate address detection (DAD) because it can be changed over time. Since the IEEE 802.15.4 devices do not support link-layer multicast, IPv6 multicast packets will be carried as link-layer broadcast frames inside WPANs.

2.3.1 Bootstrapping

An IPv6 node requires at least the *default router*'s address and IPv6 prefixes in order to communicate with others using Internet Protocol. The process of obtaining this information is called *bootstrapping.* This could be done either manually or via autoconfiguration without user intervention. Since sensor networks can have hundreds of nodes, it is almost impossible to configure them manually. Furthermore, since sensor networks are expected to be deployed in ad hoc manner and nodes may be mobile, it is important to use autoconfiguration in network partition or merge events due to mobility of nodes.

A node requires a link-local IP address to communicate with on-link nodes, but link-local addresses cannot be used to communicate with off-link nodes. In order to communicate with nodes in the Internet, a sensor node has to perform the gateway discovery to obtain the address of the default router. Sometimes additional configuration information may be required to access specific services (e.g. the IP addresses of DNS servers) and can be configured automatically using service discovery.

2.3.2 Router and Prefix Discovery

IPv6 uses Neighbor Discovery Protocol (NDP) [32] to find neighboring routers and to retrieve prefixes and other configuration parameters related to address autoconfiguration. Routers send *router advertisement* messages that indicate whether the sender is willing to be a default router. Router advertisement messages also contain prefix information options that list the set of prefixes to configure routable IP addresses. To use IP in IEEE 802.15.4 networks, the PAN coordinator acts as the default router and may periodically send router advertisement messages in LoWPAN encapsulated mesh frames. These messages are delivered to the network using mesh routing algorithm.

2.3.3 Stateless Address Autoconfiguration

IPv6 uses NDP and Stateless Address Autoconfiguration (SAA) [35] to configure IPv6 addresses automatically. IPv6 nodes learn about IPv6 routers and prefixes by exchanging *router solicitation* and *router advertisement* multicast messages. In [27], authors proposed to use unicast *router solicitation* message to obtain the IPv6 address of the access router and the prefix instead of periodic multicast. A node generates the Interface Identifier (IID) from the MAC address and append to the prefix obtained from the access router to configure its routable IPv6 address. The IPv6 link-local address for an IEEE 802.15.4 interface is formed by appending the IID to the prefix FE80::/64 as shown in Fig. 2.4.

If a node uses the IEEE 64-bit extended MAC address (EUI-64), the IID is derived from the EUI-64 according to the *IPv6 over Ethernet* specification [36]. If a 16-bit short MAC address is used, a *pseudo 48-bit address* is derived first by appending 16 zero bits to the 16-bit PAN ID and then the 16-bit short address, i.e. *48-bit pseudo address = 16-bit PAN ID + 16 zero bits + 16-bit short address.* The IID is then derived from this 48-bit pseudo address as per the *IPv6 over Ethernet* specification.

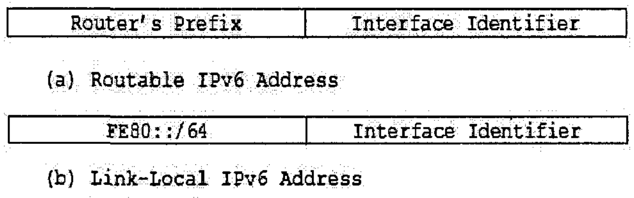


Figure 2.4: IPv6 Addresses Autoconfiguration

Duplicate Address Detection

A node verifies the uniqueness of a newly generated IPv6 address in the network using Duplicate Address Detection (DAD). A node sends a Neighbor Solicitation (NS) message via solicited node multicast and a node that detects an address conflict replies with a Neighbor Advertisement (NA) message. However, DAD could be optimized for IEEE 802.15.4 networks by using proxy DAD. In proxy DAD, when a node configures its IPv6 address using SAA, it unicasts an NS message to the PAN coordinator, allowing the PAN coordinator to keep track with IP to MAC address bindings of all sensor nodes of the WPAN. Upon receipt of an NS message, the PAN coordinator searches its DAD cache for a duplicate address. If the soliciting address is found duplicate, the PAN coordinator unicasts an NA message to the soliciting node on behalf of the node that owns the address.

Address Resolution

Address resolution is the process through which a node determines the link-layer address of a neighbor given only its IP address. Address resolution is performed only on addresses that are determined to be on-link and for which the sender does not know the corresponding link-layer address. When a node has a unicast packet to send to a neighbor, but does not know the neighbor's link-layer address, it performs address resolution by transmitting an NS message to the solicited-node multicast address corresponding to the target address. Upon receipt of an NS message, the node corresponding to the IP address sends an NA message with its link-layer address [32]. The address resolution procedure for mapping IPv6 unicast addresses into IEEE 802.15.4 link-layer addresses in LoWPAN with mesh routing can use the same process.

IPv6 multicast addresses can be mapped to 16-bit addresses to be used in IEEE 802.15.4 networks as shown in Fig. 2.5. The initial 3 bits of the first byte are set to 100 according the 16-bit address format for multicast addresses as defined in [25]. The next 5 bits of the first byte come from the last five bits of 15th octet (DST[14]) and the second byte is set to the 16th octet (DST[15]) of the IPv6 address (DST[0..15]).

For sensor networks, IPv6 address resolution can be optimized by limiting packet exchange between the requesting node and the PAN coordinator to link-layer unicast.

1	0	0	DST[14]*	DST[15]

Figure 2.5: IPv6 Multicast Address Mapping

Neighbor Unreachability Detection

Communication to or through a neighbor may fail for numerous reasons at any time, including hardware failure, node movements, etc. Neighbor Unreachability Detection (NUD) is used for all paths between hosts and neighboring nodes, including host-to-host, host-to-router, and router-to-host communication [32]. NUD is performed only for neighbors to which unicast packets are sent and not for multicast addresses. When a path to a neighbor appears to be failing, the specific recovery procedure depends on how the neighbor is being used, for example, if the neighbor is the ultimate destination, *address resolution* should be performed again. The same NUD procedure can be used in IEEE 802.15.4 networks with mesh routing.

2.4 IP Security Protocol Support

IP Security protocol (IPSec) [33] provides per-packet authenticity and confidentiality guarantees between communicating peers and is mandatory for IPv6 implementation. However, power constraints and limited processing capability of sensor devices make IPSec be computation intensive and the Internet Key Exchange (IKE) [34] be communication intensive. Since IEEE 802.15.4 devices already have security support at the link layer for confidentiality, integrity and replay protection, we need to define efficient authentication and key management protocols for sensor devices.

2.4.1 Securing Neighbor Discovery

Securing Neighbor Discovery Protocol (SEND) [37] has been designed to mitigate potential threats against the IPv6 neighbor discovery protocol. SEND protocol is based uniquely on the Cryptographically Generated Address (CGA) [38], which relies heavily on heavyweight RSA signatures and is not suitable for low-power wireless sensor networks. Another way of authenticating neighbor discovery messages is to use the *one-way hash chain* without using RSA signatures except for the first router solicitation message which uses CGA as proposed in [39]. However, the broadcast authentication using one-way hash chain is vulnerable to denial of service attacks [40]. In broadcast authentication using one-way hash chain, the broadcasting node first sends a packet with message integrity code using a key from the hash key chain and it sends the authentication key in some later packet. The receiving node has to store all the packets until it receives the packet containing the key in order to verify the authenticity of the previously received packet. A malicious node can send forged or fabricated broadcast packets and exhaust the receiver's buffer.

As discussed in Sec. 2.3, it is possible to optimize IPv6 neighbor discovery process for wireless sensor networks by using unicasts instead of multicasts. A node that needs neighbor information (e.g. router address, IPv6 prefix, duplicate address detection, etc.) can send router solicitation or neighbor solicitation message to the PAN coordinator which is the default access router. The PAN coordinator can then unicast router advertisement or proxy neighbor advertisement message to the soliciting node. These unicast messages can be protected using the shared secret key between the PAN coordinator and the soliciting node. We will explore the security issues in detail in Chapter 4 and present a security architecture for sensor networks in Chapter 5.

Chapter 3

Mesh Routing and Reliable Transport

Wireless sensor networks are expected to be deployed in mesh topology (see Fig. 3.1) for a wide variety of low-cost control and monitoring applications. In order to deliver IP packets inside the mesh of sensor devices multi-hop routing protocols are required. Furthermore packets may be lost or corrupted due to link failure or transmission errors. Therefore, it is also important to use some sort of reliable transport to ensure proper IPv6 operations. In following sections, we present efficient routing mechanism and reliable data transport for wireless sensor networks based on the IEEE 802.15.4 standard.

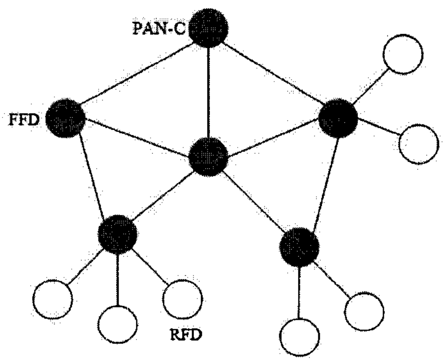


Figure 3.1: IEEE 802.15.4 Mesh PAN Architecture

3.1 Mesh Routing

Mesh routing protocols must take into account the resource constraints of sensor devices, as well as network topology changes due to a number of reasons such as uncertain radio connectivity or battery drain [26] [41]. Several unicast routing protocols have been proposed for resource-constrained sensor environments [42] [43] [44] [45] [46], most them are variants of the Ad-hoc On-demand Distance Vector routing (AODV) [47]. The IETF 6LoWPAN working group has been working on how to apply MANET routing protocols into LoWPANs [41].

In addition to unicast routing, many emerging sensor network applications involve mobile nodes with communication patterns requiring one-to-many and many-to-one routing topologies and hence require multicast routing. Multicasting is also required to support IPv6 operations in WPAN, for example, the all-node or solicited-node multicast is used to transmit NDP messages to on-link neighbors [32]. A number of ad hoc multicast routing protocols have also been proposed for MANETs [48] [49] [50] [51].

Most of the unicast and multicast routing protocols use network flooding for route discovery. Furthermore the all-node multicast of the IPv6 neighbor discovery protocol is to be realized with link-layer broadcasts because IEEE 802.15.4 devices do not provide link-layer multicast. An efficient flooding protocol underneath the IPv6 layer is also necessary to provide a virtual single-link broadcast WPAN similar to that of the Ethernet spanning tree bridges for proper IPv6 operations. Minimum connected dominating set (MCDS) routing [52] based on multipoint relays (MPR) can offer efficient way of flooding in IEEE 802.15.4 based sensor networks by reducing the number of redundant retransmissions while diffusing a broadcast message in the network.

3.1.1 Proposed Mesh Routing Protocol

In *classical flooding* (CF), a broadcast packet is forwarded by every node in the network exactly once. In wireless networks, CF uses many redundant retransmissions which may cause the *broadcast storm* problem [53]. It is possible to select a small subset of nodes called *minimum connected dominating set* (MCDS) to forward broadcast packets to all nodes in wireless networks [52] (see Fig. 3.2). It has been shown that the task of finding MCDS nodes is *NP-hard* [54]. Many heuristic solutions have been proposed for the MCDS problem in recent years for wireless ad hoc networks [54] [55] [56] [57] [58]. Some of them solve the MCDS problem with polynomial complexities, while others use link-state information of 2-hop neighbors for selecting broadcast forwarders and are not suitable for resource-poor wireless sensor networks.

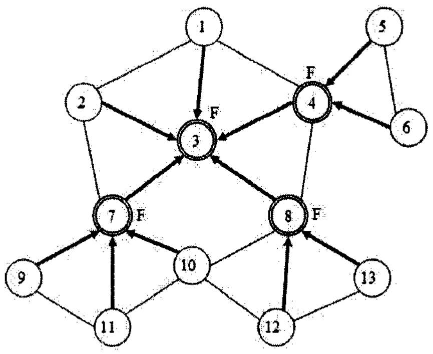


Figure 3.2: The MCDS Tree for the Network of Fig. 3.1

The Optimized Link State Routing (OLSR) [58] protocol optimizes the overhead from flooding of control traffic by using only selected nodes called *multipoint relays* (MPRs). Each node selects its MPR set from among its 1-hop symmetric neighbors so that all 2-hop neighbors are reachable through the selected MPRs. Zigbee [59] uses a tree-based data broadcast technique for IEEE 802.15.4 networks, which minimizes the number of forwarders called *routers* required for flooding [60]. Zigbee *routers* employ self-pruning and forward node selection algorithms based on the hierarchical addressing of the logical spanning tree that is formed during association in which a node can determine the addresses of tree neighbors from its own address and prunes forwarders which are within the radio range.

Ruiz et al. [55] present a distributed approximation algorithm to the MCDS problem, which produces a minimal cost multicast tree for multi-hop wireless mesh networks. Receivers of a multicast group initiate the multicast tree formation by broadcasting a request only to their first-hop neighbors indicating the multicast group to join. Each of the neighbors acknowledges by sending the number of receivers which it covers, by counting the number of requests received. Each of the receivers then selects the node that has the highest number of receivers as its root or forwarder. Those nodes which are selected as forwarders repeat the process acting as receivers. This process yields a number of cost-efficient subtrees which might be isolated from one another. The multicast tree (i.e. Steiner tree) among the roots of the subtrees is built afterwards.

In this section, we present the design and evaluation of an *optimized broadcast tree* construction algorithm for efficient flooding inside low-power wireless networks. The proposed algorithm is adapted from [55], where every node selects its forwarder from a number of candidate nodes which has maximum degree of neighbors and thus produces an approximate MCDS tree. The novel features of the proposed algorithm are the broadcast tree construction without using periodic HELLO messages or routing table exchanges, 2-hop neighbors information, or any other unicast protocol; the broadcast tree maintenance uses the link-layer feedbacks in mobile environments.

Optimized Broadcast Tree Construction

The *optimized broadcast tree* construction uses CF for selecting forwarding nodes dynamically based on their degree of neighbors. The PAN coordinator (PANC) starts the process by flooding a broadcast tree discovery request (BCAST DISCOVERY) message to its neighbors to join the broadcast tree (see Fig. 3.3). A BCAST DISCOVERY message is transported in mesh frames and contains PANC's link-layer address as the *Originator Address*, the broadcast address as the *Final Destination Address*, *Hops Left* for computing distance from the PANC, *Sequence Number* for duplicate packet detection. In addition to above mesh header fields, a BCAST DISCOVERY message also contains the number of neighboring nodes associated to this node called *degree of neighbors*, and the link-layer address of the forwarder toward the PANC called *parent address.* The PANC puts its own address in the *Originator Address* and the *parent address* fields and a predefined value of maximum number of hops in the *Hops Left* field.

A node that receives BCAST DISCOVERY messages, first looks up into its *membership table* if it already joined the tree. If it has not joined the tree before, then it sets the sender that has the maximum *degree of neighbors* as its forwarder and unicasts a broadcast tree join (BCAST JOIN) message to the selected forwarder. A BCAST JOIN message contains the *broadcast address* field in addition to mesh header fields. If the receiver is already a member of the broadcast tree, then it changes its forwarder only if the sender's *degree of neighbor* is higher than that of its current forwarder and if selecting this node as the forwarder does not create a local loop. In order to change the current forwarder, the receiver sends a BCAST JOIN message to the newly selected forwarder. The old forwarder removes a child node from its dependents list that joins a new forwarder. A BCAST JOIN message is acknowledged with a BCAST JOINACK message containing the *broadcast address* and a *role* fields in addition to mesh header fields. The value of the *role* field can be either *forwarder* or *leaf.*

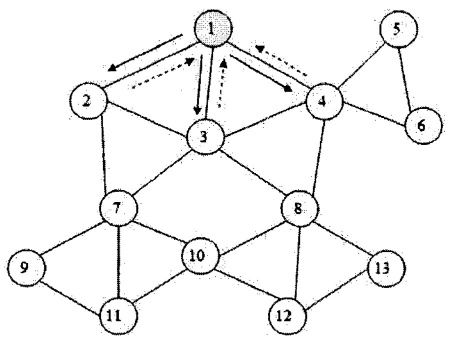


Figure 3.3: *Broadcast Tree Construction (Hop 1)*

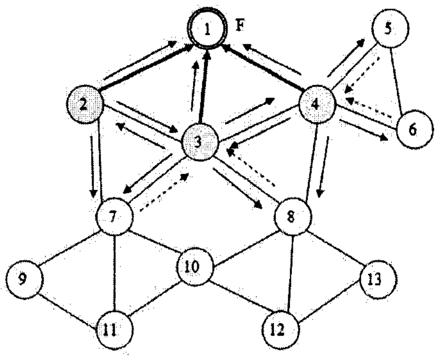


Figure 3.4: *Broadcast Tree Construction (Hop 2)*

When the PAN coordinator receives a BCAST JOIN message from one of its neighbors, it increments the *children counter* by one and changes the *role* to *forwarder* for the broadcast tree of *membership table* and then sends a BCAST JOINACK message to the dependent as an acknowledgment. When a broadcast forwarder overhears a BCAST JOIN message from one of its dependents that is joining another forwarder, it decrements *children counter* by one and removes that node from its *membership table*. If a forwarder does not have anymore child after removing a dependent from the *membership table*, it resets its *role* to *leaf*.

Each of the newly joined nodes then rebroadcasts the BCAST DISCOVERY message to its neighbors after decrementing the *Hops Left* by one and replacing the *degree of neighbors* with its own. Each node uses an exponential backoff timer for broadcasting BCAST DISCOVERY message in order to avoid the *broadcast storm* problem and waits for a pre-determined amount of time for BCAST JOIN message. A node with higher *degree of neighbors* has more chance to broadcast the BCAST DISCOVERY message first, this minimizes the need to change forwarders later. This method of hop-by-hop propagation of BCAST DISCOVERY message leads to the creation of a connected logical broadcast tree to be shared by all nodes in the network. Since every node chooses the node that advertises the maximum degree of neighbors as its forwarder, this approach yields a good approximation to the corresponding MCDS tree.

An Illustration: Here we give an illustrative example to explain the *optimized broadcast tree* construction algorithm. The PANC starts the process of optimized broadcast tree formation by broadcasting BCAST DISCOVERY message to its neighbors, nodes 2, 3 and 4, as shown in Fig. 3.3 using outbound solid arrows. The neighbors select the PANC as their forwarder and send BCAST JOIN message indicated by inbound dotted arrows. In the second round, the nodes 2, 3 and 4 broadcast BCAST DISCOVERY messages to their neighbors and nodes 5, 6, 7 and 8 choose their forwarders from advertisements with maximum degree of neighbors and minimum distance from the PANC and join the tree as shown in Fig. 3.4. When a node receives two or more advertisements with the same degree of neighbors and distance from the PANC, it selects the node with minimum node ID as its forwarder (e.g.

node 8 chooses node 3 instead of 4). In the third round, nodes 5, 6, 7 and 8 broadcast BCAST DISCOVERY messages to their neighbors and nodes 9, 10, 11, 12 and 13 join the tree as shown in Fig. 3.5. In the fourth round, nodes 9, 10, 11, 12 and 13 advertise BCAST DISCOVERY messages to their neighbors but no more nodes to join the tree and the process terminates. The final optimized broadcast tree is shown in Fig. 3.6 which is an approximation to the MCDS tree shown in Fig. 3.2.

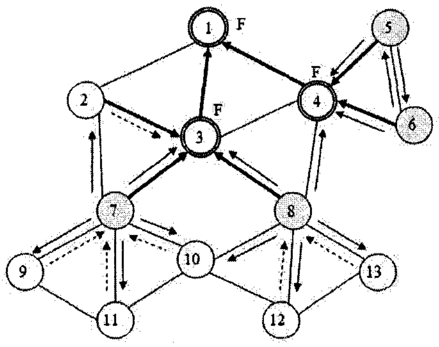


Figure 3.5: *Broadcast Tree Construction (Hop 3)*

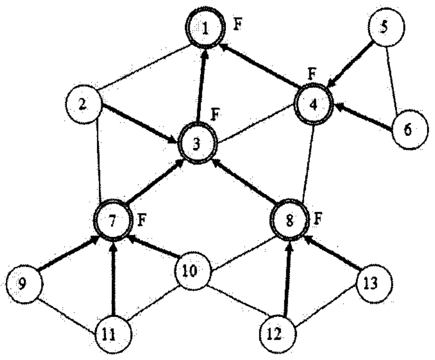


Figure 3.6: *The Optimized Broadcast Tree*

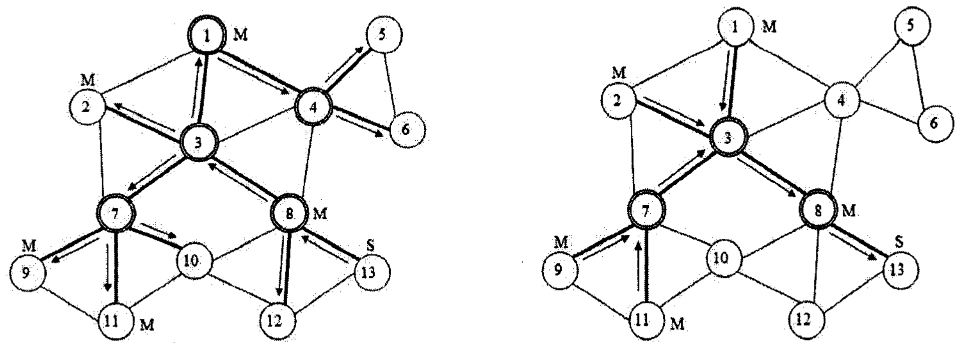
Broadcast Tree Maintenance: In mobile environments, a forwarding node of the broadcast tree can either die or move away from its dependents. In such cases, the broadcast tree could be partitioned or leaf nodes could be isolated from the tree in spite of having connected physical network. To keep the broadcast tree up to date, we devise a reactive tree maintenance process. From link-layer feedback, when a *forwarder* sees that a *leaf node* dies or moves away, it removes that child from its *dependent list* and decrements its *children counter* by one and if it reaches zero, then the forwarder changes its role to *leaf.*

When a *leaf node* notices that its *forwarder* dies or goes away, it initiates a local recovery by broadcasting a BCAST SOLICITATION message to its neighbors. A BCAST SOLICITATION message contains only mesh header fields. If a node that receives the BCAST SOLICITATION message has a valid forwarder, it replies with a BCAST DISCOVERY message to the sender. Otherwise the receiver decrements the *Hops Left* by one and if *Hops Left* reaches zero, it drops the packet otherwise it rebroadcasts the BCAST SOLICITATION message to its neighbors. When the soliciting node receives a BCAST DISCOVERY message, it selects its forwarder and sends a BCAST JOIN message to the sender of BCAST DISCOVERY message. The BCAST JOIN message is then processed by the receiver as discussed in Sec. 3.1.1.

When a new node joins the network and does not receive a BCAST DISCOVERY message within a specified time, it broadcasts a BCAST SOLICITATION message to its neighbors. The neighbors replies with BCAST DISCOVERY message and the new node selects its forwarder based on the sender's degree of neighbors and sends a BCAST JOIN message to the selected forwarder. The selected forwarder then changes its role to *forwarder* (if needed), increments *children counter* by one and sends a BCAST JOINACK message to the newly joined node.

Optimized Multicast Routing

The On Demand Multicast Routing Protocol (ODMRP) [48] operates by periodically flooding a control packet to recreate the multicast forwarding tree. When a source has data to send to a multicast group, the source floods a query control packet which is forwarded by every node in the network. Each node sets the reverse path, so that it can forward query reply messages from the receivers of the group to the source. Each node that forwards a query reply message sets its group forwarding flag for the group indicated in the header of the packet. When a node receives a packet for a multicast group for which it has a set forwarding flag, the node broadcasts the packet if it is not a duplicate. The Adaptive Demand-Driven Multicast Routing (ADMR) [49], similar to ODMRP, dynamically establishes and maintains multicast forwarding states only for active groups and only in nodes located between multicast senders and receivers. The TinyADMR [61] is an optimized version of the ADMR protocol for wireless sensor networks. TinyADMR uses Path Delivery Ratio (PDR) as the routing metric, which is computed from the Link Quality Indicator (LQI) and the Link Delivery Ratio (LDR) other than hop count.



(a) Flooding MCAST DISCOVERY Message (b) Members Join Multicast Tree

Figure 3.7: Optimized Multicast Tree Formation

In this section, we present an *optimized multicast routing* protocol based on [61], which uses the *optimized broadcast tree* for flooding route discovery message instead of CF. When a source has data to send to a multicast group, it floods a multicast route discovery (MCAST DISCOVERY) message using the broadcast tree. The MCAST DISCOVERY message containing group address is delivered to all nodes in the network by the broadcast tree forwarders. Broadcast tree forwarders set reverse path for this sender and this group address before forwarding the MCAST DISCOVERY message. Upon receipt of MCAST DISCOVERY message, each of the group members sends MCAST JOIN message back to the sender using the reverse path. The broadcast tree forwarders on the path of MCAST JOIN message, set themselves as forwarders for this group address and hence the multicast tree is established. The other broadcast forwarders, which set reverse path but has not received MCAST JOIN message, remove the entry after a predefined amount of time. Once a sender receives a MCAST JOIN message, it can send data packets for this group over the multicast tree. Fig. 3.7 illustrates the process of creating a multicast tree on top of the broadcast tree, where double circles indicate forwarders, S stands for source node, and M represents group membership.

Data Structures: In order to support the functions described in Sec. 3.1.1, each node maintains two tables: *membership table* and *request table.* The membership table keeps records whether a node is a receiver or forwarder for a given group. The request table is used for duplicate packet detection.

- *Membership table:* This table stores one entry for each multicast group address for which this node is either a receiver member of a forwarder. Note that broadcast address is considered as a special multicast group address and forwarders for this address form the broadcast tree. Each entry in the membership table includes a flag to indicate if this node is a receiver and a flag to indicate if this node is a forwarder. This table also stores other parameters of the forwarding tree such as source address, parent address, distance from source, if this node is a forwarder for a group, then number of dependent nodes and their addresses for this forwarder.

- *Request table:* This table maintains the history of route discoveries in order to avoid forwarding duplicate route discovery messages. It stores requesting node's address, group address, and sequence number. When a node receives a route discovery message to join a multicast group, it checks its request table for an entry for that group. If no entry for the group exists in the table, the node records the address of the group, together with the address of the requesting node's address and the sequence number.

Forwarding Application Data: When the PAN coordinator receives an IP packet from the Internet to be delivered to a destination inside the PAN, it passes the packet to the mesh layer. The mesh layer constructs a mesh delivery frame for the IP packet as discussed in chapter 2. If it is a unicast packet, the mesh routing sublayer will employ the unicast routing to deliver the IP packet to its destination.

If it is a multicast packet, the mesh routing sublayer will employ the *optimized multicast routing* protocol to discover the multicast tree. The PAN coordinator then broadcasts the frame to its neighbors. A node that receives a mesh frame with a multicast destination address, first checks the *Sequence Number* for freshness and then checks its *membership table* to determine if it is a forwarder for this group. If the receiver is a forwarder for this group, it decrements the *Hops Left* by one. If *Hops Left* reaches zero, it drops the frame otherwise rebroadcasts the frame. A node will forward a packet only once by keeping records of *Originator Address* and *Sequence Number* pairs of the mesh header. For data packets, each receiving node checks its *membership table* to determine if it is a receiver for this group. If so, it will pass the packet up within the protocol stack to allow the packet to be processed as a received multicast packet.

In the reverse direction, when a node has a packet to be sent to the Internet, it forwards the packet to the PAN coordinator using mesh routing, the PAN coordinator then forwards the packet to the outside world like a conventional router. If an IP packet does not fit in a single IEEE 802.15.4 frame, it is broken into link fragments by the LoWPAN adaptation layer before passing it to the mesh routing sublayer. On the receiving side, the LoWPAN adaptation layer performs the reassembly of IP packet fragments as discussed in chapter 2.

3.1.2 Simulation Results

In order to evaluate the performance of the *optimized broadcast tree* construction algorithm with the Zigbee forward node selection algorithm [60], we have simulated the protocol using NS-2 simulator [62] with the IEEE 802.15.4 MAC protocol in peer-to-peer mode. We have simulated the broadcast tree construction for networks of 15, 20, 30, 40 and 50 nodes with randomly generated topologies using the CMU's *cmu-scen-gen* utility. The area of the sensor field has been chosen based on the number of nodes so that the network has been connected. We have chosen 8, 9, 12, 17 and 23 group members randomly from the networks of 15, 20, 30, 40 and 50 nodes respectively for simulating the *multicast tree construction* algorithm.

Sizes of Broadcast and Multicast Trees

The number of nodes and the depth of the broadcast or multicast tree depends on the density and distributions of nodes in a network. However, the shortest path tree does not ensure the minimum connected dominating set forwarding tree. Fig. 3.8 shows the average numbers of forwarding nodes for the broadcast trees for different sized networks for our algorithm and the ZigBee forward node selection algorithm. Fig. 3.9 shows the average number of forwarding nodes for the multicast trees for specific group sizes for different sized networks. Note that the group source can be any node and may not be a broadcast forwarder. Thus the number of multicast forwarders including the group source sometimes is higher than the number of broadcast tree forwarders for smaller networks.

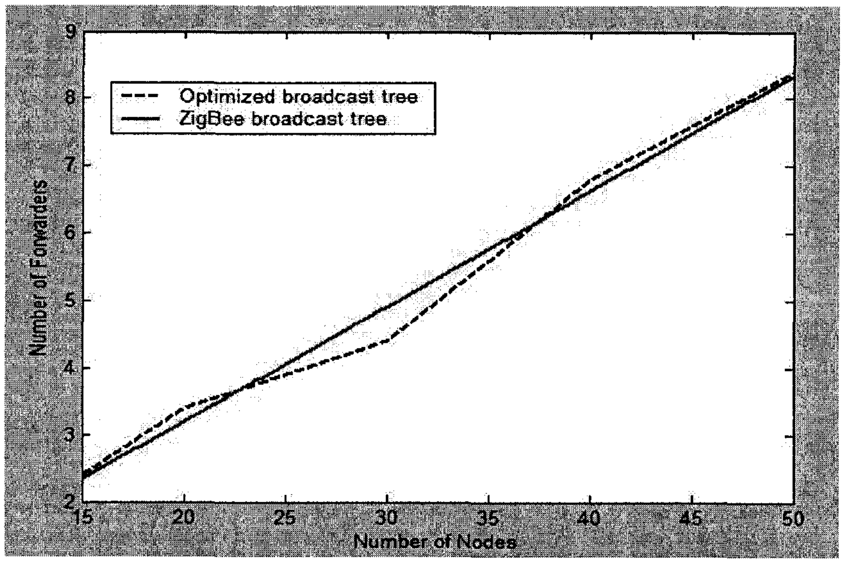


Figure 3.8: Sizes of Broadcast Trees

Complexity of Broadcast Tree Construction

Simulation results show that the number of messages need to be exchanged and the time it takes to converge the broadcast tree by the broadcast tree construction algorithm are linearly proportional to the number of nodes in the network. Fig. 3.10

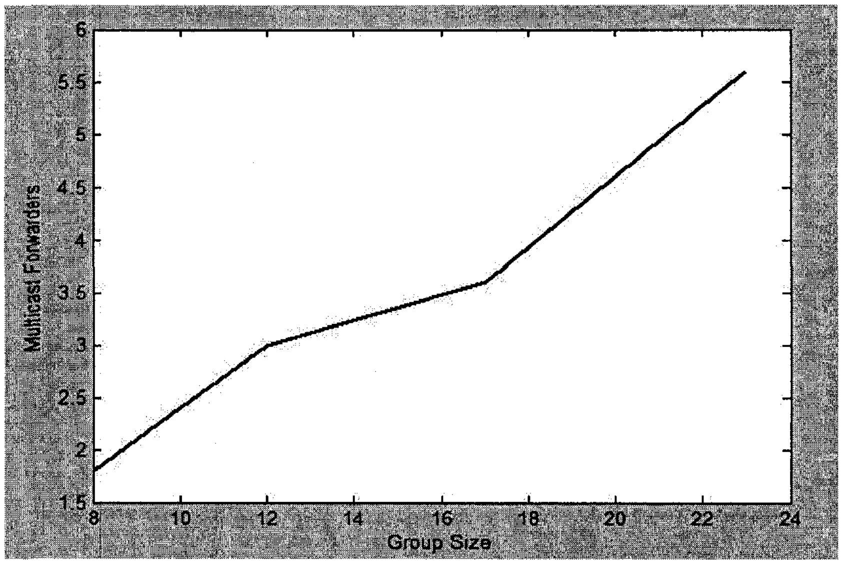


Figure 3.9: Sizes of Multicast Trees

shows the average number of messages exchanged in order to elect broadcast tree forwarders for different sized networks by the optimized broadcast tree construction algorithm and the ZigBee forward node selection algorithm. Note that the number of retransmissions needed for lost control messages are also taken into account, otherwise the number of messages would be much less.

The average time it takes to construct the broadcast tree during simulation is shown in Fig. 3.11. From the simulation results, it is shown that the time it takes to converge the broadcast tree is linear to the number of nodes in the network.

Redundant Retransmissions

The broadcast tree is shared by all the nodes in the network and can be used to flood higher layer data to the network. Broadcast packets get delivered with minimum number of retransmissions by intermediate nodes because only the forwarders relay packets sent by a source node. Fig. 3.12 presents simulation results to compare the average redundant retransmissions for delivering broadcast packets by the optimized broadcast tree and the ZigBee forward node selection algorithm.

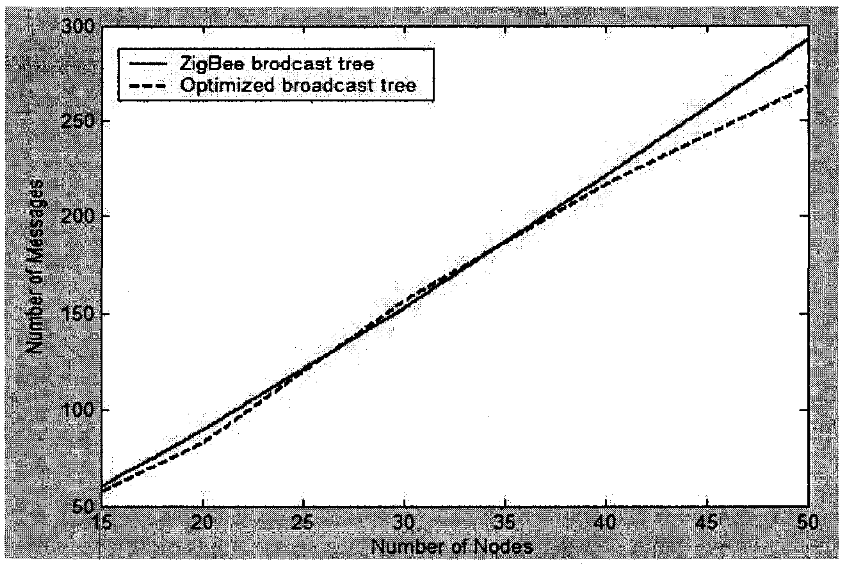


Figure 3.10: Communication Complexity of Broadcast Tree Construction

3.2 Reliable Data Transport

Many sensor network applications including medical monitoring, require reliable delivery of sensor readings. However, TCP, the widely used reliable transport protocol in the Internet, is not suitable for wireless sensor networks due to resource limitations. A large number of reliable transport protocols have been proposed for wireless sensor networks so far [63] [64] [65] [66] [67]. Some of them are used to deliver sensor readings or event data reliably in the sensors-to-sink direction, while others are used to disseminate reprogramming code or control information in the sink-to-sensors direction.

3.2.1 Proposed Reliable Transport Protocol

In this section, we present a *reliable data transport* (RDT) protocol for muti-hop wireless sensor networks based on hop-by-hop error recovery and congestion control mechanisms [14]. We have considered the scarcity of bandwidth and battery power of sensor nodes and the requirement of real-time delivery of sensor readings to the sink in designing the reliable transport protocol for wireless sensor networks. With above

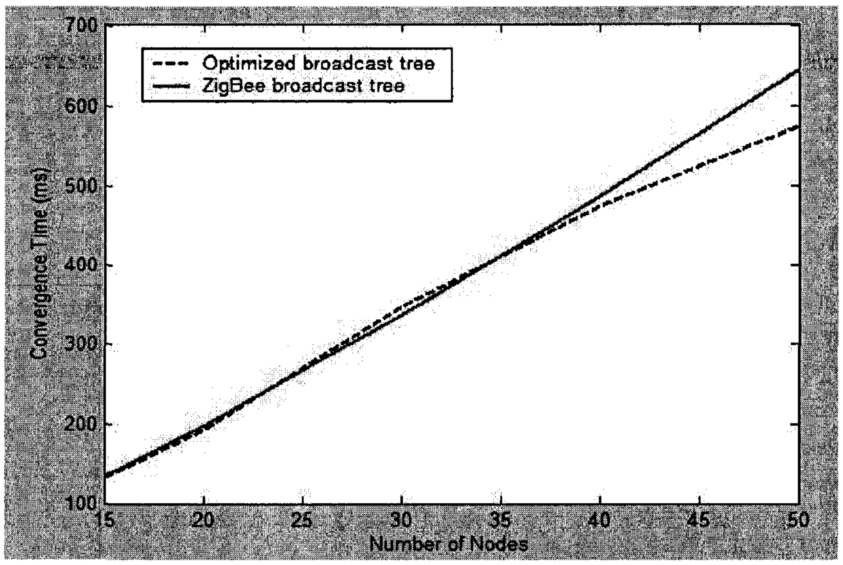


Figure 3.11: Time Complexity of Broadcast Tree Construction

constraints in mind, our objectives are to minimize the number of retransmissions due to packet loss and to detect packet loss as early as possible.

Hop-by-Hop Error Recovery

The Internet's reliable transport protocols (e.g. TCP) commonly use end-to-end error detection and recovery process. The number of retransmissions required to deliver a packet successfully in a multi-hop network depends on the link error rates on the path. For low error rate links, an end-to-end approach shows better performance than that of a hop-by-hop approach. In an end-to-end scheme, the number of retransmissions required to deliver data successfully in multi-hop networks increases exponentially to the number of hops [63]. For example, if the error rate of a wireless channel is p, then the probability of delivering a packet successfully across a single hop is $1-p$. Therefore, the probability of delivering a packet successfully across n hops is $(1-p)^n$. Moreover, if a packet is lost or corrupted at $(n-1)^{th}$ hop and the destination node requests a retransmission, then it requires $n-2$ unnecessary retransmissions by nodes from the source to the $(n-2)^{th}$ node and the probability of getting lost this retransmission before reaching $(n-1)^{th}$ node is p^{n-2}. For larger networks, it is almost

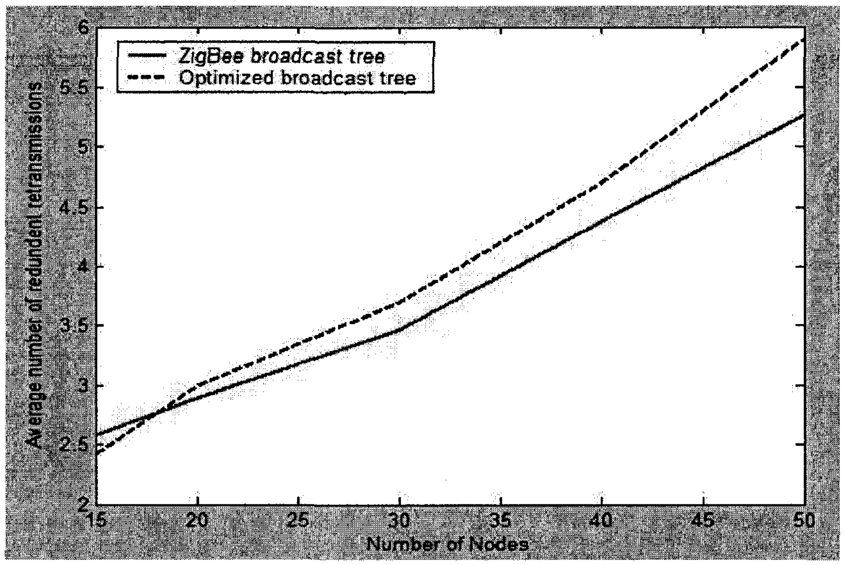


Figure 3.12: Redundant Retransmissions

impossible to deliver a message successfully using end-to-end mechanism in a lossy environment such as wireless sensors networks. On the other hand, in a hop-by-hop scheme, the number of retransmissions required to deliver data packets successfully in multi-hop networks increases linearly to the number of hops.

Therefore, to minimize the number of retransmissions and hence to save battery power, the reliable data transport protocol should use hop-by-hop error recovery procedure in which every node on the path detects packet error and performs error recovery by asking retransmission to its preceding node.

Selective Repeat (SR) protocol used in the Internet uses a fixed sized window and requests retransmissions selectively. The proposed reliable data transport protocol uses an ACK-based SR protocol, where every node uses a small sized buffer of length four. Packets in the buffer wait for an ACK from its successor node or recently received from the predecessor node. Packets in the window do not need to be received sequentially. The receiving node sends selective ACK when a corrupted packet or an out of order packet is received. The control packet includes the sequence base, i.e. the largest sequence number without any missing packets, and the sequence numbers of lost or corrupted packets. By using selective ACK mechanism, the number of control packets is minimized and the protocol works very much like negative ACK requests for lost or corrupted packets. For data integrity check, the cyclic redundancy code (CRC) is used like TCP.

The reliable transport protocol establishes a two-way connection between the source and the destination before transmitting application data. The source node sends a RDTSYN message containing a number of connection parameters such as connection identifier, buffer size, data size, start sequence number to the destination. Each node on the path sets up the connection by initializing connection parameters. The destination upon receiving RDTSYN replies with RDTSYNACK to notify the source that the connection has been established successfully. At the end of application data transmission, the source node sends RDTFIN to signal nodes along the path to close the connection. Each node on the path starts connection timeout timer after receiving RDTFIN message and forwards it. The destination after receiving RDTFIN releases all resources and sends a RDTFINACK to the source. The intermediate nodes forward RDTFINACK and release all resources. If RDTFINACK is lost, intermediate nodes releases connection when the connection timeouts. Appendix B and Appendix C present simplified state machines for the RDT sender and the RDT receiver respectively.

Congestion Control

The reliable data transport protocol uses BUFFER FULL and BUFFER EMPTY control messages for congestion control. Intermediate nodes on the path use small sized buffers of length four. When the *receive buffer* of an intermediate node becomes full, the receiving node sends a BUFFER FULL message to its preceding node, so that the preceding node stops sending further packets. When the node has buffer space enough to receive further packets, it sends a BUFFER EMPTY message to its preceding node. In wireless sensor networks, control messages can be lost due to transmission errors. To avoid the problems of missing control message, control messages are acknowledged with ACK. If an acknowledgement for a control message

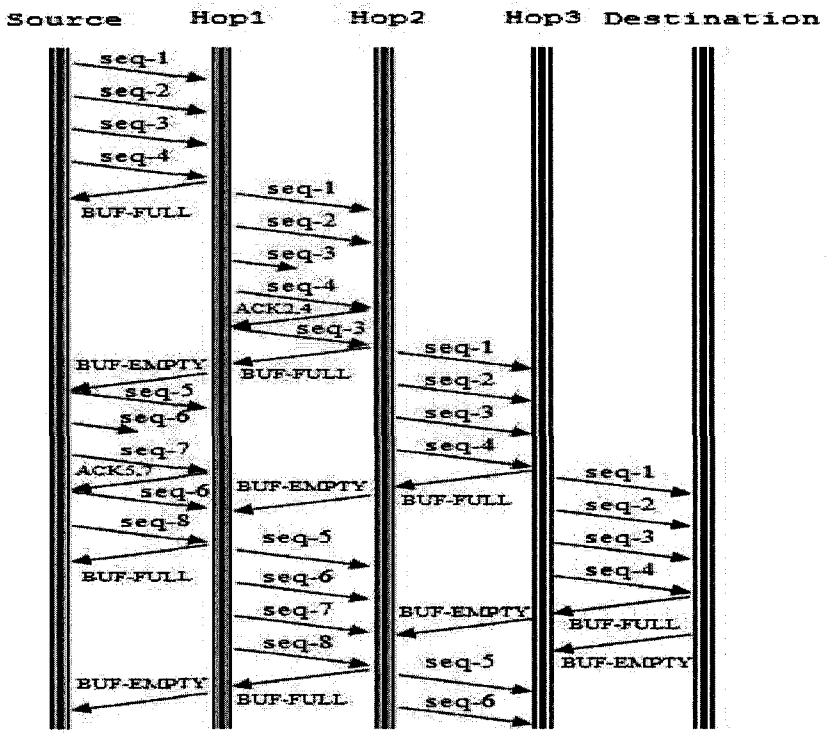


Figure 3.13: RDT Congestion Control

is not received within predetermined time, the sender retransmits the control message. If packet loss occurs due to congestion, the receiving node sends an ACK message to its preceding node to indicate congestion. After receiving an ACK packet, the preceding node retransmits missing packets that are lost due to congestion. Fig. 3.13 illustrates the selective repeat process for error recovery and hence reliable data transport.

Chapter 4

Sensor Network Security

The deployment of wireless sensor networks is becoming more and more common in a wide variety of applications for collecting and disseminating important and sensitive information. As the application of wireless sensor networks increasing, security becomes an important concern since any vulnerability in the system will limit its practical use. However, designing security protocols for resource-poor sensor devices is a difficult problem [68] [69]. A number of security protocols have been proposed for sensor networks in recent years [68] [69] [70] [71] [72] [59]. Boyle and Newe [40] present a detailed survey of security protocols proposed for wireless sensor networks.

4.1 Security Threats

Wireless sensor networks are vulnerable to a large number of security attacks because of their wireless broadcast communication media and resource constraints. In a broadcast medium, adversaries can easily eavesdrop on, intercept, inject, and alter transmitted data. Due to limited resources, (e.g. battery, bandwidth, processing power, etc.) an adversary can easily launch resource exhaustion attacks on sensor networks. Furthermore, since sensor networks are deployed in a variety of physically insecure environments, adversary can steal nodes, recover their cryptographic material, modify programs in sensor nodes, and pose as authorized nodes in the network.

Several types of denial of service (DoS) attacks can be launched against wireless sensor networks; at the PHY layer, the DoS attacks could be tempering and jamming electromagnetic signals. At the link layer, it could be collision and contention of deliberate stray frames. At the network layer, it could be an outburst of packets in the name of network traffic during homing. At the transport layer, attacks could be realized by keeping half-open and half-close TCP connections in sensor networks connected to the Internet.

In wireless sensor networks, it is very common to perform a task by a number of nodes together, there is a burgeoning need to distribute subtasks and ensure redundancy of information. In such cases, a malicious node can launch a *Sybil attack* by spoofing other nodes identities [73]. Sybil attacks can be performed against the distributed storage, routing mechanism, data aggregation, voting, fair resource-allocation and misbehavior detection, etc.

A malicious node can launch a *black hole attack* [74] to attract all the traffic in the sensor network. In this attack, an attacker listens to route requests and then replies to the requesting nodes that it has the high-quality or shortest path to the destination. Once a malicious node is able to insert itself between the communicating nodes, it is able to do anything with packets passing through it. Similar to black hole attack, is a *wormhole attack* [75], where an attacker records packets at one location in the network and tunnels those to another location.

4.2 Security Requirements

In this section, we formalize the security properties required by sensor networks, and show how they are directly applicable in typical sensor network applications.

4.2.1 Data Confidentiality

Data confidentiality is the process of keeping sensitive information secret from unauthorized parties. The main goal of data confidentiality is to ensure that sensitive information is not disclosed to any one other than the intended receivers. The standard way of achieving data confidentiality is to employ *encryption* with a secret key that is known only by the intended receivers. An encryption scheme should not only protect data from unauthorized parties, but also prevent adversaries from retrieving any information about the messages that have been encrypted. This strong property is known as *semantic security* [76].

4.2.2 Data Integrity

Data integrity protection is the process by which a node can verify that the received message is not altered in-transit by an adversary. However, due to malicious attacks or benign failures such as transmission collisions and radio propagation impairment, a message may be corrupted in-transit. Data integrity guarantees that a message is delivered as it is, without replacement, deletion, injection, resorting, or any other modifications. Data integrity is usually achieved by using symmetric-key message integrity code in a two-party communication.

4.2.3 Source Authentication

Source authentication is the process by which a node can verify the identity of the originating node of a message. It is very important in many sensor network functionalities such as beaconing, association, PAN ID conflict resolution, network reprogramming, etc. In wireless networks, an adversary can easily inject messages, so the receiver needs to make sure that the received message is from the claimed source and not from an adversary. In a two-party communication, source authentication is usually performed by using symmetric-key message integrity code with a shared secret key.

4.2.4 Replay Protection

Information flowing in sensor networks is often time-sensitive. A malicious node that eavesdrops on a legitimate communication between two authorized nodes can store valid messages and can replay them at some later time. Therefore, it is not enough to only guarantee confidentiality and authentication; it is also important to ensure that each message is fresh and valid in the context of the applications. Replay protection can be achieved with either a monotonically increasing sequence number, or a random nonce or a timestamp on each message.

4.2.5 Access control

Access control means the link layer protocol should prevent unauthorized parties from participating in the network. Since a sensor device is designed for a specific application and is not shared by different applications or users, device authentication is sufficient for access control and user authentication is not required. Legitimate nodes should be able to detect messages from unauthorized nodes and discard them.

4.2.6 Availability

The goal of availability is to ensure the survivability of network services despite DoS attacks or any other failures (e.g. link or node failure). Since sensor network devices are highly resource-constrained, they can easily suffer from attacks based on resource consumption. Availability of sensor networks is commonly ensured using redundant nodes in case of DoS attacks or node failures.

4.3 Security Building Blocks

4.3.1 Data Encryption

Data confidentiality is achieved by means of *encryption*, which is the process of transforming information by using some security parameters called *keys*, to make it unreadable to anyone except those possessing the secret key. There are two different encryption technologies: *symmetric-key* and *public-key*. Public-key encryption techniques are flexible for many applications; however, they are expensive in terms of computation, processing time, and key sizes and therefore are not suitable for resource-poor sensor devices. On the other hand, symmetric-key encryption techniques use simple computation and small sized keys and therefore suitable for securing sensor networks. There is a large number of symmetric-key encryption algorithms available and they are classified into two categories: *block ciphers* and *stream ciphers.*

A stream cipher uses a secret key and an Initialization Vector (IV) as the seed and generates a large pseudo-random keystream from them. The keystream is then XORed with the message to obtain *ciphertext*, the IV is also sent with the ciphertext. Stream ciphers are lightweight and very fast and suitable for real-time and resource-poor devices. However, stream ciphers require unique IV value for each block, otherwise it is often possible to recover both plaintexts, when the same IV is used to encrypt two different messages [69]. In order to ensure IV uniqueness, it must be long enough (e.g. 8 or 16 octets), however this increases the message length significantly and is not acceptable in wireless sensor networks.

A block cipher is a symmetric-key cipher, which performs a keyed pseudo-random permutation on fixed-length groups of bits, called *blocks.* There is a large number of block cipher algorithms, such as the Data Encryption Standard (DES), triple DES, Data Encryption Algorithm with Larger blocks (DEAL), the Advanced Encryption Standards (AES), RC family, International Data Encryption Algorithm (IDEA), Blowfish, Skipjack, etc. Since messages to be encrypted usually longer than the block size, block ciphers break the message into a number of pieces of that size. If the length of the last block is smaller than the block size, padding zeros are appended to the block. Block ciphers can be used in different modes: electronic codebook (ECB), cipher block chaining (CBC), cipher feedback (CFB), output feedback (OFB), counter (CTR), CBC in counter mode (CCM), EAX mode, Galois/Counter Mode (GCM), etc. The CBC is the most common mode of operation for block ciphers, in which the current block is XORed with the preceding ciphertext block using the same key for all blocks. A block cipher can also be used as a message integrity protection algorithm like AES-CCM.

4.3.2 Message Integrity Code

A common solution to message authenticity and integrity protection is to use a Message Integrity Code (MIC), also known as message authentication code. A MIC can be viewed as a cryptographically secure checksum of a message. Computing a MIC requires the sender and the receiver to share a secret key, and this key is a part of the input to the MIC computation. The sender computes a MIC over the message with the shared secret key and appends it to the message. The receiver sharing the same secret key recomputes the MIC on the received message and compares resulting MIC with the received one. If the received one matches with the computed one, the receiver accepts the message as authentic and rejects it otherwise. MIC computation must be strong so that an adversary cannot forge a MIC without knowing the secret key. There is a number of standard algorithms available for computing message integrity code such as Message Digest (MD5), Secure Hash Algorithm (SHA) family, Hashed Message Authentication Code (HMAC), AES cipher-based message authentication code (AES-CMAC), AES-CCM, etc.

4.4 Authentication and Key Management

Authentication and key management are two important services required to provide the proper operation of a network. Authentication is the process by which a node verifies the identity of another node in the network. Key management is the process by which cryptographic keys are distributed and shared throughout the network. There is a number of authentication and key management methods available, such as pre-shared key, trusted third party, and public-key certificate.

4.4.1 Key Distribution Methods

The simplest way of key distribution is to load one or more keys onto each sensor node prior to deployment. Most practical security protocols based on pre-deployed keying use either a single network-wide key shared by all sensor nodes or a set of keys randomly chosen from a key pool so that two nodes will share (at least) one key with a certain probability [77]. These protocols are easy to implement and have only little overhead since no complex computations or communication need to be performed. However, these protocols are not scalable for larger networks and are vulnerable to node capture attacks.

The second way of key distribution is to use a trusted third-party like a Public Key Infrastructure (PKI) or a Key Distribution Center (KDC) to establish shared keys between sensor nodes. Each node shares a long-term shared secret with the trusted-server and nodes relies completely on that server for key derivation. If the server is compromised, the trust amongst sensor nodes is gone. These protocols use symmetric-key cryptography for key derivation and have high communication overhead.

The Diffie-Hellman (DH) key exchange protocol is the commonly used key agreement protocol based on public-key cryptography and widely used in the Internet. The DH key exchange protocol is usually implemented using the multiplicative group of a finite field of prime numbers of the order of 1024 or 2048 bits [78], which is not suitable for resource-poor sensor devices. The recent implementation of the DH key exchange using Elliptic Curve Cryptography (ECC) has proven its feasibility for sensor networks [79]. The Elliptic-Curve Diffie-Hellman (ECDH) uses additive group of points on an elliptic curve defined over a finite field, but still consumes 67 times more energy than that of trusted third-party based key agreement using symmetric-key cryptography [80].

4.4.2 Authentication Methods

Challenge-Handshake Authentication Protocol (CHAP) is the commonly used authentication protocol, which uses a 3-way handshake to authenticate communicating parties. The initiator sends a random challenge to the responder in the first message. The responder replies with a value calculated from the initiator's random challenge and responder's random challenge using a one-way hash function. The initiator checks the response against its own calculation of the expected hash value. If the values match, the authenticator acknowledges with authentication success; otherwise it sends authentication failure to the responder.

Another way of authentication is to use a third-party authentication server, where communicating parties share secret keys with the server. The source sends authentication request to the authentication server for a destination. The authentication server generates a temporary authentication key and sends the key to both the source and the destination. Upon receipt of the temporary authentication key, the source generates a session key and sends it to the destination encrypted with the authentication key. The destination verifies the authenticity of the key and replies with an authentic success message.

The Extensible Authentication Protocol (EAP) [81] provides an infrastructure for network access clients and authentication servers to host plug-in modules for current and future authentication methods and technologies. An EAP infrastructure consists of an *EAP Peer* or *Supplicant* that is attempting to access the network, an *EAP Authenticator* or *Network Access Server* (NAS) that is requiring EAP authentication prior to granting access to the network, and an *Authentication Server* (AS) that negotiates the use of a specific EAP method with an EAP peer, validates the EAP peer's credentials, and authorizes access to the network. Typically, the authentication server is a Remote Authentication Dial-In User Service (RADIUS) server [82]. The EAP peer and the EAP authenticator exchange EAP messages using link-layer transport protocol such as Point-to-Point Protocol (PPP) [83] or the IEEE 802.1x protocol [84]. The EAP authenticator and the authentication server exchange EAP messages using RADIUS.

Overview of GPSK EAP Authentication

The Generalized Pre-Shared Key (GPSK) EAP authentication (EAP-GPSK) performs mutual authentication between an *EAP Peer* and the AS using a Pre-Shared

Key (PSK). The GPSK authentication protocol consists of four message exchanges (*GPSK-1, GPSK-2, GPSK-3, and GPSK-4*), in which both parties exchange *nonces* and their *identities*, compute and exchange MICs over the previously exchanged values using with the PSK. The MIC is considered as the proof of possession of the PSK [85]. A successful EAP-GPSK authentication exchange in pass-through mode is shown in Fig. 4.1.

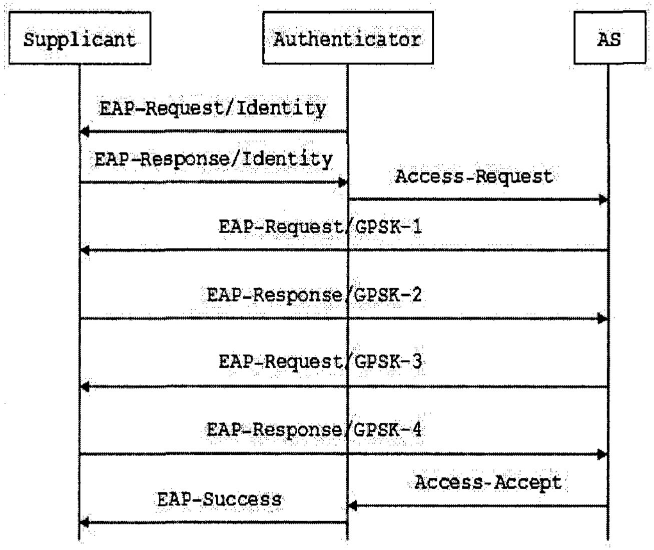


Figure 4.1: GPSK EAP Authentication Exchanges

The authenticator initiates the process by sending *EAP-Request/Identity* to the supplicant. In response to *EAP-Request/Identity*, the supplicant sends *EAP-Response /Identity* containing peer's Network Access ID (NAI) to the authenticator. Upon receipt of *EAP-Response/Identity* from a supplicant, the authenticator sends an *EAP-Request/Access-Request* RADIUS access request to the AS for the supplicant. The AS initiates the EAP-GPSK authentication process by sending *EAP-Request/GPSK-1* message consisting of the server's identity, a random number, and a list of available cipher suites to the supplicant. Upon receipt of *EAP-Request/GPSK-1*, the supplicant derives all GPSK keys from its own and received parameters using the PSK. In response to *EAP-Request/GPSK-1*, the supplicant sends *EAP-Response/GPSK-2* containing its identity, a random number, received parameters from the server, selected cipher suite, and a MIC computed over all these parameters. The AS derive keys from received and its own parameters using the PSK. The AS compares the received MIC with the computed MIC and if succeeds, it computes another MIC over the session parameters and returns it to the peer in *EAP-Request/GPSK-3* message. The supplicant verifies the received MIC and if the verification is successful, it sends *EAP-Response/GPSK-4* to the AS. Upon receipt of *EAP-Response/GPSK-4*, the AS assures that the supplicant has derived session necessary keys properly. The AS then sends an *EAP-Request/Access-Accept* RADIUS message to the authenticator to indicate the successful outcome of the authentication and exports the Master Session Key (MSK) to the authenticator.

4.5 WSN Security Protocols

4.5.1 SPINS

Perrig et al. [68] proposed a set of security protocols for TinyOS based wireless sensor networks (SPINS) including Secure Network Encryption Protocol (SNEP) and the micro version of Timed Efficient Stream Loss-tolerant Authentication (μTESLA). The SNEP provides data confidentiality and message integrity and authentication. The μTESLA provides a mechanism for broadcast authentication using hash chain and symmetric key primitives.

SNEP uses RC5 in counter (CTR) mode for encryption and the same encryption algorithm in cipher block chaining mode for message authentication (CBC-MAC). It provides semantic security by using a counter, which is incremented after each message, to prevent eavesdroppers from inferring the message content from the encrypted message. The counter value is sufficiently long enough never to repeat within the lifetime of the node. The monotonically increasing counter provides replay protection and weak freshness. It has a very low communication overhead, adding only 8 bytes per message.

μTESLA provides broadcast authentication by using a chain of hash values used as symmetric keys. It relies on delayed disclosure of symmetric keys and requires that the base station and other nodes are loosely time synchronized. The base station chooses a random value as the seed and generates a chain of key values from that seed using some public one-way hash function, F. For an authenticated packet to be sent, the base station computes a CBC-MAC on the packet with the key that is secret at that point in time (initially the last value of the key chain). When a node gets a packet, it can confirm that the base station did not yet disclose the corresponding MIC key, using its loosely synchronized clock, maximum synchronization error and the time at which the keys are to be disclosed. The node stores the packet in a buffer, aware that the MIC key is only known to the base station, and that no adversary could have altered the packet during transmission. After a predefined amount of time, the base station broadcasts the key to all receivers. The receiver can then verify the correctness of the key and use it to authenticate the packet stored in the buffer.

4.5.2 TinySec

Karlof et al. [69] designed link layer security architecture for TinyOS [2] based wireless sensor networks called TinySec, which borrowed similar concepts of authentication, encryption, message integrity, and replay protection from the SNEP. TinySec provides two different security options: *authenticated encryption* (TinySec-AE) and *authentication only* (TinySec-Auth). With TinySec-AE, TinySec encrypts the data payload and authenticates the packet with a MIC. The MIC is computed over the encrypted data and the packet header. In TinySec-Auth, TinySec authenticates the entire packet with a MIC, but the data payload is not encrypted. In TinySec-AE, a TinyOS packet can carry a payload of up to 29 bytes, with a packet header of 8 bytes and a MIC of 4 bytes, whereas in TinySec-Auth, a TinyOS packet can carry up to 29 bytes of payload, with a packet header of 4 bytes and a MIC of 4 bytes long.

TinySec uses either Skipjack or RC5 in CBC mode with cipher text stealing for encryption with and CBC-MAC for message authentication. TinySec uses specially formatted 8-byte IV with CBC encryption. Since the security of CBC mode of encryption depends on no reuse of IV and using longer IV increase packet overheads. Thus, to minimize packet overhead, TinySec generates IV from a 2-byte counter, the 2-byte source address, 2-byte destination address, 1-byte message type, and 1-byte payload length.

The security of CBC-MAC depends on the length of the MIC and TinySec uses a 4 byte MIC, which is much less than the conventional size of 8 or 16 bytes. Authors argue that 4 byte MIC provides an adequate level of security for sensor networks because an adversary needs 2^{31} attempts to forge one packet blindly. In sensor networks, it would take approximately 20 months with a 19.2 kbps radio to forge one packet blindly. Implicitly, there is an effective denial of service attack launched in this way, as the radio channel would be locked for an extended period as attempts are made. It is argued that a simple heuristic, whereby the nodes signal the base station when the rate of MIC failures exceeds a predetermined threshold would alleviate the problem should such an attack occur.

4.5.3 Sizzle

Gupta et al. [70] proposed an end-to-end security architecture for sensor networks. They have shown that public-key cryptography is feasible for sensor devices and existing public key protocols can be implemented on sensor devices. They implement a small-footprint HTTPS stack, nicknamed Sizzle, on different versions of the Berkeley/Crossbow motes. They have showed that Sizzle runs in less than 4KB of RAM, performs a full SSL handshake in 1 second (session reuse takes 0.5 seconds) and transfers 1KB of application data over SSL in 0.4 seconds.

Sizzle implements a subset of SSL features to meet tight resource constraints while addressing the security needs for a wide array of usage scenarios. Sizzle implements the MD5 and SHA1 hashing and RC4 for bulk encryption. It uses ECDH-ECDSA for key exchange due to its efficiency and uses RSA only for compatibility with existing web browsers. The cipher suites enabled in Sizzle do not entail sending the ServerKeyExchange message and the servers ECC or RSA public key is sent in a certificate and omits optional extensions. It uses 4-byte session identifier values rather than the 32-byte values used in Apache and other servers with much larger scalability requirements. Besides implementing session reuse, Sizzle also implements persistent HTTP(S) to further reduce the overhead of public-key cryptography. It does not send out the CertificateRequest message and clients are authenticated using passwords. It maintains state for only one SSL connection at a time. Another bandwidth saving feature of Sizzle is to use reduced HTTP headers by eliminating optional fields.

4.5.4 Zigbee Security Architecture

The Zigbee specification [59] defines security architecture for the IEEE 802.15.4 networks, which splits the security functionalities into three layers - medium access control (MAC), network (NWK) and application (APS). The main responsibilities of the NWK layer include the mechanisms used to join and leave a network, apply security to frames and to route frames to their intended destinations. The APS layer is responsible for the establishment and maintenance of security relationships. Zigbee uses a trust center, usually the Zigbee coordinator, for authentication and key management. The trust center plays three roles: trust manager, network manager, and configuration manager. The trust manager authenticates devices requesting to join the network. A network manager distributes and maintains network keys to devices. A configuration manager enables end-to-end security between devices by binding two applications. Zigbee define two security modes: residential for low-security applications and commercial for high-security applications. In residential mode, the trust center uses pre-shared keys to authenticate devices and does not provide key management. In commercial mode, the trust center establishes and maintains keys and freshness counters with every device in the network, providing centralized control and update of keys.

When a device completes IEEE 802.15.4 association procedure successfully with a coordinator, *router* in Zigbee terminology, the router initiates authentication procedure by sending Update Device request to the trust center. If the device is not pre-configured with a shared master key, the trust center sends a master key to the device in plain text. Otherwise, the trust center and the device use the Symmetric-Key Key Exchange (SKKE) protocol, an authentication protocol for mutual authentication and key agreement. The trust center initiates the SKKE protocol by issuing an authenticator's challenge to the device. The device replies with a device's challenge, a bit string chosen by the device, the device also includes a message integrity code (MIC) computed over the device's ID, the trust center's ID, the device's challenge, the trust center's challenge, the random bit string chosen by the device, and a constant 0x02 using the shared MIC key. The trust center validates the MIC and, if succeeds, responds with a random bit string with a MIC computed over the device's ID, the trust center's ID, the device's challenge, the trust center's challenge, the random bit string selected by the device, the random bit string chosen by the trust center, and a constant 0x03 using the shared MIC key. After successful authentication process, the trust center sends the network key securely to the device using *transport key* primitive.

Zigbee uses a triangular handshake procedure for mutual authentication and key exchange between any pair of devices in the PAN. The initiator device starts the process by sending key request including the responder's address to the trust center. Upon receipt of key request, the trust center generates a temporary master key and sends the key securely to both the initiator and the responder using Key Transport primitive. The initiator and the responder now employ the SKKE protocol with the temporary master key to authenticate each other and establish a shared session key.

Chapter 5

Authentication Framework for Sensor Networks

The IEEE 802.15.4 specification defines security primitives for low-power wireless devices to provide data confidentiality, message authentication and replay protection. Our main objective is to design an authentication mechanism on top of the IEEE 802.15.4 MAC layer to provide access control and key management. This prohibits unauthorized nodes from taking part in activities of the network such as forwarding data for other devices as coordinators, responding to a query by spoofing a legitimate node, or injecting fabricated packets into the network.

In order to provide authentication and key management in wireless sensor networks, we extend the EAP authentication framework for IEEE 802.15.4 networks. However, integrating EAP authentication model into IEEE 802.15.4 networks is a challenging task due to smaller MAC frame and multi-hop access network. The IEEE 802.15.4 MAC PDU leaves only 81 octets for upper layer after enabling MAC layer security. This is obviously far below for the existing authentication and key exchange protocols commonly used in the Internet such as the EAP-TLS [86] or the IKEv2 [34]. Note that a single EAP-TLS message can be up to 16 KB in length and the IKEv2 needs to support 1280 octets message exchange, which is impractical for IEEE 802.15.4 devices even if fragmentation and reassembly is used. Furthermore the EAP model assumes supplicants are within the radio range of the authenticator. But in IEEE 802.15.4 networks, it is very likely to have devices several hops away from the PAN coordinator. Therefore, the EAP authentication model is not directly applicable to the IEEE 802.15.4 compliant sensor networks.

A number of EAP methods has been in-use in different commercial networks, for example, the EAP-TLS in the Internet, the Global System for Mobile Communications (GSM) uses the EAP-SIM [87] and the Universal Mobile Telecommunications System (UMTS) and the CDMA2000 networks use the EAP-AKA [88]. Some of them are heavyweight and the others are specifically designed for a specific network. We propose to use a simple and generalized EAP method that is lightweight for sensor network implementation. The GPSK EAP method (EAP-GPSK) [85] discussed in Sec. 4.4.2 is the most suitable for resource-poor sensor devices and can be implemented with some optimizations.

5.1 Components of Proposed Architecture

The proposed EAP authentication framework consists of an *authentication server* (AS), an *authenticator*, *relay authenticators* (RA), and *supplicants.* We assume the PAN coordinator as the authenticator and we define a new role, *relay authenticators* for coordinators. We also assume that there is a pre-established security association between the AS and the authenticator. The EAP messages are transported in mesh frames in stead of MAC frame because mesh topology and hence mesh routing may be required to exchange EAP messages between the authenticator and a supplicant. Fig. 5.1 shows the components of proposed EAP authentication model for IEEE 802.15.4 compliant wireless sensor networks.

Authentication Server

We propose to use an authentication server, different from the PAN coordinator, to authenticate devices during joining the network. It can be a RADIUS server [82] with

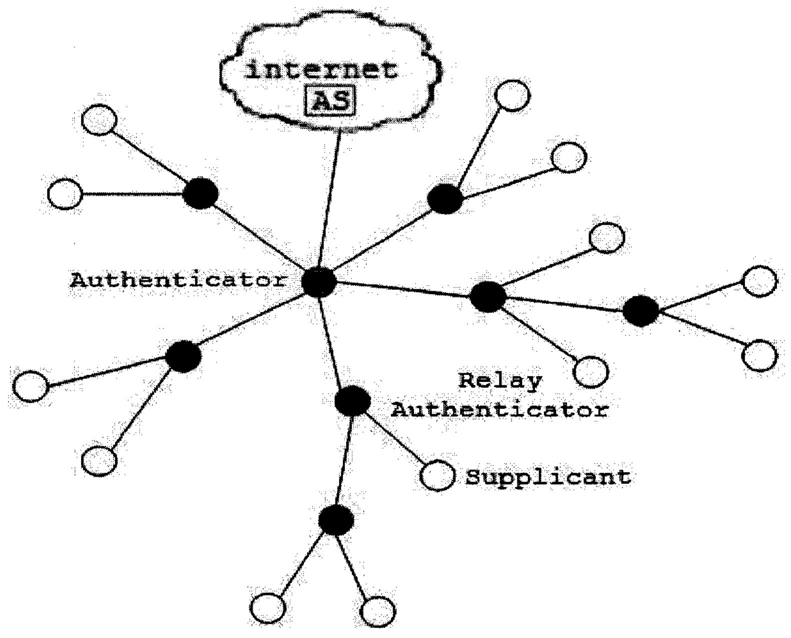


Figure 5.1: Components of Proposed Authentication Model

EAP-GPSK protocol configured to use the IEEE 802.15.4 supported cipher suites. The server shares 128-bit PSKs with each of the devices in the network. We assume that the server has a pre-established security association with the PAN coordinator.

Authenticator

We propose to use the PAN coordinator as the authenticator in order to authenticate newly joined devices using EAP-GPSK in pass-through mode. The authenticator has a pre-established security association with the AS to exchange EAP messages securely. After a successful authentication, the authenticator receives the Master Session Key (MSK) for the supplicant that took part in the authentication. Upon receipt of the MSK, the authenticator and the supplicant execute a 3-way Security Association (SA) protocol to establish session keys. The 3-way handshake establishes an Auxiliary MSK (AMSK), a Key Encryption Key (KEK) and a Temporal Session Key (TK). If the supplicant is authenticated via an RA, the authenticator exports the AMSK to the RA securely using previously established security association. The authenticator also derives message encryption key (Kenc) and message authentication key (Kmac) from the TK for the supplicant.

Relay Authenticator

To enable EAP authentication for IEEE 802.15.4 networks, we define a new role, *relay authenticator*, for devices that act as coordinators. Initially, all devices are configured as supplicants. After successful authentication and security association with the PAN coordinator, a coodinator changes its *role* to RA and enables MAC sublayer security. An RA accepts EAP messages from its children and forwards to its parent toward the PAN coordinator. Before forwarding an EAP message, an RA appends the MIC to the message. Thus, messages between two RAs or between the authenticator and the RA are cryptographically protected to avoid relaying forged EAP messages. When a child of an RA completes EAP authentication and the 3-way handshake successfully, the authenticator exports the AMSK securely to the RA. The RA then uses the received AMSK to establish session keys with the child by executing the 3-way handshake.

Supplicant

All devices in the network except the PAN coordinator need to be authenticated by the AS with the help of the PAN coordinator. After link-layer association with a coordinator, a supplicant initiates EAP authentication process with its coordinator. A supplicant assumes its coordinator as the authenticator and transparently performs the EAP authentication. If the coordinator is not the PAN coordinator, it acts as the relay authenticator provided that it has been authenticated earlier. After successful EAP-GPSK authentication, a supplicant derives the MSK, Extended MSK (EMSK), Session Key (SK) and Payload encryption Key (PK) keys and uses the MSK as the long term security association with the PAN coordinator. A supplicant then executes the 3-way handshake to establish session keys with the PAN coordinator. If the supplicant is a coordinator, it changes its role to relay authenticator after establishing security association with the PAN coordinator.

5.1.1 The Key Hierarchy

Each device in the network shares a PSK with the AS, which is used as the long term security association for authentication only. After a successful authentication, each of the devices derives a number of keys for different purposes as shown in Fig. 5.2.

The EAP-GPSK authentication process derives 4 keys: MSK, EMSK, SK, and PK. SK and PK are used during EAP authentication process, the AS exports MSK to the PAN coordinator and is used by the peer and the PAN coordinator as the long-term security association to derive future session keys. EMSK can be used by the authenticator and the supplicant to derive subsequent MSKs without using EAP exchanges.

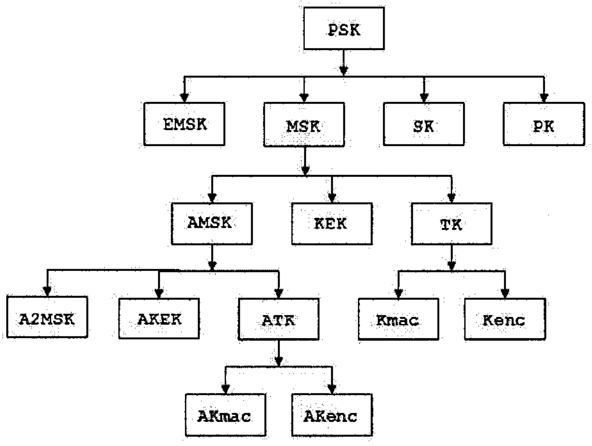


Figure 5.2: Proposed Key Hierarchy

The authenticator and the supplicant then use the 3-way SA protocol to establish following keys: AMSK, KEK, and TK. AMSK is not used for supplicants that are neighbors to the PAN coordinator and is discarded. KEK is used during security association and later for key transports. TK is the session key used for data encryption and authentication purposes. To avoid using the same key for encryption and message authentication, devices derive message encryption key (Kenc) and message authentication key (Kmac) from the TK.

If a device is authenticated via a RA, the authenticator exports the AMSK securely to the RA to be used as the long term security association between the supplicant and the RA. The supplicant and the RA then use the same 3-way SA protocol for mutual authentication and session key establishment. The RA and the supplicant derive an Auxiliary AMSK (A2MSK), an Auxiliary KEK (AKEK) and an Auxiliary TK (ATK) to be used as session keys. They also derive Auxiliary Kenc (AKenc) and Auxiliary Kmac (AKmac) from the ATK to avoid using the same key for message encryption and authentication. Note that auxiliary keys are not derived for supplicants that are neighbors to the PAN coordinator; the A2MSK is generated as part of the 3-way SA protocol because the same code is reused, which is never used and discarded.

5.1.2 IEEE 802.15.4 EAP Encapsulation

We use the IEEE 802.1X EAP over LAN (EAPOL) encapsulation [84] as the basis for our EAP encapsulation for LR-WPAN. To enable IEEE 802.15.4 devices to use EAP authentication, we use the frame format of Fig. 5.3(a) to carry EAP packets. We propose to use a reserved value (e.g. 100) for the *Frame Type* subfield of the *Frame Control* field of IEEE 802.15.4 MAC header to indicate IEEE 802.1X packets. All EAP messages are transported in mesh frames defined in [25] and contain the mesh header.

Fig. 5.3(d) shows the *EAP-Key* mesh frame format which is used to transport cryptographic keys and SA protocol messages. An *EAP-Key* mesh frame consists of a mesh header followed by an IEEE 802.1X header followed by a 1-octet *Desc Type* field which indicates the packet type, a 2-octet *Key Length* field which indicates the length of the key, an 8-octet *Replay Counter* field which is used to prevent replay attack, a 16-octet *Key IV* field which carries the IV value used to encrypt the key, a 1-octet *Key Index* field which is used to store and retrieve keys, a 16-octet *Key Signature* field which is used to carry signature of the key, an N-octet *Key* field which carries actual key data in encrypted form.

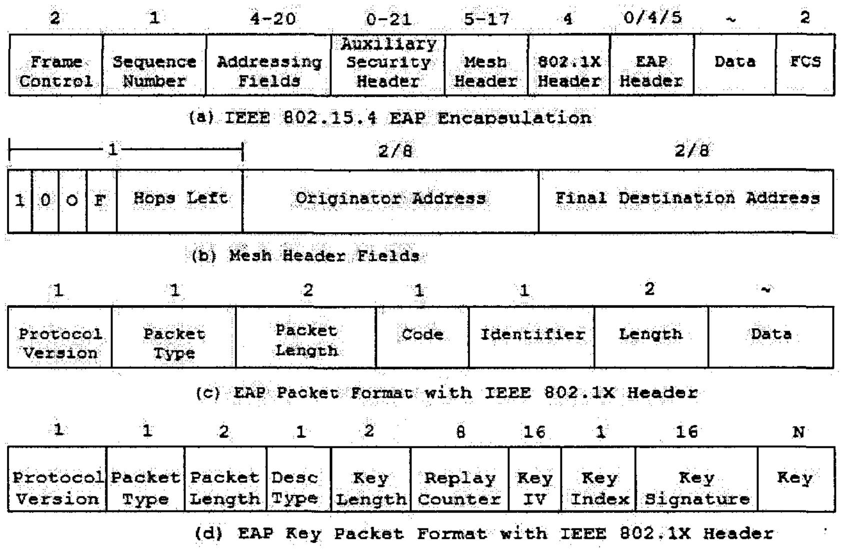


Figure 5.3: LoWPAN EAP Encapsulation

Mesh Header Fields

We have taken mesh header format from the IETF 6LoWPAN draft [25] as shown in Fig. 5.3(b) . The *O* is a 1-bit field and is set to 0 if the *Originator Address* is an IEEE extended 64-bit address (EUI-64), or to 1 if it is a short 16-bit address. The *F* is a 1-bit field and is set to 0 if the *Final Destination* address is an IEEE extended 64-bit address (EUI-64), or to 1 if it is a short 16-bit address. The *Hops Left* is a 6-bit field and is decremented by each forwarding node before sending this frame towards the next hop. The frame is not forwarded any further if *Hops Left* is decremented to 0. The *Originator Address* is the link-layer address of the originator of the mesh frame. The *Final Destination* address is the link-layer address of the intended destination of the mesh frame.

IEEE 802.1X Header Fields

The IEEE 802.1X header as shown in Fig. 5.3(c) consists of a 1-octet *Protocol Version* field which indicates the protocol version, a 1-octet *Packet Type* field which indicates the type of this EAP packet as shown in Table 5.1, and a 2-octet *Packet Length* field which contains the length of the packet excluding the IEEE 802.15.4 MAC header, the LoWPAN mesh header and the IEEE 802.1X header.

Packet Type	EAP Packet
0	EAP-Packet
1	EAP-Start
2	EAP-Logoff
3	EAP-Key

Table 5.1: IEEE 802.1X Types of EAP Packets

EAP Header Fields

The EAP header as shown in Fig. 5.3(c), consists of a 1-octet *Code* field which indicates the type of an EAP-Packet as shown in Table 5.2, a 1-octet *Identifier* field which matches the responses with requests, and a 2-octet *Length* field indicates the length of the EAP packet which includes only an EAP header and data field. The *EAP-Request* and *EAP-Response* packets have another 1-octet *Type* field after the *Length* field which is used to identify the EAP method being used (see Fig. 5.5).

Code	EAP-Packet
0	EAP-Request
1	EAP-Response
2	EAP-Success
3	EAP-Failure

Table 5.2: Different EAP Packets

5.2 Protocol Operations

5.2.1 Authenticating First Hop Neighbors

For the first hop neighbors of the PAN coordinator, EAP authentication process follows the standard rules of EAP pass-through mode (Sec. 4.4.2). The supplicant starts the process by sending a *EAP-Start* message to the PAN coordinator as shown in Fig. 5.4. Upon receipt of the *EAP-Start* message from one of its children, the PAN coordinator sends an *EAP-Request/Access-Request* RADIUS message to the AS to begin EAP-GPSK procedure.

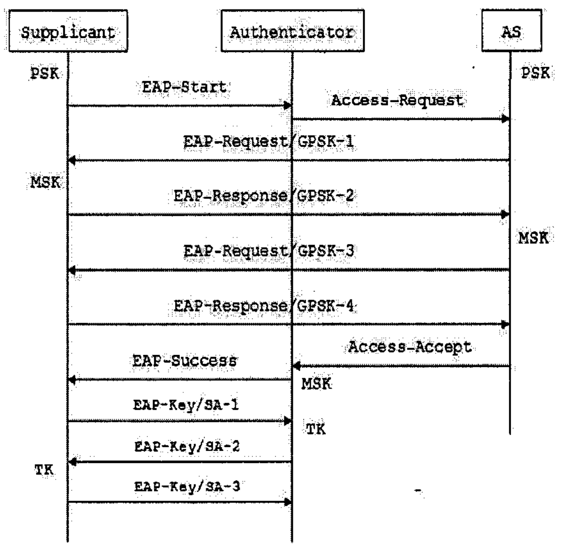


Figure 5.4: EAP Authentication for PAN Coordinator's Neighbors

In *EAP-Request/GPSK-1* message, the AS sends its identity, a random nonce, and an optional list of available cipher suites to the supplicant as shown in Fig. 5.5. Upon receipt of *EAP-Request/GPSK-1* message from the AS, the supplicant generates its own random nonce and derives the 128-bit master key from the server ID, server nonce, client ID, and client nonce using the generalized key derivation function (GKDF) of the selected cipher suite as follows:

$MK = GKDF16(0 \times 00, PL|PSK|CSuite|RndClnt|IdClnt|RndSrv|IdSrv)$

Here, *PL* represents the length of PSK (16 octets by default), *PSK* is the pre-shared key between the AS and the supplicant, *CSuite* is the selected cipher suite (ENC-MIC-64 by default), *RndClnt* is the client's random nonce, *IdClnt* is the client's ID, *RndSrv* is the AS's random nonce, and the *IdSrv* is the authentication server's ID.

The supplicant then derives MSK from the MK using GKDF as follows:

$KeyString = GKDF160(MK, RndClnt|IdClnt|RndSrv|IdSrv)$

The KDF generates a 160-octet *KeyString*, where *MSK* is the left most 64 octets (i.e. *KeyString[0..63]*), EMSK is the next 64 octets (i.e. *KeyString[64..127]*), *SK* is the next 16 octets (i.e. *KeyString[128..143]*) and *PK* is the last 16 octets (i.e. *KeyString[144..159]*).

1	1	2	1	1	2	16	16	2	~
Code	Identifier	Length	Type	OpCode	Server ID Length	Server ID	Server Random Number	CSuite Length	CSuite List

Figure 5.5: EAP-Request/GPSK-1 Packet

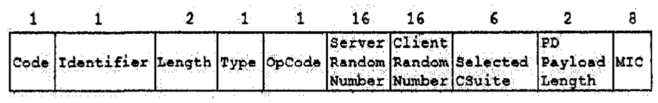
After deriving above keys, the supplicant sends *EAP-Response/GPSK-2* message to the AS which consists of the client's ID, client's random nonce, selected cipher suite, optional PD payload, and a message integrity code (MIC) computed over client's ID, client's random nonce, server's ID, server's random nonce, selected cipher suites, and optional PD payload, if present, with the SK (see Fig. 5.6).

1	1	2	1	1	2	8	16	0/6	2	8
Code	Identifier	Length	Type	OpCode	Client ID Length	Client ID	Client Random Number	Selected CSuite	PD Payload Length	MIC

Figure 5.6: EAP-Response/GPSK-2 Packet

Upon receipt of *EAP-Response/GPSK-2* message from the supplicant, the AS derives MSK, PMK, SK, and PK as described above and verifies the MIC in the message. If *EAP-Response/GPSK-2* passes the MIC verification, authentication is successful and the AS replies with *EAP-Request/GPSK-3* message including server's random nonce, client's random nonce, optional PD payload and a MIC computed over the client's ID, client's random nonce, selected cipher suites, and optional PD payload (if present) with the SK (see Fig. 5.7. When the supplicant receives the *EAP-Request/GPSK-3* message, it verifies the MIC and if verification succeeds, the supplicant assures that the server has established all the keys.

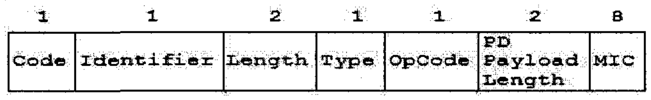
The supplicant acknowledges *EAP-Request/GPSK-3* by sending *EAP-Response/ GPSK-4* message containing optional PD payload and the MIC of the message (see Fig. 5.8). When the server receives the *EAP-Response/GPSK-4* message, the au-



1	1	2	1	1	16	16	6	2	8
Code	Identifier	Length	Type	OpCode	Server Random Number	Client Random Number	Selected CSuite	PD Payload Length	MIC

Figure 5.7: EAP-Request/GPSK-3 Packet

thentication is complete, the AS sends the *EAP-Response/Access-Accept* RADIUS message including the MSK to the PAN coordinator. The authenticator, upon receipt of the *EAP-Response/Access-Accept* RADIUS message and the MSK from the AS, sends the *EAP-Success* message to the supplicant, which indicates that the authenticator possesses the MSK.



1	1	2	1	1	2	8
Code	Identifier	Length	Type	OpCode	PD Payload Length	MIC

Figure 5.8: EAP-Response/GPSK-4 Packet

The 3-Way Security Association The supplicant starts the 3-way Security Association (SA) protocol with the PAN coordinator by sending the first message (*EAP-Key/SA-1*) to the authenticator, which carries SA message number in *Desc(riptor) Type*, initiator's ID length in *Key Length*, initiator's ID in *Key*, and initiator's random challenge in *Key IV* field of *EAP-Key* packet as shown in Fig. 5.9. Upon receipt of the *EAP-Key/SA-1* message, the authenticator generates its own random challenge and derives AMSK, KEK, and TK from the supplicant's ID, supplicant's random challenge, authenticator's ID, authenticator's random challenge using the generalized key derivation function (GKDF) with the MSK as follows:

$$Key = GKDF16(0 \times 00, ML|MSK|DefCS|RndSp|IdSp|RndAu|IdAu)$$

$$KeyString = GKDF64(Key, RndSp|IdSp|RndAu|IdAu)$$

Here, *ML* is the length of MSK, *DefCS* is the default cipher suite (ENC-MIC-64), *RndSp* is the supplicant's random challenge, *IdSp* is the supplicant's ID, *RndAu* is the authenticator's random challenge, and *IdAu* is the authenticator's ID. The GKDF generates a 64-octet *KeyString*, where *AMSK* is the left most 32-octet (i.e.

KeyString[0..31]), *KEK* is the next 16-octet of the *KeyString* (i.e. *KeyString[32..47]*) and *TK* is the right most 16-octet of the *KeyString* (i.e. *KeyString[48..63]*).

1	1	2	1	2	8	16	1	8	8
Protocol Version	Packet Type	Packet Length	SA-1	8	-	Initiator Nonce	-	-	Initiator ID

Figure 5.9: EAP-Key/SA-1 Packet

In response to SA-1, the authenticator sends the second message of the 3-way handshake (*EAP-Key/SA-2*) to the supplicant containing of SA message number in *Desc(riptor) Type*, responder's ID length in *Key Length*, responder's ID in *Key*, responder's random challenge in Key IV, and the MIC computed over initiator's ID, initiator's random challenge, responder's ID, and responder's random challenge using the KEK in *Key Signature* field of *EAP-Key* packet as shown in Fig. 5.10. The supplicant derives AMSK, KEK, and TK as described above and verifies the received MIC upon receipt of the *EAP-Key/SA-2* message.

1	1	2	1	2	8	16	1	8	8
Protocol Version	Packet Type	Packet Length	SA-2	8	-	Responder Nonce	-	MIC	Responder ID

Figure 5.10: EAP-Key/SA-2 Packet

If MIC verification succeeds, the key exchange is successful and the supplicant replies with the final message of the 3-way handshake (*EAP-Key/SA-3*) containing initiator's random challenge in *Key IV*, responder's random challenge in *Key*, and the MIC of the message computed using the KEK in *Key Signature* field of *EAP-Key* packet as shown in Fig. 5.11. Note that the *Key Length* field of *EAP-Key/SA-3* message contains the length of the *Key* field, i.e. the length of responder's random challenge. When the authenticator receives the *EAP-Key/SA-3* message from the supplicant, authentication is complete and the supplicant and the authenticator derive *message encryption key* (Kenc) and *message authentication key* (Kmac) from the TK as follows:

$KeyString = GKDF32(TK, RndSp|IdSp|RndAu|IdAu)$

Here, the *KeyString* is a 32-octet value and *Kenc* is the left 16-octet of the *KeyString* (i.e. *KeyString[0..15]*) and *Kmac* is the right 16-octet of the *KeyString* (i.e. *KeyString[16..31]*). Note that if the supplicant is a coordinator, then it changes its role to RA after successful authentication and key exchange. The AMSK is discarded by the authenticator and the supplicant if the supplicant is the first-hop neighbor of the authenticator.

1	1	2	1	2	8	16	1	8	16
Protocol Version	Packet Type	Packet Length	SA-3	16	-	Initiator Nonce	-	MIC	Responder Nonce

Figure 5.11: EAP-Key/SA-3 Packet

5.2.2 Authenticating Distant Nodes

Supplicants that are 2 or more hops away from the PAN coordinator use an RA to get authenticated. The coordinator of the supplicant acts as the RA for the supplicant provided that the coordinator has already been authenticated and established security association with the PAN coordinator. Fig. 5.12 illustrates the authentication and key exchange mechanisms for supplicants that are not neighbors of the PAN coordinator.

After completing association procedure with a coordinator, a node starts the EAP authentication by sending the *EAP-Start* message to its coordinator. The coordinator acts as the RA for the supplicant and forwards the *EAP-Start* message toward the PAN coordinator encapsulating in a mesh frame. Note that the RA inserts its own address into the frame and appends MIC to the mesh frame computed with the Kmac key shared with the PAN coordinator in order to protect the frame from in-flight tampering. The RA also applies MAC sublayer authentication mechanism before forwarding the frame to its parent to prevent malicious node from injecting false EAP frames into the network or replaying old frames. The intermediate nodes along the path employ MAC sublayer security mechanism to exchange EAP messages securely between the supplicant and the PAN coordinator. In order to deliver EAP messages to the PAN coordinator, intermediate nodes forward the message to their parents.

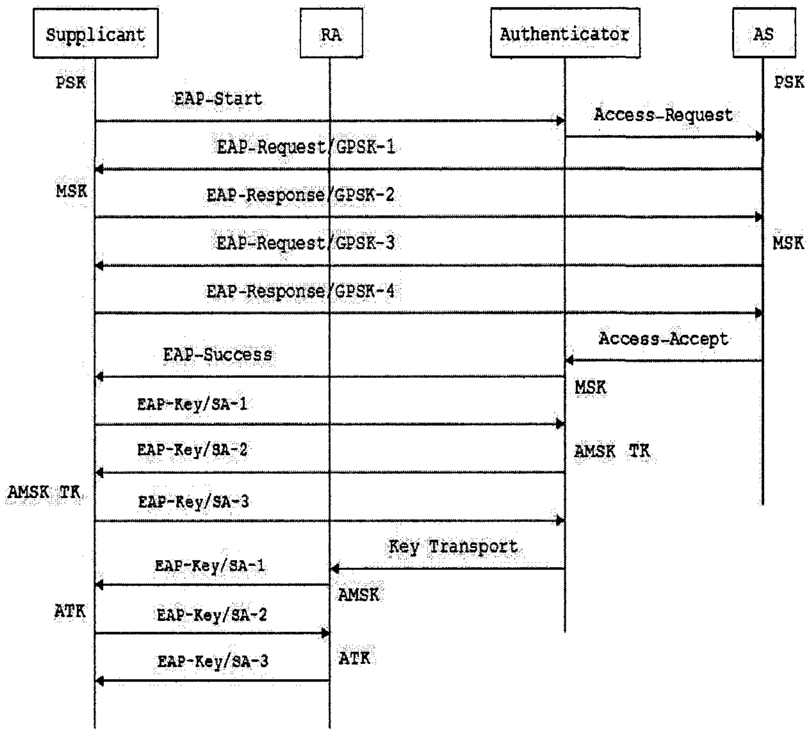


Figure 5.12: EAP Authentication for Distant Nodes

They also set the reverse path to the supplicant by inserting an entry into the mesh routing table.

When the authenticator receives an *EAP-Start* message from an RA, it verifies the MIC and if the message passes the MIC verification, the authenticator saves the addresses of the RA and the supplicant and sends an *EAP-Request/Access-Request* RADIUS message for the supplicant to the AS. The AS and the supplicant then perform EAP-GPSK exchanges to authenticate each other and derive EMSK, MSK, SK, and PK as described in section 5.2.1. The messages in reverse direction, from the PAN coordinator to the RA, are also transported in mesh frames and protected using MAC sublayer security mechanism. Upon successful EAP authentication, the AS sends the MSK securely to the PAN coordinator. The PAN coordinator and the supplicant then execute the 3-way SA protocol to derive AMSK, KEK, and TK as described in section 5.2.1.

For supplicants that use RA for EAP authentication, the PAN coordinator exports the AMSK to the RA after successful key establishment with the supplicant. The PAN coordinator uses *EAP-Key* mesh frame as shown in Fig. 5.3(d) to transport the AMSK to the RA. The key is encrypted using the KEK shared between the PAN coordinator and the RA, the IV value used in encryption and a signature of the key are included in the *EAP-Key* mesh frame. An *EAP-Key* mesh frame is also protected by using MAC sublayer security mechanism along the path from the PAN coordinator to the RA. The RA, on reception of the key frame, decrypts the key and verifies the signature of the key before accepting it. If the signature verification succeeds, the RA accepts the key and initiates the same 3-way SA handshake with the supplicant to derive auxiliary keys: AMSK, AKEK and ATK. The auxiliary key establishment process uses the same process as described in section 5.2.1. The RA and supplicant also derive AKenc and AKmac from the ATK. The A2MSK is derived as part of the 3-way handshake because the same code is reused and is discarded by the RA and the supplicant.

5.2.3 Mutual Authentication

In order to establish security association between any two nodes inside the PAN, we use the triangular symmetric-key authenticated key agreement protocol used in [59]. The source node sends an authentication request to the PAN coordinator for the intended destination. The source node uses authenticated mesh frame for sending authentication request. It computes MIC over the authentication request message using the shared authentication key with the PAN coordinator. The PAN coordinator verifies the authentication request and then checks whether the destination is authentic or not. If the destination is not an authentic node (i.e. the destination has not been authenticated with the PAN coordinator), it replies with an authentication failure message, otherwise the PAN coordinator generates a temporary key agreement key and sends the key securely to both the source and the destination nodes. The PAN coordinator uses *EAP-Key* mesh frame to transport the key encrypted using the KEK shared with the source or the destination; it also appends a signature of the key to the *EAP-Key* mesh frame. *EAP-Key* mesh frame is also protected by using MAC sublayer security mechanism along the path from the PAN coordinator to the source or the destination node. When the source node receives the temporary key agreement key, it knows that the destination is authentic and initiates the 3-way SA protocol with the destination and establishes session keys as described in Sec. 5.2.1.

5.3 Implementation Issues

Sensor devices have very little amount of computing resources such as 8 or 16-bit processor, few kilo bytes of RAM, few hundred bytes of instruction memory, low bandwidth radio, and more importantly limited battery power. Thus designing any protocol for sensor network must consider the resource utilization and keep resource requirement as low as possible. The authentication framework we have designed in this thesis is based on the extensible authentication protocol which was originally designed to be used in commercial networks. Therefore there is a strong need to adopt it so that it keeps resource utilization to a minimum.

We have proposed following optimizations to the EAP-GPSK implementation to keep GPSK messages as small as possible so that each of them can be transported in a single IEEE 802.15.4 MAC frame. The EAP-GPSK uses NAI as the *Client ID* and *Server ID*, which can be up to 256 octets in length and we propose to use sensor node's IEEE 64-bit extended MAC address as the *Client ID* and the IPv6 address of the AS as the *Server ID*. The EAP-GPSK uses 32-octet *random numbers*, which is too large for sensor networks and we propose to use 16-octet *random numbers* instead of 32-octet. In addition, an EAP peer repeats the *Server ID* and server's *random number* in GPSK-2 message, which increases the packet length significantly and we propose not to repeat them in GPSK-2, just use them in MIC computation. Finally, we propose to minimize cipher suite list to IEEE 802.15.4 standard cipher suites and use the ENC-MIC-64 cipher suite as the default cipher suite.

We propose that the supplicant starts the process in order to minimize communication overheads by avoiding a pair of *EAP-Request/Response* messages. In addition, when a distant node joins the network, the PAN coordinator is unaware of that and cannot initiate EAP authentication.

Since we are using the IEEE 802.15.4 security primitives for encryption, message authentication, and replay protection, our authentication framework implements only the EAP state machine partially, the GPSK protocol, and the 3-way SA protocol. We use the same key derivation function for deriving GPSK keys and SA keys. We have implemented the security framework as an NS-2 module and then converted the NS-2 code for TinyOS and compiled for Tmote.

We have implemented the supplicant functionalities in approximately 1800 lines of NS-2 C++ code including debugging codes. For relay authenticator functionalities, it requires about 2000 lines of NS-2 C++ code including debugging codes. In nesC, we have implemented the supplicant functionality in about 1500 lines of code, the relay authenticator functionality with about 1700 lines of code, and authenticator function with approximately 1400 lines of code. Note that TinySec implementation requires about 3000 lines of nesC code, which includes encryption and authentication primitives. After compilation for Tmote, it requires about 14 KB of program space and less than 2 KB of RAM for data structures. Thus the proposed solution can be implemented on sensor devices like Mica2, TelosB, or Tmote and can be used in many sensor network applications such as medical monitoring and meter readings for utility services.

5.4 Simulation Results

We have simulated the proposed security protocol using NS-2 for performance evaluation. We have implemented the protocol as the mesh sublayer on top of the MAC sublayer and below the IP layer. We have also implemented the MAC layer security primitives as defined in [9] because the NS-2 module for IEEE 802.15.4 did not implement security primitives. We have simulated the protocol with the IEEE 802.15.4

MAC protocol in beacon enabled mode for estimating the amount of time it takes to complete EAP authentication for nodes at different hops away. We have used the CMU's *cmu-scene-gen* utility for generating topologies randomly. The area of the network has been chosen based on the number of nodes in the network so that the network has been connected. We ran the simulation for several times and took the average with nodes at 1, 2, 3, 4, 5, and 6 hops away from the authenticator. Among the parameters evaluated are the time it takes to authenticate, to establish an SA, to transport the keys, etc. For each parameter a figure is obtained. The X-axis in the figures below represents the hop count and the Y-axis represents the amount of time in milliseconds.

We have plotted the average time requirements for GPSK authentication, SA handshake, and key transport after running several times. In different simulation runs, the variation of time requirement was small, for example, for GPSK authentication for first-hop neighbors the best case was 71 ms whereas the worst case was 80 ms. The maximum variation for GPSK authentication in all cases was 9 ms, for security association 5 ms, and for key transport 4 ms.

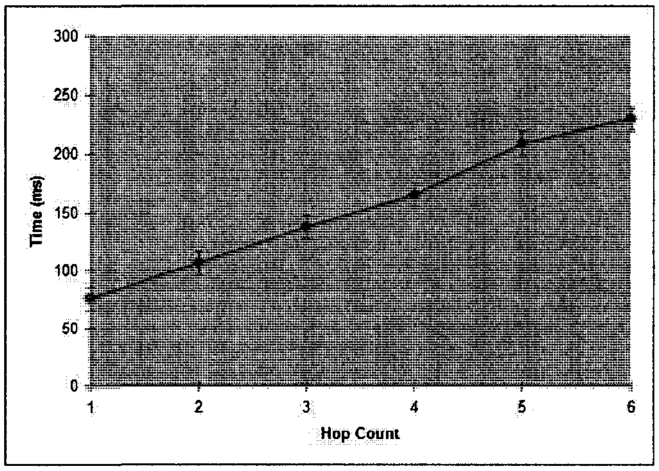


Figure 5.13: EAP-GPSK Authentication Time

5.4.1 EAP-GPSK Authentication Time

The amount of time it takes to complete the EAP-GPSK exchange is estimated beginning at the time when a node sends the *EAP-Start* message and ending at the time when it receives the *EAP-Success* message. Fig. 5.13 shows the simulation time it takes to complete the EAP-GPSK authentication for nodes at different hops away from the authenticator.

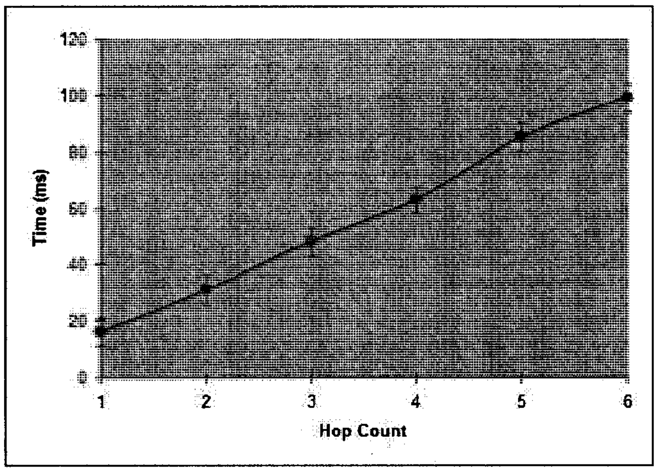


Figure 5.14: 3-Way Security Association Time

5.4.2 3-Way Security Association Time

The amount of time it takes to complete the 3-way security association is calculated from the time when a node sends the first message to the time when the authenticator completes processing the third message. Fig. 5.14 shows the simulation result for completing the 3-way SA protocol for nodes at different hops away from the PAN coordinator.

5.4.3 Key Transport Time

For distant nodes, the PAN coordinator exports the AMSK key to the RA after completing the 3-way handshake with the supplicant so that the RA can authenticate

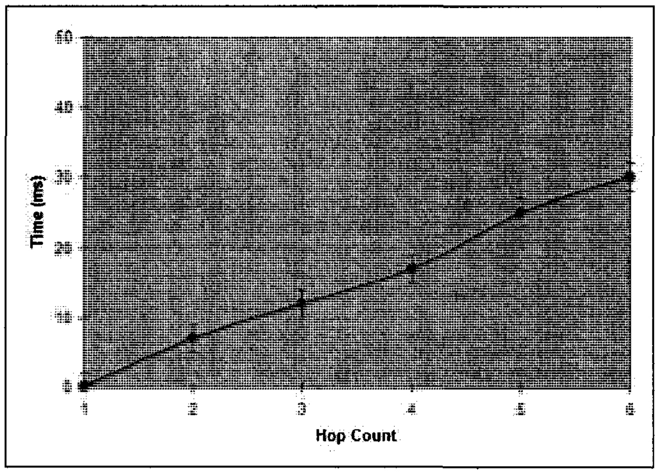


Figure 5.15: Key Transport Time

the supplicant and establish session keys using the 3-way handshake. This time varies for nodes that are different hops away from the PAN coordinator and contributes to the total authentication time. Fig. 5.15 shows the simulation results for transporting AMSK securely to RAs

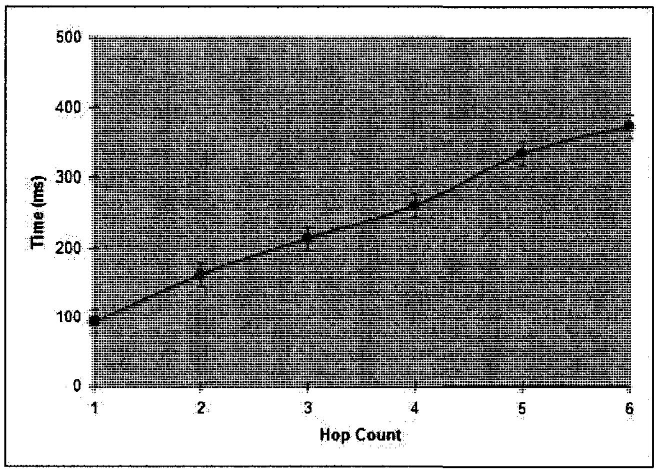


Figure 5.16: Network Access Time

5.4.4 Network Access Time

To get access to the network a node needs to get authenticated and to establish security association with the PAN coordinator. For distant nodes, they also need to establish security association with their relay authenticators (i.e. coordinators). Therefore, the network access time for PAN coordinator's first hop neighbors is the total of the EAP-GPSK time and a 3-way SA time. But for distant nodes, the network access time is the total of the EAP-GPSK time, a 3-way SA time, the key transport time and another 3-way handshake time. Fig. 5.16 shows the simulation results for network access time.

Chapter 6

Conclusions

Medical monitoring based on wireless sensor networking can provide better quality of medical care by exploiting low-cost, mobile, continuous, real-time, flexible and self-organizing properties of wireless sensor networks. IP enabled sensor networks have many advantages over classical sensor networks such as wide interoperability, existing security, naming, addressing, name-resolution, service discovery, network management, and reliability. However, due to limited resources of sensor devices, IP integration faces a number of challenges such as transporting large IP packets over small link-layer frames, inefficient routing and transport protocols, heavyweight security protocols etc. and requires optimization to save scarce resources of sensor devices (e.g. battery power, memory, bandwidth, etc.). The IETF 6LoWPAN working group has defined message formats and adaptation layer for transporting IPv6 packets over the IEEE 802.15.4 radio and MAC protocols, the de facto standard for wireless sensor networks.

In this thesis, we have presented an optimized broadcast tree construction algorithm for efficient flooding in the mesh of sensor devices and adapted on-demand unicast and multicast routing algorithms to use the broadcast tree for flooding; an efficient reliable transport protocol based on hop-by-hop error recovery; and finally a security architecture for authentication and key management for sensor devices using IEEE 802.15.4 security primitives. We have devised the solution to save memory by reusing existing security primitives, to reduce communications overhead to save energy and improve scalability.

First, we have shown that real-time medical monitoring using wireless sensor networks is feasible by implementing a wireless EEG system with TinyOS based sensor nodes. We have also shown that it is possible to incorporate machine intelligence algorithms to make decisions based on sensor readings. Since medical monitoring requires reliable delivery of sensor readings (e.g. heart-rate, blood-pressure, EKG readings, etc.), we have presented a reliable data transport protocol for sensor networks. The reliable transport performs hop-by-hop error recovery by using a modified selective repeat protocol with small sized window. We have simulated the protocol using the TinyOS simulator (TOSSIM) for performance evaluation and shown that the hop-by-hop technique performs better than end-to-end technique in highly error prone wireless sensor networks.

Second, we have developed an optimized broadcast tree construction algorithm based on minimum connected dominating set heuristics for wireless sensor networks. The algorithm produces smaller forward node sets than that of existing broadcast algorithms using only one-hop neighborhood information. We have adapted the on-demand multicast routing algorithm optimized for sensor networks to employ the broadcast tree for efficient flooding of control messages. We have simulated the broadcast and multicast tree construction algorithms using NS-2 with IEEE 802.15.4 MAC protocol. Our simulation results showed that the communication and time complexities for the creation of broadcast tree are linear to the number of nodes in the network.

Finally, we have designed the security architecture on top of the IEEE 802.15.4 security primitives for authentication and key management in wireless medical sensor networks. Since IEEE 802.15.4 standard provides data confidentiality, authenticity, and replay protection with symmetric-key cryptography, we have designed the authentication and key management protocols based on symmetric-key cryptography to reuse the available codes. We have extended the 3-party EAP authentication model for the multi-hop access network of IEEE 802.15.4.

We implemented an optimized version of the EAP-GPSK authentication protocol for authenticating sensor nodes before joining the network. For this purpose, we have defined EAP encapsulation over the IEEE 802.15.4 WPAN and key hierarchy for sensor devices. We have used the 3-way security association handshake mechanism for key establishment between any pair of nodes in the network. We have simulated the security protocol using the NS2 for performance study. Simulation results showed that the EAP authentication time in multi-hop access networks increases linearly as the number of hops increases. The major drawback of using EAP authentication in multi-hop access networks is the communications overhead since intermediate nodes have to retransmit EAP messages in either direction from the supplicant to the authenticator and vice-versa and it increases with number of hops.

6.1 Limitations

The main drawback of the broadcast tree construction algorithm is that it considers only the degree of neighbors for selecting forwarders and does not consider link quality and other path selection metric. The reliable transport protocol proposed in this thesis considers error recovery and congestion control. However, sensor networks require real-time delivery of sensor readings and not investigated in this thesis.

A number of limitations of the security architecture needs to be reconsidered for performance improvements. The major drawback of the design is the requirement of sequential authentication, i.e. a distant node cannot be authenticated until its parent is authenticated. The use of MAC address or IPv6 address as NAI also requires further consideration because this necessitates the authenticator to know the address of the AS for supplicants which is not the case in traditional EAP model. Another important issue not considered in this thesis and needs to be resloved is the broadcast authentication mechanism. Finally, the security architecture is designed for medical monitoring applications, therefore, it is not clear whether it is directly applicable for other sensor network applications and it requires further research.

6.2 Future Work

The routing algorithms presented in this thesis do not consider the dynamic concerns such as path selection metrics for selecting good paths from the source to the destination, routing table size limitations and link asymmetries because they are implementation-oriented but important in sensor networks. In IEEE 802.15.4 based sensor networks such concerns have greater impacts on the designs than other wireless networks and thus we leave such investigations for further research. Important future work may include performance evaluation under diverse mobility patterns and asymmetric links. In our experiments we used a sliding window protocol to ensure reliable message delivery and congestion control, which require further research for wireless sensor networks.

We have studied the performance of the proposed security protocol only in static environment, it is necessary to study the protocol performance in mobile environments especially in cluster architecture. It is important to secure IPv6 neighbor discovery messages and the broadcast and multicast tree construction messages, however existing broadcast authentication protocols are not suitable for low-power wireless sensor networks. Further research needs to be done to provide lightweight solutions to broadcast authentication. Most importantly, we have studied our protocols under constrained simulation environments and needs to be evaluated in real sensor deployment. Since now IPv6 enables sensor devices are in the market, the performance of the proposed solutions can be evaluated in real IPv6 enabled wireless sensor networks.

Bibliography

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A Survey on Sensor Networks. *IEEE Communications Magazine*, 40(8):102–114, 2002.

[2] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler. TinyOS: An Operating System for Wireless Sensor Networks. In *Ambient Intelligence*. Springer-Verlag, 2004.

[3] Upkar Varshney. Enhancing Wireless Patient Monitoring by Integrating Stored and Live Patient Information. In *Proceedings of the 19th IEEE Symposium on Computer-Based Medical Systems (CBMS'06)*, Salt Lake City, Utah, USA, June 2006. IEEE Computer Society.

[4] Kay Romer and Friedemann Mattern. The design space of wireless sensor networks. *IEEE Wireless Communications Magazine*, 11(6):54–61, December 2004.

[5] Krishna Venkatasubramanian, Guofeng Deng, Tridib Mukherjee, John Quintero, Valliappan Annamalai, and S. K. S. Gupta. Ayushman: A Wireless Sensor Network Based Health Monitoring Infrastructure and Testbed. In *IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS)*, June 2005.

[6] David Malan, Thaddeus Fulford-Jones, Matt Welsh, and Steve Moulton. CodeBlue: An Ad Hoc Sensor Network Infrastructure for Emergency Medical Care. In *International Workshop on Wearable and Implantable Body Sensor Networks*, April 2004.

[7] B. Sarikaya, M. A. Alim, and S. Rezaei. Integrating Wireless EEGs into Medical Sensor Networks. In *Proceedings of the International Wireless Communications and Mobile Computing Conference (IWCMC'06)*, Vancouver, BC, Canada, July 2006.

[8] A. Bastani, P. Medado, R. Qadir, and P. Manthena. The Feasibility of Acquiring and Transmitting Real-Time Wireless EEGs in the Emergency Department. *Acad Emerg Med*, 12(5):172–a, 2005.

[9] IEEE Computer Society. *IEEE Standard Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs), IEEE Std 802.15.4-2006*, September 2006.

[10] Konrad Lorincz, David Malan, Victor Shnayder, and Matt Welsh. Sensor Networks for Emergency Response: Challanges and Opportunities. *IEEE Pervasive Computing Magazine*, 3(4):16–23, October-December 2004.

[11] Moteiv Corporation, http://www.moteiv.com. *Tmote Sky : Quick Start Guide*, March 2006.

[12] Z. Keirn. Classification of EEG Signals for Brain-Machine Interfaces. *Colorado State University*, 2003.

[13] Crossbow Technology Inc., http://xbow.com. *Stargate Developer's Guide*, Rev. B edition, January 2006.

[14] A. Alim and B. Sarikaya. Designing Reliable Wireless EEG System with Motes. In *Proceedings of the International Conference on Computer and Information Technology (ICCIT'06)*, Dhaka, Bangladesh, December 2006.

[15] S. Makeig, M. Masterfield, and Tzyy-Ping Jung. ICA Toolbox Tutorial. *University of California-San Diego*, 2001.

[16] S. Young, G. Evermann, D. Kershaw, G. Moore, J. Odell, D. Ollason, V. Valtchev, and P. Woodland. The HTK Book, November 2001.

[17] J. Wallerius, L. J. Trejo, R. Matthews, R. Rosipal, and J. A. Caldwell. Robust Feature Extraction and Classification of EEG Spectra for Real-time Classification of Cognitive State. In *Proceedings of 11th International Conference on Human Computer Interaction*, Las Vegas, NV, USA, July 2005.

[18] N. Friedman and M. Goldszmidt. Building Classifiers Using Bayesian Networks. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, Menlo Park, CA, USA, 1996.

[19] Crossbow Technology Inc., http://xbow.com. *MPR-MIB Users Manual*, Revision B edition, June 2006.

[20] J. C. Haartsen. The Bluetooth Radio System. *IEEE Personal Communications*, February 2000.

[21] Karl Mayer and Wolfgang Fritsche. IP-enabled Wireless Sensor Networks and their integration into the Internet. In *Proceedings of the First International Conference on Integrated Internet Ad hoc and Sensor Networks (InterSense'06)*, Nice, France, May 2006.

[22] Tiago Camilo, Jorge S Silva, and Fernando Boavida. IPv6 in Wireless Sensor Networks, a New Challenge. In *Proceedings of the ConWiN 2005 - First International Workshop on Convergence of Heterogeneous Wireless Networks*, Budapest, Hungary, July 2005.

[23] M. Zuniga and B. Krishnamachari. Integrating Future Large-scale Wireless Sensor Networks with the Internet, 2005.

[24] A. Dunkels. Full TCP/IP for 8-bit Architectures. In *Proceedings of The First Internacional Conference on Mobile Systems, Applications, and Services (MOBISYS03)*, San Francisco, CA, USA, May 2003.

[25] G. Montenegro and N. Kushalnagar. Transmission of IPv6 Packets over IEEE 802.15.4 Networks. draft-ietf-6lowpan-format-06, November 2006.

[26] N. Kushalnagar, G. Montenegro, and C. Schumacher. 6LoWPAN: Overview, Assumptions, Problem Statement and Goals. draft-ietf-6lowpan-problem-08, February 2007.

[27] S. Chakrabarti and E. Nordmark. LoWPAN Neighbor Discovery Extensions. IETF-draft-chakrabarti-6lowpan-ipv6-nd-03, March 2007.

[28] Sensinode Ltd., Oulu, Finland. http://www.sensinode.com. *NanoStack with 6LoWPAN*, April 2007.

[29] Arch Rock Corp., San Francisco, CA, USA. http://www.archrock.com. *Arch Rock Primer Pack/IP Product Datasheet*, 2006.

[30] David E. Culler. 6LoWPAN: low-power IP connectivity. *Network World*, May 05 ,2007.

[31] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6), Specification. RFC 2460, December 1998.

[32] T. Narten, E. Nordmark, W. Simpson, and H. Soliman. Neighbor Discovery for IP version 6 (IPv6). IETF-draft-ietf-ipv6-2461bis-07, May 2006.

[33] S. Kent and R. Atkinson. Security Architecture for the Internet Protocol. RFC 2401, November 1998.

[34] C. Kaufman. Internet Key Exchange (IKEv2) Protocol. RFC 4306, December 2005.

[35] S. Thomson and T. Narten. IPv6 Stateless Address Autoconfiguration. RFC 1971, August 1996.

[36] M. Crawford. Transmission of IPv6 Packets over Ethernet Networks. RFC 2464, December 1998.

[37] J. Arkko, J. Kempf, B. Zill, and P. Nikander. SEcure Neighbor Discovery (SEND). RFC 3971, March 2005.

[38] T. Aura. Cryptographically Generated Addresses (CGA). RFC 3972, March 2005.

[39] W. Haddad, S. Krishnan, and J. Choi. Secure Neighbor Discovery (SEND) Optimization and Adaptation for Mobility: The OptiSEND Protocol. IETF-draft-haddad-mipshop-optisend-02, October 2006.

[40] D. Boyle and T. Newe. Security Protocols for use with Wireless Sensor Networks - A Survey of Security Architectures. In *Proceedings of the Third International Conference on Wireless and Mobile Communications (ICWMC'07)*, Guadeloupe, French Caribbean, March 2007.

[41] D. Kaspar, E. Kim, M. Shin, and H. Kim. Design Goals and Requirements for 6LoWPAN Mesh Routing. IETF-draft-dokaspar-6lowpan-routreq-00, February 2007.

[42] K. Kim, S. Daniel Park, G. Montenegro, S. Yoo, and N. Kushalnagar. 6LoWPAN Ad Hoc On-Demand Distance Vector Routing (LOAD). IETF-draft-daniel-6lowpan-load-adhoc-routing-02, March 2006.

[43] G. Montenegro and N. Kushalnagar. AODV for IEEE 802.15.4 Networks. IETF-draft-montenegro-lowpan-aodv-00, July 2005.

[44] I. Chakeres and L. Klein-Berndt. AODVjr, AODV Simplified. *Mobile Computing and Communications Review*, 6(3):100–101, July 2002.

[45] TinyOS Source Repository, http://tinyos.cvs.sourceforge.net/tinyos/tinyos-1.x/contrib/hsn/. *TinyAODV Implementation.*

[46] C. Gomez, P. Salvatella, O. Alonso, and J. Paradells. Adapting AODV for IEEE 802.15.4 Mesh Sensor Networks: Theoretical Discussion and Performance Evaluation in a Real Environment. In *Proceedings of the 2006 International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM'06)*, Niagra Falls, Buffalo, NY, USA, June 2006.

[47] Charles E. Perkins and Elizabeth M. Belding-Royer. Ad hoc On-Demand Distance Vector (AODV) Routing. In *Workshop on Mobile Computing Systems and Applications*, pages 90–100. ACM Press, February 1999.

[48] S. J. Lee, Mario Gerla, and C. C. Chiang. On Demand Multicast Routing Protocol. In *Proceedings of the IEEE Wireless Communications and Networking Conference, WCNC 99*, page 12981304, September 1999.

[49] J. G. Jetcheva and D. B. Johnson. Adapative Demand-Driven Multicast Routing in Multi-hop Wireless Ad Hoc Networks. In *ACM Mobihoc2001*, pages 33–44, October 2001.

[50] E. Royer and C. Perkins. Multicast operation of the Ad hoc On-demand Distance Vector routing protocol. In *ACM Mobicom*, pages 207–218, August 1999.

[51] A. Laouiti, P. Jacquet, P. Minet, L. Viennot, T. Clausen, and C. Adjih. Multicast Optimized Link State Routing. INRIA Rocquencourt, Tech. Rep., February 2003.

[52] Bevan Das and Vaduvur Bharghavan. Routing in Ad-Hoc Networks Using Minimum Connected Dominating Sets. In *IEEE International Conference on Communications (ICC'97)*, pages 376–380, 1997.

[53] S.Y. Ni, Y.C. Tseng, Y.S. Chen, and J.P. Sheu. The Broadcast Storm Problem in a Mobile Ad Hoc Network. In *ACM MobiCom99*, August 1999.

[54] Sergiy Butenko, Robert Murphey, and P. M. Pardalos. A New Heuristic for the Minimum Connected Dominating Set Problem on Ad hoc Wireless Networks. In *Recent Developments in Cooperative Control and Optimization*, pages 61–73. Kluwer Academic Publishers, 2004.

[55] Pedro M. Ruiz and Antonio F. Gomez-Skarmeta. Approximating Optimal Multicast Trees in Wireless Multihop Networks. In *Proceedings of the 10th IEEE Symposium on Computers and Communications (ISCC 2005)*, Cartagena, Spain, June 2005.

[56] A. Laouiti, A. Qayyum, and L. Viennot. Multipoint Relaying: An Efficient Technique for Flooding in Mobile Wireless Networks. In *35th Annual Hawaii International Conference on System Sciences (HICSS2001)*. IEEE Computer Society, 2001.

[57] J. Wu and F. Dai. A Generic Distributed Broadcast Scheme in Ad hoc Wireless Networks. In *ICDCS*, 2003.

[58] T. Plesse, J. Lecomte, C. Adjih, M. Badel, P. Jacquet, A. Laouiti, P. Minet, P. Muhlethaler, and A. Plakoo. OLSR Routing for Military Ad-hoc Networks. In *5th IEEE International Conference on Mobile and Wireless Communications Networks, MWCN'03*, Singapore, October 2003.

[59] ZigBeeTM Alliance, http://www.zigbee.org/en. *ZigBee Specification*, December 2006.

[60] G. Ding, Z. Sahinoglu, J. Zhang, and B. Bhargava. Tree-Based Data Broadcasting in IEEE 802.15.4 and ZigBee Networks. *IEEE Transaction on Mobile Computing*, pages 1561 – 1574, November 2006.

[61] Bor rong Chen, Kiran-Kumar Muniswamy-Reddy, and Matt Welsh. Ad-Hoc Multicast Routing on Resource-Limited Sensor Nodes. In *Proceedings of the Second ACM/Sigmobile workshop on Multi-hop Ad Hoc Networks: from theory to reality (REALMAN'06)*, Florence, Italy, May 2006.

[62] Kevin Fall and Kannan Varadhan. The ns Manual, June 2007.

[63] C. Wan, A. Campbell, and L. Krishnamurthy. PSFQ: A reliable transport protocol for wireless sensor networks. In *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications (WSNA'02)*, Atlanta, Georgia, USA, 2002. ACM Press.

[64] Fred Stann and John Heidemann. RMST: Reliable Data Transport in Sensor Networks. In *Proceedings of the First International Workshop on Sensor Net*

Protocols and Applications, pages 102–112, Anchorage, Alaska, USA, April 2003. IEEE Computer Society.

[65] Y. Sankarasubramaniam, O. Akan, and I. Akyildiz. ESRT: Event-to-Sink Reliable Transport in Wireless Sensor Networks. In *Proceedings of the ACM MobiHoc 03*, Annapolis, Maryland, USA, 2003.

[66] Yangfan Zhou, Michael R. Lyu, Jiangchuan Liu, and Hui Wang. PORT: a price-oriented reliable transport protocol for wireless sensor networks. In *Proceedings of 16th IEEE International Symposium on Software Reliability Engineering (ISSRE'05)*, Washington, DC, USA, November 2005. IEEE Computer Society.

[67] Peter Volgyesi, Andras Nadas, and Akos Ledeczi. Reliable Multihop Bulk Transfer for Wireless Sensor Networks. In *Proceedings of the 13th Annual IEEE International Symposium and Workshop on Engineering of Computer Based Systems (ECBS'06)*, Potsdam, Germany, March 2006. IEEE Computer Society.

[68] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. D. Tygar. SPINS: Security Protocols for Sensor Networks. In *Mobile Computing and Networking 2001*, Rome, Italy, 2001.

[69] Chris Karlof, Naveen Sastry, and David Wagner. TinySec: A Link Layer Security Architecture for Wireless Sensor Networks. In *Second ACM Conference on Embedded Networked Sensor Systems (SenSys 2004)*, November 2004.

[70] V. Gupta, M. Millard, S. Fung, Y. Zhu, N. Gura, H. Eberle, and S. Chang-Shantz. Sizzle: A Standards based End-to-end Security Architecture for the Embedded Internet. In *Third IEEE International Conference on Pervasive Computing and Communications (PerCom)*, Kauai, Hawaii, USA, March 2005.

[71] J. Heo and C. S. Hong. Efficient and Authenticated Key Agreement Mechanism in Low-Rate WPAN Environment. In *International Symposium on Wireless Pervasive Computing 2006*, pages 1–5, Phuket, Thailand, January 2006.

[72] A. Jehangir and S. M. Heemstra de Groot. A Security Architecture for Personal Networks. In *First International Workshop on Personalized Networks (PerNets 2006)*, San Jose, California, USA, July 2006.

[73] S. Daniel Park, K. Kim, E. Seo, and S. Chakrabarti. IPv6 over Low Power WPAN Security Analysis. IETF-draft-daniel-6lowpan-security-analysis-02, June 2006.

[74] Yih-Chun Hu, Adrian Perrig, and David B. Johnson. Ariadne: A Secure OnDemand Routing Protocol for Ad Hoc Networks. In *ACM MobiCom02*, Atlanta, Georgia, USA, September 2002.

[75] Yih-Chun Hu, Adrian Perrig, and David B. Johnson. Packet Leashes: A Defense Against Wormhole Attacks in Wireless Ad hoc Networks. In *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2003)*, San Francisco, CA, USA, April 2003.

[76] Shafi Goldwasser and Silvio Micali. Probabilistic Encryption. *Journal of Computer Security*, 28:270299, 1984.

[77] L. Eschenauer and V. Gligor. A Key Management Scheme for Distributed Sensor Networks. In *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS 2002)*, page 4147. ACM Press, 2002.

[78] W. Diffie and M. E. Hellman. New Directions in Cryptography. *IEEE Transaction of Information Theory*, IT-22:644654, November 1976.

[79] K. Kaabneh and H. Al-Bdour. Key Exchange Protocol in Elliptic Curve Cryptography with No Public Point. *American Journal of Applied Sciences*, 2(8):1232–1235, 2005.

[80] A. Hodjat and I. Verbauwhede. The Energy Cost of Secrets in Ad-hoc Networks. In *Proceedings of the 5th Workshop on Wireless Communications and Networking*, 2002.

[81] B. Aboba, L. Blunk, J. Vollbrecht, J. Carlson, and H. Levkowetz. Extensible Authentication Protocol (EAP). RFC 3748, June 2004.

[82] C. Rigney, S. Willens, A. Rubens, and W. Simpson. Remote Authentication Dial In User Service (RADIUS). RFC 2865, June 2000.

[83] W. Simpson. The Point-to-Point Protocol (PPP). RFC 1661, July 1994.

[84] IEEE Computer Society. *IEEE Standard for Local and Metropolitan Area Networks: Port-Based Network Access Control, IEEE Std 802.1X-2001*, June 2001.

[85] T. Clancy and H. Tschofenig. EAP Generalized Pre-Shared Key (EAP-GPSK). draft-ietf-emu-eap-gpsk-05.txt, April 2007.

[86] B. Aboba and D. Simon. PPP EAP TLS Authentication Protocol. RFC 2716, 1999.

[87] H. Haverinen and J. Salowey. Extensible Authentication Protocol Method for Global System for Mobile Communications (GSM) Subscriber Identity Modules (EAP-SIM). RFC 4186, January 2006.

[88] J. Arkko and H. Haverinen. Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA). RFC 4187, January 2006.

Appendix A

Users Manual

The simulation modules, *optimized broadcast and multicast tree construction algorithms* for efficient mesh routing in IEEE 802.15.4 networks and *authentication and key exchange protocol* for IPv6 enabled medical sensor networks, are written for NS-2.29 and run under Linux, Unix or Cygwin platforms. These implementations are the outcome of the master's thesis *A Security Architecture for IPv6 Enabled Medical Sensor Networks* and tested under Linux with NS-2.29.

A.1 Installing Modules

In the following, it is assumed that the NS-2 distribution *ns-allinone-2.29* is installed in */ns2.29* directory and tested that it works properly.

- Extract *thesis.tgz* into */ns2.29/ns-2.29/* directory using *tar -xvzf thesis.tgz*
- Compile NS-2 with the *make* command in */ns2.29/ns-2.29* directory

This will create two directories *mesh* and *security* in the */ns2.29/ns-2.29* directory. The */ns2.29/ns-2.29/mesh* directory contains all the source files of mesh routing protocol and the */ns2.29/ns-2.29/security* directory contains all the source files of the security protocol. It also replaces the */ns2.29/ns-2.29/Makefile*, the */ns2.29/ns-2.29/common/packet.h*, the */ns2.29/ns-2.29/queue/priqueue.cc*, the */ns2.29/ns-2.29*

/trace/cmu-trace.h, the */ns2.29/ns-2.29/trace/cmu-trace.cc*, the */ns2.29/ns-2.29/tcl /lib/ns-default.tcl*, the */ns2.29/ns-2.29/tcl/lib/ns-lib.tcl* and the */ns2.29/ns-2.29/tcl/ lib/ns-packet.tcl*. Both the */ns2.29/ns-2.29/mesh* and */ns2.29/ns-2.29/security* directories have sub-directories *test*, which contain sample simulation scripts.

A.2 Simulating Routing Protocols

A sample simulation script *mesh_demo.tcl* along with a deployment topology *mesh_demo.scn* is given for optimized broadcast tree construction and on-demand multicast tree construction algorithm simulation. To run the simulation, change the current directory to the */ns2.29/ns-2.29/mesh* and issue the command */ns2.29/ns-2.29/ns mesh_demo.tcl*. The simulation will print nodes status for broadcast and multicast trees while save the trace records in the *mesh_demo.tr* file to be used for performance study.

In order to customize a new simulation, one needs to generate a new topology using the CMU's scene generator utility, for example, *setdest -n 15 -p 100 -s 0.0 -t 100 -x 50 -y 50 test.scn* creates a topology of 15 nodes in a 50 × 50 grid with no mobility and save into the *test.scn* file.

Now modify the *mesh_demo.tcl* simulation script according to your requirements. The line *set val(rp) Mesh* indicates that we are going to simulate mesh routing protocol, i.e. optimized broadcast tree and multicast tree constructions algorithms. The *set val(nn) 15* indicates network size, *set val(x)* and *set val(y)* indicate the area of the field. Use the *source test.scn* command to include the topology generated using cmu-scene-gen utility. The *$ns_ at ⟨time⟩[$node_(⟨nodeid⟩) agent 255] pan-coordinator* starts the optimized broadcast tree construction process. The *$ns_ at ⟨time⟩[$node_(⟨nodeid⟩) agent 255] print-status* is used to display the status of nodes after broadcast tree construction. The command *$ns_ at ⟨time⟩ [$node_(⟨nodeid⟩) agent 255] mcast-group* is used to set group membership of a node. Note that we use only one multicast group at a time. The command *$ns_ at ⟨time⟩[$node_(⟨nodeid⟩)*

agent 255] mcast-source is used to initiate the multicast tree formation process. Finally, the command *$ns_ at ⟨time⟩ [$node_($i) agent 255] print-group-status* can be used to display nodes status after a multicast tree construction. Now, you can run the new simulation and analyze the simulation results by examining the *mesh_demo.tr* trace file.

A.3 Simulating Security Protocol

A sample simulation script *sec_demo.tcl* along with a topology *sec_demo.scn* and a node configuration file *nodes.cfg* is given for simulating the authentication and key exchange protocol. To run the simulation, change the current directory to the */ns2.29/ns-2.29/security* and issue the command */ns2.29/ns-2.29/ns sec_demo.tcl*. The simulation will print nodes activity during authentication and key exchange on the screen while save the trace records in the *sec_demo.tr* file to be used for performance study.

In order to customize a new simulation, one can generate a new topology using the CMU's scene generator utility, for example, *setdest -n 15 -p 100 -s 0.0 -t 100 -x 50 -y 50 test.scn* creates a topology of 15 nodes in a 50×50 grid with no mobility and save into the *test.scn* file. Note that we assume that node 0 acts as the PAN coordinator and hence the *authenticator* and that the node 1 acts as the EAP *authentication server* and are within the radio range of each other for simplicity. Write a configuration file for initializing nodes with bootstraping information (see the *nodes.cfg* for formats). Note that some pieces of the bootstraping information can be obtained from the IEEE 802.15.4 association procedure e.g. Node's short address, PAN coordinator's address, coordinator's address, chlidren count and children addresses while others need to be configured prior to deployment, e.g. the pre-shared key, authentication server's ID. All nodes are initialized to *supplicant* except the PAN coordinator which is initialized to *authenticator* and the *authentication server*. To simplify the implementation, we use the configuration file instead of obtaining from the link-layer.

Now modify the *sec_demo.tcl* for customized simulation. The command *$ns_ at ⟨time⟩ [$node_(⟨nodeid⟩) agent 255] eapconfig /ns2.29/ns-2.29/security/test/nodes.cfg* is used to configure nodes with bootstraping information from *nodes.cfg* file. The command *$ns_ at ⟨time⟩ [$node_(⟨nodeid⟩) agent 255] eapstart* is used to start authentication and key exchange process for node 2 with the authentication server. Now, you can run the new simulation and analyze the simulation results by examining the *sec_demo.tr* trace file.

Appendix B

Simplified RDT Sender

```
N = buff_size;
seqno = start_seq;
end_seq = start_seq + ceil(sizeof(app_data) / packet_size);
while(seqno <= end_seq){
switch(event){
case: application data received
make RDT packets from app_data;
start timer;
send N packets with sequence no. starting from seqno;
break;
case: timer timeout
retransmit N packets with sequence no. starting from seqno;
restart timer;
break;
case: ACK received with values x and y
retransmit packets with seqno between x and y;
restart timer;
break;
case: BUFFER FULL received
seqno = seqno + N;
stop timer;
if(this is an intermediate node){
send BUFFER EMPTY to its preceding node;
change state to RDT receiver;
}
break;
case: BUFFER EMPTY received
start timer;
send N packets with sequence no. starting from seqno;
break;
}
}
```

Appendix C

Simplified RDT Receiver

```
N = buff_size;
seqno = start_seq;
end_seq = start_seq + ceil(sizeof(app_data) / packet_size);
last_seq = start_seq;
while(seqno <= end_seq){
switch(event){
case: RDT packets received
if(not corrupted and in-order){
buffer packet;
start timer;
acksent = false;
}
else if(not corrupted and out-of-order){
buffer packet;
send ACK with seqno of last in-order and this packet;
acksent = true;
start timer;
}
else if(corrupted){
```

```
send ACK with seqno of last in-order packet;
acksent = true;
start timer;
}
if(buffer is full){
send BUFFER FULL packet;
seqno = seqno + N;
stop timer;
if(it is destination)
send BUFFER EMPTY packet;
else
change state to RDT Sender;
}
break;
case: timer timeout
if(not acksent){
send ACK with seqno of last in-order packet;
start timer;
acksent = true;
}
else{
retransmit ACK packet;
start timer;
}
break;
}
}
```

Appendix D

List of Acronyms

A2MSK – Auxiliary AMSK

ACK – Acknowledgement

ADC – Analog to Digital Converter

ADMR – Adaptive Demand Driven Multicast Routing

AES – Advanced Encryption Standard

AES-CCM – AES with CBC in Counter Mode

AKenc – Auxiliary Kenc

AKmac – Auxiliary Kmac

AMSK – Auxiliary MSK

AODV – Ad-hoc On-demand Distant Vector

APS – Application Layer

AS – Authentication Server

ATK – Auxiliary TK

BCAST – Broadcast

BP – Blood Pressure

CBC – Cipher Block Chaining CBQ – CodeBlue Query

CCM – CBC in Counter Mode

CDMA – Code Division Multiple Access

CF – Classical Flooding

CFB – Cipher Feedback

CMAC – Cipher based Message Authentication Code

CPU – Central Processing Unit

CRC – Cyclic Redundancy Check

CTS – Clear to Send

DAC – Digital to Analog Converter

DAD – Duplicate Address Detection

DEAL – Data Encryption Algorithm with Larger block

DES – Data Encryption Standard

DH – Diffie-Hellman

DMA – Dynamic Memory Allocation

DNS – Domain Name System

DoS – Denial of Service

DSDV – Destination Sequenced Distance Vector

DSR – Dynamic Source Routing

DYMO – Dynamic MANET On-demand

EAP – Extensible Authentication Protocol

EAP-AKA – EAP - Authentication and Key Agreement

EAP-GPSK – EAP - Generalized Pre-shared Key

EAP-SIM – EAP - GSM Subscriber Identity Module

EAP-TLS – EAP - Transport Layer Security

EAPOL – EAP over LAN

EAX – Encryption and Authentication mode

ECB – Electronic Codebook

ECC – Elliptic Curve Cryptography

ECDH – Elliptic Curve Diffie-Hellman

ECDSA – Elliptic Curve Digital Signature Algorithm

EC-MQV – Elliptic Curve Menezes-Qu-Vanstone

EEG – Electroencephalogram

EKG – Electocardiogram

EMSK – Extended MSK

ENC – Encryption

ESRT – Event to Sink Reliable Transport

EUI – Extended Unique Identifier

FFD – Full Function Device

GCM – Galois/Counter Mode

GKDF – Generalized KDF

GSM – Global Systems for Mobile Communications

HC – Header Compression

HMAC – Hashed Message Authentication Code

HTTP – Hyper-Text Transfer Protocol

I2C – Inter Integrated Circuit

ICA – Independent Component Analysis

ICMP – Internet Control Message Protocol

IDEA – International Data Encryption Algorithm

IEEE – Institute of Electrical and Electronics Engineers

IETF – Internet Engineering Task Force

IID – Interface Identifier

IKE – Internet Key Exchange

IKEv2 – IKE version 2

IP – Internet Protocol

IPSec – IP Security

IPv6 – Internet Protocol version 6

ISM – Industrial, Scientific and Medical

IV – Initialization Vector

KDF – Key Derivation Function

KEK – Key Encryption Key

Kenc – Session key for Encryption

Kmac – Session key for Authentication

LAN – Local Area Network

LDR – Link Delivery Ratio

LOAD – 6LoWPAN Ad hoc On-demand Distance Vector Routing

LoWPAN – Low power Wireless Personal Area Network

LQI – Link Quality Indicator

6LoWPAN – IPv6 for Low power Wireless Personal Area Network

LR-WPAN – Low-Rate Wireless Personal Area Network

μIP – Micro Internet Protocol

MAC – Medium Access Control

MAODV – Multicast AODV

MANET – Mobile Ad hoc Network

MCAST – Multicast

MCDS – Minimum Connected Dominating Set

MD5 – Message Digest version 5

MIC – Message Integrity Code

MFCC – Mel Frequency Cepstral Coefficient

MPR – Mutipoint Relay

MSK – Master Session Key

NA – Neighbor Advertisement

NACK – Negative Acknowledgement

NAI – Network Access Identifier

NAS – Network Access Server

NAT – Network Address Translator

NDP – Neighbor Discovery Protocol

NEMO – Network Mobility

NS – Neighbor Solicitation

NS-2 – Network Simulator version 2

NUD – Neighbor Unreachability Detection

NWK – Network Layer

ODMRP – On-demand Multicast Routing Protocol

OFB – Output Feedback

OLSR – Optimized Link State Routing

PDA – Personal Digital Assistant

PDR – Path Delivery Ratio

PAN – Personal Area Network

PANC – PAN Coordinator

PDU – Protocol Data Unit

PHY – Physical Layer

PK – Payload Encryption Key

PORT – Price Oriented Reliable Transport

POS – Personal Operating Space

PSFQ – Pump Slowly, Fetch Quickly

PSK – Preshared Key

RA – Relay Authenticator

RADIUS – Remote Authentication Dial In User Service

RAM – Random-Access Memory

RC – block cipher (Rivest Cipher)

RDT – Reliable Data Transport

RFD – Reduced Function Device

RISC – Reduced Instruction Set Computer

RMST – Reliable Multi-Segment Transport

RREP – Route Reply

RREQ – Route Request

RERR – Route Error

RSA – public cryptography (Rivest, Shamir and Adleman)

RTS – Request to Send

SAA – Stateless Address Autoconfiguration

SHA – Secure Hash Algorithm

SNEP – Secure Network Encryption Protocol

SK – Session Key

SKKE – Symmetric Key Key Exchange

SPI – Serial Peripheral Interface

SPINS – Security Protocols for Sensor Networks

SR – Selective Repeat

SSL – Secure Socket Layer

TCP – Transmission Control Protocol

TESLA – Timed, Efficient, Streaming, Loss-tolerant Authentication

μTESLA – Micro TESLA

TK – Temporal Key

UART – Universal Asynchronous Receiver/Transmitter

UDP – User Datagram Protocol

UMTS – Universal Mobile Telephone System

Wi-Fi – Wireless Fidelity (WLAN)

WLAN – Wireless Local Area Network

WPAN – Wireless Personal Area Network

WSN – Wireless Sensor Network