

M.Sc. Engg. Thesis

Hierarchical Numbering Based
Addressing and Routing Architecture for
Wireless Sensor Networks

by
Md. Yusuf Sarwar Uddin

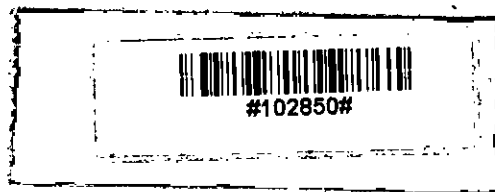
Submitted to



Department of Computer Science and Engineering
in partial fulfilment of the requirements for the degree of
Master of Science in Computer Science and Engineering

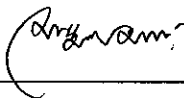
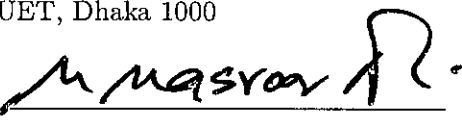
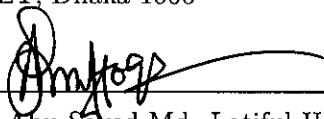
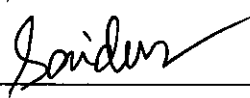
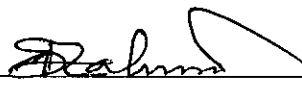
Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology (BUET)
Dhaka 1000

June 2006



The thesis titled "Hierarchical Numbering Based Addressing and Routing Architecture for Wireless Sensor Networks," submitted by Md. Yusuf Sarwar Uddin, Roll No. 040405017P, Session April 2004, to the Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, has been accepted as satisfactory in partial fulfillment of the requirements for the degree of Master of Science in Computer Science and Engineering and approved as to its style and contents. Examination held on June 20, 2006.

Board of Examiners

1. 
Dr. Md. Mostofa Akbar
Assistant Professor
Department of CSE
BUET, Dhaka 1000
Chairman
(Supervisor)
2. 
Dr. Muhammad Masroor Ali
Professor & Head
Department of CSE
BUET, Dhaka 1000
Member
(Ex-officio)
3. 
Dr. Abu Sayed Md. Latiful Hoque
Associate Professor
Department of CSE
BUET, Dhaka 1000
Member
4. 
Dr. Md. Saidur Rahman
Associate Professor
Department of CSE
BUET, Dhaka 1000
Member
5. 
Dr. Md. Saifur Rahman
Professor
Department of EEE
BUET, Dhaka 1000
Member
(External)

Candidate's Declaration

It is hereby declared that this thesis or any part of it has not been submitted elsewhere for the award of any degree or diploma.



Md. Yusuf Sarwar Uddin
Candidate

Contents

<i>Board of Examiners</i>	i
<i>Candidate's Declaration</i>	ii
Abstract	xiii
1 Introduction	1
1.1 Sensor Networks	2
1.2 Research Problems in Sensor Networks	3
1.3 Motivation for the Work	4
1.3.1 Still, Why Global Addresses?	6
1.3.2 Why Address Based Routing?	7
1.4 Problem Statement and Related Works	7
1.5 Design focus of HN-Addressing	9
1.6 Scope of The Work	10
1.7 Overview of the Thesis	10
2 Preliminaries	12
2.1 Wireless Sensor Networks	12
2.1.1 How sensor networks differ from other wireless networks?	14

2.1.2	Sensor Networks Applications	16
2.1.3	Factors of Sensor Network Design	16
2.2	Research Issues in Sensor Networks	20
2.3	Addressing and Routing Protocols for Sensor Networks	22
2.3.1	Addressing Protocols	23
2.3.2	Routing Protocols	27
2.4	Graph Related Terminologies	32
2.4.1	Graph	32
2.4.2	Connected Graph	33
2.4.3	Unit Disk Graph	34
2.4.4	Tree	35
2.5	TreeCast	35
2.5.1	Address Allocation	35
2.5.2	Routing	39
2.5.3	Node Failure	41
2.6	Sensor Network Simulator	42
2.6.1	TOSSIM	42
2.6.2	SensorSim	43
2.6.3	NS-2	43
2.6.4	PARSEC	44
2.6.5	GloMoSim	45
3	The HN-Addressing Protocol	46
3.1	Assumptions	47

3.2	Design Goals	48
3.3	Basic Idea and Methodology of HN-Addressing	49
3.3.1	Addressing without Hierarchical Levels	49
3.3.2	New Hierarchical Addressing Architecture	53
3.4	Address Allocation Technique	55
3.4.1	Address Allocation without Hierarchical Levels	56
3.4.2	Address Allocation with Hierarchical Levels	59
3.5	Node Failure and Address Reallocation	61
3.5.1	Node Failure at Address Allocation Time	62
3.5.2	Node Failure at Event Sensing Time	64
3.6	The Address Allocation Algorithm	65
3.7	Address Parameters	72
3.8	Routing	78
3.8.1	Routing Rules	80
4	Simulation Results	84
4.1	Simulation Setting	84
4.1.1	Simulation Environment and Parameters	84
4.1.2	Energy Consumption Model	87
4.1.3	Node Failure Model	89
4.1.4	Performance Metrics	90
4.2	Performance Evaluation of HN-Addressing	90
4.2.1	Tree Formation	91
4.2.2	Message Cost for Addressing	92

4.2.3	Address Length	94
4.2.4	Communication Energy	99
4.3	Performance on Node Failure	104
4.4	Comparison with <i>TreeCast</i>	105
4.4.1	Message Cost for Addressing	105
4.4.2	Average Address Length	106
4.4.3	Communication Energy	108
4.4.4	Address Allocation Time	109
4.5	Summary	111
5	Conclusion	112
5.1	Major Contributions	112
5.2	Future Research Directions	114

List of Figures

1.1	A typical sensor network.	2
2.1	Sensors are scattered in a sensing field.	13
2.2	Sensor network application and the future trend. (Excerpted from <i>Wireless Sensor Networks: An Information Processing Approach</i> [99])	16
2.3	The components of a sensor node.	19
2.4	Dynamic addressing scheme in [10]. (a) Base stations (b) Handoff.	24
2.5	Distributed Assignment of MAC address scheme in [77]. For node A , B and I are bi-neighbors and C and E are in-neighbors. It should be ensured that B and I different address and C , E do not collide with them. But C and E may have same address.	25
2.6	SPIN Protocol. (a) Node A starts by advertising its data to node B , (b) Node B responds by sending a request to node A , (c) Node B receives the requested data, (d) Node B sends out advertisements to its neighbors (e-f) Neighbors in turn send requests back to B	29
2.7	Directed-Diffusion protocol phases.	30
2.8	A Graph.	33
2.9	A disconnected graph	34

2.10	Unit disk graph (a) Connected (b) Disconnected (Circles are assumed to be of radius 1)	34
2.11	(a) Tree (b) Spanning tree of a graph. (Dark edges belong to spanning tree).	35
2.12	Address Allocation for nodes in <i>TreeCast</i>	36
2.13	Address assignment algorithm of <i>TreeCast</i>	37
2.14	Routing in <i>TreeCast</i> (Arrow shows the direction of the message propagation). (a) Query message: message reaches every node, message bears the forwarder's address (b) Response message: Node 1.4.6.2.4 sends a response to the sink, message forwarded along the reverse prefix match path (c) Control message: sink sends to 1.4.6.2.4, message forwards along the prefix match path	40
3.1	Addressing steps in HN-Addressing. (a) Underlying communication graph (Hex numbers are hardware ID) (b) Formation of spanning tree (c) <i>preorder</i> numbering of nodes.	50
3.2	Addressing of nodes without levels with <i>routing</i> and <i>subordinate number</i>	51
3.3	Routing packet to node 8 from the sink. (<i>dark lines show the path</i>).	52
3.4	Addresses with hierarchical levels for $bpl = 4$	54
3.5	Comparison of two Addressing schemes. (a) Without level (b) With level for $bpl = 2$	54
3.6	Transitions of states of node during address allocation.	58
3.7	Sequence of message passing in address allocation of HN-Addressing.	60
3.8	Reallocation of address. (a) node sends ASSIGN-ADDRESS message to neighbors (b) obtains address from 7.	63
3.9	Reallocation of address. (a) Node 6 fails, (b) Node 7 detects parent failure and sends DISCARD-ADDRESS to neighbors, (c) Sink timeouts for DONE from 6 and assigns address to another child, whereas nodes leave their parent and seek address by ASSIGN-ADDRESS, (d) Nodes get addresses from 6.	64

3.10	Reallocation of address. (a) Addressing for $bpl = 2$ (b) Node 1.2 fails and children get orphan (c) Orphan nodes get address from 1.1.1 (d) Node 1.2 revives and adjusts its address.	66
3.11	State diagram of entire address allocation procedure in HN-Addressing. . .	67
3.12	Various address parameters in HN-Addressing.	73
3.13	Demonstration of property 3.7.4. (a) Condition I: y gets address from x in the last address level (b) Condition II: y gets address from x in the next address level	75
3.14	Detecting parent nodes (a) Bare nodes (b) Underlying tree (Number inside {} means depth).	76
3.15	Illustration of Property 3.7.5. (b) Possible nodes in the subtree of 1.4.3(5)[2].	77
3.16	Flooding query packet in the entire network.	81
3.17	Sink sending request packet to node 1.3. (Packet does forward after the node indicated by cross)	82
3.18	Node 3.2 sends response packet to the sink.	83
4.1	Scenario of 200 nodes in $1000m \times 1000m$ area. (a) Nodes are scattered randomly in the area (b) Generated Tree for the scene (Squared node indicates <i>sink</i>).	86
4.2	First Order Radio Model for sensor node.	87
4.3	Depth of nodes against the distance from the <i>sink</i>	91
4.4	Message Cost for Addressing in HN-Addressing. $HN(x)$ indicates HN-Addressing with $bpl = x$	92
4.5	Parent fails to assign address to child. (a) Parent 2.1 cannot give address to one of its child node, (b) child changes the parent and gets address 2.1.2.1 from 2.1.2.	94
4.6	Address level scenario for 10 nodes. (a) $bpl = 2$ (b) $bpl = 3$	95

4.7	Depth vs. Level for 200 nodes. (a) $bpl = 3$ (b) $bpl = 5$	96
4.8	Distribution of nodes against depth and level. (In legend, number in parenthesis indicates the bpl value).	97
4.9	Impact of bits per level (bpl) on average address length on varying network size.	98
4.10	Impact of bits per level (bpl) on average address length for a network of size 200.	99
4.11	Impact of number of nodes on average address length.	99
4.12	Packet fields of HN-Addressing (field lengths are measured in bits).	100
4.13	Energy consumption of packet routing in HN-Addressing with varying network size (b) for network of size 200.	102
4.14	Energy consumption of packet routing in HN-Addressing for a network of size 200.	103
4.15	Average address length against node depth in case of node failure.	104
4.16	Initial message cost for address allocation in HN-Addressing and <i>TreeCast</i>	106
4.17	Address allocation scenario for 10 nodes by HN-Addressing and <i>TreeCast</i>	107
4.18	Average address length in HN-Addressing and <i>TreeCast</i>	108
4.19	Energy/packet for HN-Addressing and <i>TreeCast</i>	109
4.20	Address allocation time for a network of size 200 with $bpl = 5$	110
4.21	Address allocation. (a) HN-Addressing: address allocation is going on in subtree A under x , until node x returns $DONE(l)$ message to its parent, no node in subtree B under node y can have address, (b) <i>TreeCast</i> : if both x and y are approved with their addresses, both subtree A and B can be active in address allocation.	111

5.1 Hypothetical addressing approach with greater energy saving. (a) Currently proposed HN-Addressing without leveling, (b) *TreeCast*, (c) Future approach. 115

List of Tables

2.1	Applications of Sensor Networks.	17
2.2	Research issues in sensor networks (Communication architecture).	21
2.3	Research issues in sensor networks (Information Processing and operation).	21
2.4	Current Research projects on sensor networks.	22
4.1	Simulation Setting.	85
4.2	Average address length calculation for Figure 4.6.	95
4.3	Calculation of average communication cost for the scenario at Figure 4.6.	102
4.4	Average address length calculation for Figure 4.17	107
4.5	Summary of comparison between HN-Addressing and <i>TreeCast</i>	111

Acknowledgments

All praises due to Allah, the most benevolent and merciful.

I express my heart-felt gratitude to my supervisor, Dr. Md. Mostofa Akbar for his constant supervision of this work. He helped me a lot in every aspect of this work and guided me with proper directions whenever I sought one. His patient hearing of my ideas, critical analysis of my observations and detecting flaws (and amending thereby) in my thinking and writing have made this thesis a success.

I would also want to thank the members of my thesis committee for their valuable suggestions. I thank Professor Dr. Muhammad Masroor Ali, Dr. Md. A. S. M. Latiful Hoque, Dr. Md. Saidur Rahman and specially the external member Professor Dr. Md. Saifur Rahman.

Besides my supervisor and board members, I also thank Mr. Salahuddin Mohammad Masum, an ex-thesis student of my supervisor, for his IEEE and ACM library account that I frequently used in my every day of study to download necessary papers from there. Those papers played a vital role in my research. I also thank one of my friend Mohammad Ashiqur Rahman, who was my partner in the early works of my research. We jointly wrote a paper on the very basic idea of this work and got the paper published in a conference, ICNEWS'06.

In this regard, I commemorate my late father, and remain ever grateful to my beloved mother who always exists as a single source of inspiration behind every success of mine I have ever made. I also remember two of my brothers for their consistent support in my everyday living.

Abstract

Wireless sensor network is a collection of large number of sensors, some tiny battery-powered sensing devices equipped with short range radio, deployed in a geographic area to perform some distributed sensing task over the target region. For the collaboration of the sensing activity sensors are networked among themselves through a multihop communication paradigm. Due to some well-established constraints, addressing of nodes and data routing techniques in sensor networks differ significantly from other networks. In this thesis, we present an architecture for efficient addressing and routing in sensor networks. We propose a dynamic and globally unique address allocation scheme for sensors in such a way that these addresses can later be used for data routing. We build a tree like organization of sensor nodes rooted by the sink node based on their transmission adjacency and then set label on each sensor with a number according to the preorder traversal of the tree from the root. In this addressing process, each sensor keeps necessary information so that they can route data packets later on depending on these addresses, rather than keeping large routing table or previous routing states. We conduct simulations to measure the soundness of our approach and make a comparison with another similar technique.



Chapter 1

Introduction

Wireless networking has become an essential part of technologies, allowing nomadic users to keep in touch with their family or office using miscellaneous devices such as laptops, PDA's or mobile phones. The most deployed wireless technologies, known as cellular network, wireless LAN or WiFi, are still restrictive as users must be within the range of correctly configured access points. People dwelling areas like homes, offices, airports or train stations can be equipped with needed infrastructure if needed, but it is not the case for other areas where human reachability may not be possible. In that case infrastructureless and self-functioning multi-hop wireless communication may be desirable. In recent years multihop wireless ad hoc and sensor networks are getting popular in wireless technology world and have been (and still being) studied extensively in recent years. These networks are composed of a set of hosts operating in a self-organized and decentralized manner, which can communicate together using a radio interface. Sensor network, one sort of multihop ad hoc network, is an emerging network technology in current time and is getting more and more attention from many researchers all over the world for its tremendous applications in various aspects of life.

1.1 Sensor Networks

A sensor network is a collection of many battery-powered low-cost sensing devices, commonly known as sensors, which collaborate to perform some distributed sensing task in an environment. These sensors have power supply unit, sensing equipments to collect information, central processing unit and radio transceiver to communicate with other sensors. Due to the advancement in microelectronics and small scale fabrication, production of these small sensing devices with computational and communication units has become technically and economically feasible. A sensor node senses an event or measures an ambient condition of the environment (like temperature, pressure, humidity, etc) or detects the presence of an object or such other physical attributes in its vicinity, and as a response of the event it generate some data and transmit it to a remote base-station that makes the necessary aggregated interpretation of the events.

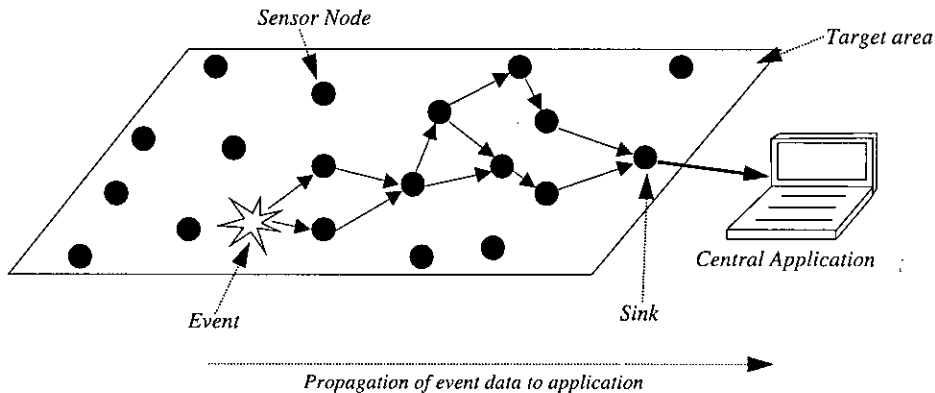


Figure 1.1: A typical sensor network.

Though individual sensors can do very little of their task, large number of sensors can be coordinated into an organized network to perform a big sensing task that might have some significant interpretation in its context. Sensors are normally battery-powered, and when deployed they remain in the subject area unattended for long time. Battery can hardly be recharged or replaced in case power drain-out. So sensors are by born constrained by energy. Due to this constraint, they cannot transmit signal to a very long

distance to reach a remote base-station or sink, rather they form a multihop communication scenario with their short range radio equipments, transmitting data each time to the immediate neighbors and thus fusing it into the entire network. Figure 1.1 shows a typical sensor network. A good overview of sensor networks is presented in [72].

Sensor network has vast application domains: from home to industry, from calm rural environment to battlefields, from underwater life to highest mountains. Section 2.1.2 narrates some potential applications of sensor network.

1.2 Research Problems in Sensor Networks

Sensor network is a very hot research area in recent years. Potential research activities focus on the collaboration among sensors in data actuation and processing as well as the coordination and management of sensing activity in the subject area. Some active research issues in sensor networks, as found frequently in the literature, are as follows:

- *Addressing*: Addressing is a big issue in sensor networks. Since sensor networks do not have any infrastructure like wired or cellular networks and huge number of sensors is deployed in absolutely ad hoc fashion, it is not possible to adopt any network wise ID or address assignment to sensor nodes. Sensor networks should be self-organized; hence dynamic ID assignment can be the only practical solution.
- *Routing*: Routing in sensor networks is another active research issue. The challenges of routing in sensor networks are due to small radio transmission range of sensors. A tight constraint for routing is power. Since sensors are battery driven and have little provision for recharging during their life time, they have extreme energy limitation and this requires almost every protocol for sensor networks to be energy-efficient. Energy-efficient routing for sensor networks is an excellent research topic.
- *Data aggregation*: Sensors perform sensing task in an area and an event can be sensed by several sensors at the same time. After the event, sensors in the vicinity

of the event may send data to some remote base-station reporting the sensed event. This makes transmission of same data from several sources to the single sink and thus wasting away valuable sensor energy. Data aggregation restricts this sort of extra transmission and tries the best non-redundant data transfer.

- *Clustering*: Sometimes few sensors can be accumulated together in a small group under the guidance of a single sensor for some special purpose. This phenomenon is called clustering and can be used in energy-efficient routing and effective data accumulation.
- *Data coordination*: Coordination among sensor nodes in gathering and distribution of data of an event (e.g., detecting a moving object) is another interesting issue.
- *Coverage* Coverage of sensing area by the sensors is another important point to address. Coverage problem seeks optimum number of sensors and their placement in an area to cover the entire area efficiently.
- *Protocols and architecture*: Designing energy-efficient and scalable MAC, network, transport and application protocols for sensor networks are drawing huge attention from many researchers around the world.

Section 2.2 enumerates the popular research issues in sensor network with citations.

1.3 Motivation for the Work

In this thesis we concentrate on two important networking issues for sensor networks: addressing and routing. It is needless to say how important these two things are for any network. Addressing deals with the problem of allocating unique ID to sensor nodes that is used to locate a certain destination for data packet in the network, whereas routing finds the best route for a packet to reach a specific destination. In our thesis, we combine addressing and routing matters of a sensor network into a single architecture. We develop

a network-wise unique address assignment scheme that supports address based *stateless* routing technique for sensor networks. We name our technique as Hierarchical Numbering based Addressing and Routing, in short HN-Addressing.

Two types of node addresses are possible in a wireless network: local MAC addresses and global network addresses.

MAC addresses are used to identify a neighbor node within the transmission range of a node. Since wireless channel is by nature a broadcast channel, i.e., when a packet is destined to a specific neighbor node, it is listened by other nodes within the transmission range. MAC addresses identify the node that is actually intended for the packet. These addresses can be locally assigned and should not be unique throughout the entire network as far as they are distinct in their neighborhood.

Network address acts as the global identification of nodes that is used for data routing to a specific node that remains several hops away from the source. Usually in a network each node has some unique address and routing protocols normally selects the path for a packet to reach a destination by examining the destination address. Globally unique network wide addresses enable packet communication from any node to any node.

Though network wide addressing of sensor nodes is possible, it is however widely suggested that the global network address need not to be used for routing or identifying the final network destination [72] in sensor network, because, in general, queries in sensor network are not directed toward specific sensors. Instead of the network address, attributes like sensor location or event reading can be used as network destination, and these attributes are used for routing as in [45], [35]. This sort of routing is called data centric routing that is based on the key fact that *data itself is more important than the identity of the sender*. This makes the matter of global addressing little fake for sensor networks.

1.3.1 Still, Why Global Addresses?

Still there can be some applications where allocation of network wide unique address may be necessary for administrative tasks like configuration of the network, monitoring and controlling of specific sensors tasks [25]. Consider the following scenario:

A sensor network is deployed in an area to measure the average temperature level in a chemical plant. For this task, lots of sensors are deployed in the plant area and each sensor is equipped with a temperature measuring device like thermometer. Every sensor measures temperature in its vicinity and reports the temperature reading to the remote sink, that actually computes the overall temperature level in the zone and sends the information to some central application thereafter. As far as the average temperature measurement is concerned, the sink node does not need to know (or even ever try to know) which sensors are sending what temperature reading; what is necessarily important is the temperature values themselves. This deems the need for global addresses, but the need for local MAC addresses is still alive, to select next hop in the multihop communication. But what happens, when a certain sensor is affected by the malfunction of its thermometer and it generates continuously irritating and erroneous reading, leading the entire temperature calculation to a jeopardy. If the sink node detects this anomaly of operation, how can it issue a command to the affected node to stop it sending temperature reading to the sink unless the sink can identify the node by some address?

So sensor nodes should have global addresses. It can be argued that sensor nodes can have embedded hardware ID with their equipment while having been manufactured in the factory. These addresses may have two types of problems, firstly it may be long (48 bit of usual Ethernet NIC MAC addresses) causing packet sending/receiving to be costly in terms of energy consumption and secondly it is redundant due to small network size compared to the total address bits. Hence nodes should be assigned with short addresses according the size of network. Again hundreds of sensor nodes combined into a single network are deployed to perform a common task, doing manual assignment of unique address to every node is quite cumbersome and practically not feasible. So address assignment should be dynamic and distributed.

1.3.2 Why Address Based Routing?

Again, assigning network wide unique addresses in a sensor network of thousand nodes is quite costly in terms of energy consumption. So just for identifying a specific node in the network which may not be very frequent in sensor application, address allocation in the entire network may not be feasible. Hence addressing should provide any other strong functionality in the network rather than identifying nodes merely. We target the issue in our work and therefore design a distributed globally unique address assignment scheme for sensor nodes that not only provide unique identification to each individual node, but also bears the routing information along with addresses. We dynamically assign addresses to sensor nodes and after the assignment of addresses, these addresses can be used for data routing in the network. We propose an addressing and routing architecture for sensor networks.

In our thesis, we design an address based *stateless* routing paradigm for sensor network. The routing technique is *stateless* in the sense that unlike other routing protocols it does not store bulk routing information like routing table or previously learned routing paths. Since sensor nodes are constrained by energy and processing power, stateful routing technique with long routing table (as there are large number of nodes) or huge computation in determining routes from earlier routing paths, may not be desirable for sensor network. We assign addresses to nodes and these addresses are sufficient enough for data routing.

1.4 Problem Statement and Related Works

We address the problem of addressing and routing in sensor networks. Network consists of large number of sensor nodes sparsely deployed in an area, where nodes have very little knowledge of entire network topology or total connectivity scenario, rather they know only their immediate neighbors whom they can reach to by their radio directly. There would be a special node, called sink, that connects the sensor network to other communication

infrastructure, such as wired network. Nodes sense event in their vicinity and send their observed data to the sink. Sink gathers data from sensors, aggregates them and transmits the final result to the central application where possible interpretation of the sensing observations is made.

When just deployed, sensor nodes come in void, nothing with them. After deployment, a network initialization procedure is conducted and at that time an address allocation process is initiated. This address allocation scheme, generally initiated by the sink, assigns unique addresses to each sensor node in the network and once address assignment is done, nodes can route data to any destination depending on these addresses. Data routing neither requires any routing table in nodes nor makes extra communication or computation to find routes.

Addressing and routing architectures for sensor networks do not frequently occur in the literature. Very little works have been done over this issue so far. A recent work by Ould-Ahmed-Vall *et.al.* [66] at GaTech provides a solution to the problem of dynamic ID assignments to nodes in sensor networks. In this work, it is assumed that sensor nodes do not have any sort of prior addresses (not even any hardware ID) and the protocol assigns some temporary ID to each sensor at first and then finally gives actual network ID. This ID assignment technique uses some about fourteen types of messages among sensors and runs in three phases. The scheme allocates unique global addresses to sensors, but does not contain any routing directive with it.

TreeCast, proposed by PalChaudhuri *et. al.* [67] is a *stateless* addressing and routing architecture for sensor networks and this work has some resemblance to our result. In *TreeCast*, sensors are assigned addresses based on their level in the network. Level of a sensor node is measured as the number of hops required to reach the sink from that node. In *TreeCast*, nodes are organized in a tree and sensors at certain level in the tree are assigned addresses from the parent in the immediate earlier level. Whenever a child gets address from its parent, it chooses a local address for itself and appends the local address with its parent's address constructing its own global address. *Stateless* routing can be done

based on the assigned addresses. Section 2.5 is completely dedicated to *TreeCast* and gives the detail description of *TreeCast*. To our best knowledge, ours scheme, HN-Addressing, is going to be the second of this kind of routing architecture for sensor networks.

1.5 Design focus of HN-Addressing

While designing the addressing and routing architecture, HN-Addressing, we primarily focus on the following considerations:

- Since this is an addressing scheme, it should require that all nodes would get addresses uniquely ascertaining no two nodes get the same address. Addresses should never be duplicated.
- Addressing time should be finite and must end in some moderate time. No node should wait indefinitely to collect address from nodes, and there should not be any deadlock.
- Address allocation technique is completely distributed and each sensor runs the protocol in them.
- Considering the failure of nodes while assigning addresses to nodes, the protocol is self-configurable and would be completely functional amid of node failures.
- A routing paradigm based on node addresses would be designed. In all aspects, we find an energy-efficient solution.
- The main concentration of our approach is the address length. Since address length plays a significant role in data communication, we always try to keep the node addresses as short as possible. In contrast to *TreeCast*, HN-Addressing can accommodate addresses in fewer bits than *TreeCast*. In the simulation study, as described in Chapter 4, we find that average address length is far reduced in HN-Addressing than that of *TreeCast* and it saves much energy in packet transmission.

1.6 Scope of The Work

The main task of this thesis work is to design a distributed global address assignment protocol for sensor networks with an in-built *stateless* routing facility. The primary concentration in designing the protocol, is the address length. As node address in small data packet makes considerable overhead, reduction of address bits is always desirable for energy-efficient data communication. In our protocol we realize this factor. We make simulation based experiments of our scheme using PARSEC [6] and describe our observations in Chapter 4. We evaluate the performance of our technique by varying various parameters (number of nodes, node failure, bits per level, etc). We also compare our approach with *TreeCast* in terms of message cost, average address length, communication energy, addressing time and so on. Experiments reveal that in all performance metrics except the initial address allocation time, our approach outperforms *TreeCast* with a very big margin, thus placing HN-Addressing in a dominating position over its counterpart.

Implementation or developing a working system with the proposed architecture and its associated protocols is beyond the scope of this thesis. The author neither does verify the architecture for a real scene nor does carry any practical experiments with the actual deployment of sensor nodes in a real environment. All observations are made based on simulations, in an artificial replication of the real world with lots of abstractions, and the results are obviously subject to deviation in case of real happenings.

1.7 Overview of the Thesis

The rest of the chapters are organized as follows. Chapter 2 gives a preliminary description of some terminologies and concepts related to wireless networks and sensor networks that may be helpful to understand the context of this thesis. Chapter 3, the main chapter of this dissertation, illustrates our proposed addressing and routing architecture for sensor networks. It delineates the address allocation technique and subsequent routing paradigm

of our proposed scheme. Chapter 4 contains the simulation results, performance evaluation of our scheme and a comparative study against *TreeCast* in several performance issues. Chapter 5 draws the conclusion mentioning the key contributions of this thesis followed by some future research directions.

Chapter 2

Preliminaries

2.1 Wireless Sensor Networks

A wireless sensor network is a group of sensing devices that are linked by wireless communication system to perform a distributed sensing task. These devices are manufactured of micro-electro-mechanical systems (MEMS) components such as sensors, actuators, wireless communication equipments and CMOS building blocks. The devices are popularly known as wireless integrated network sensors (WINS) or in short sensors [72]. Sensors combine microsensing technology, low power computing and wireless networking in a compact system. Recent advances in micro-fabrication has made possible to integrate all these in a single chip. Sensors are randomly dispersed over the area of interest and are capable of RF communication, and contain signal processing engines to manage the communication protocols and for data processing. As an individual, sensors may have limited capacity, but they are capable of performing a big task through a coordinated effort in a network that consists of hundreds or thousands of such nodes.

A sensor network is composed of large number of sensor nodes that are deployed densely either inside the phenomenon or very close to it. Figure 2.1 shows the operation of a typical sensor network. Sensor networks have the following features:

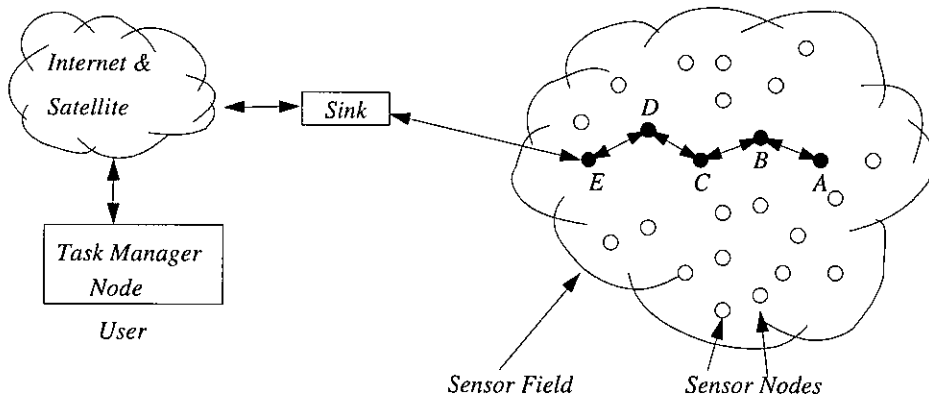


Figure 2.1: Sensors are scattered in a sensing field.

- *Ad hoc deployment:* The deployment of sensor nodes is completely ad hoc and random. The position of sensor nodes need not be pre-determined allowing their deployment in inaccessible terrain or disaster relief where human reach may not be possible.
- *Distributed and self-organizing protocols:* Since sensors are networked to perform some distributed collaboration tasks, the protocols that run on sensors should be distributed in nature, rather than central. Again due to random deployment and unattended operation, no manual configuration or control of sensor operation can be possible. So, the protocols should be self-organizing.
- *Multihop communication:* Since large number of sensor nodes are densely deployed, neighbor nodes may be very close to each other. Hence, multihop communication in sensor networks is expected to consume less power than the traditional single hop communication. Furthermore, the transmission power levels can be kept low, which is highly desired to prolong operation life. Multihop communication can also effectively overcome the signal propagation effects experienced in long-distance wireless communication.
- *Power constraint:* Sensor nodes usually carry limited power sources. Since sensors work in an unattended fashion, a depleted power source can hardly be replaced or

recharged. While traditional networks aim to achieve high quality of service (QoS) provisions, sensor network protocols must focus primarily on power conservation. They must have in built trade-of mechanisms that give the end user the option of prolonging network lifetime at the cost of lower throughput or higher transmission delay.

2.1.1 How sensor networks differ from other wireless networks?

Sensor networks significantly differs from other traditional wireless network models; namely, mobile ad hoc networks, cellular networks and a number of short range wireless local area networks.

Mobile ad hoc network (MANET) is a peer-to-peer network which usually comprises of tens or hundreds of communication nodes that are able to cover ranges of up to hundreds of meters. Each node is envisioned as a personal information appliance such as a personal digital assistant (PDA) outfitted with a fairly sophisticated radio transceiver. The nodes are fully mobile. A MANET aims to form and maintain a connected multi-hop network capable to transporting multimedia traffic. MANET protocols put focus on throughput/delay characteristics in the face of high mobility, and energy consumption is of secondary importance here, since each device is attached with a person and the depleted battery can easily be replaced or recharged when needed.

Cellular network is a network [85] that consists of both stationary and mobile nodes. The stationary nodes (called base stations) are connected in a subnetwork with a wired backbone, forming a fixed infrastructure. Mobile nodes greatly outnumber the number of base stations (tens to hundreds of mobile nodes per base stations), which are positioned quite sparsely. Each base station covers a large region (usually called cell) and mobile nodes are always one hop away from the base stations. The issue that encounters here is handoff of mobile nodes as they move from one cell to another cell. The primary goal

is high bandwidth utilization with satisfactory QoS. The base stations have effectively unlimited power supply whereas mobile nodes are battery powered.

Bluetooth [11] is a short-range wireless networking system intended to replace the cable between electronic devices and provide radio communication between them. Normally in a Bluetooth enable system, electronics devices form a *piconet*, that consists of a *master* node and up to seven other *slave* nodes. Each *piconet* uses TDMA (time division multiple access) or frequency hopping to transmit data among themselves. The *master* node coordinates data transfer among *slave* nodes and all nodes are synchronized to the *master*. Typical data transfer rate is 1Mbps, transmission energy is about 1mW and transmission range is nearly 10m.

HomeRF [43] is a recently developed short-range commercial wireless network system. The goals are similar to Bluetooth with higher data rate and longer transmission range, but it operates based on IEEE 802.11.

In contrast to all these networks, sensor network potentially comprises of hundreds or thousands of nodes. These nodes are stationary after deployment and there is hardly any mobile node. The data traffic is considerably low requiring effective data transmission rate on the order of 1-100kbps [72]. Conventional networks concentrate on throughput, bandwidth utilization and QoS, but for sensor network the primary interest is in prolonging the lifetime of the network by conserving energy and for this we can give up other performances like delay, QoS, bandwidth utilization and others. Each node is battery powered and has little chance to be replaced with newer one when deployed in hostile or remote region. So energy constraint is a very big issue for sensor networks and energy efficiency has always been considered as the single most important design challenge in sensor networks.

2.1.2 Sensor Networks Applications

Sensor networks may consist of many different types of sensors such as seismic, magnetic, thermal, acoustic, visual and infrared, which are able to monitor a wide variety of ambient conditions like temperature, humidity, vehicular movement, illumination condition, pressure, moisture, noise level, radiocivity, toxicity, mechanical stress or bend, presence or absence of certain objects and so on. Enabling wireless communication and power of computation with micro-sensing activity has made versatile applications of sensor networks. Table 2.1 shows some potential and most cited applications of sensor networks and Figure 2.2 presents potential applications with future trend.

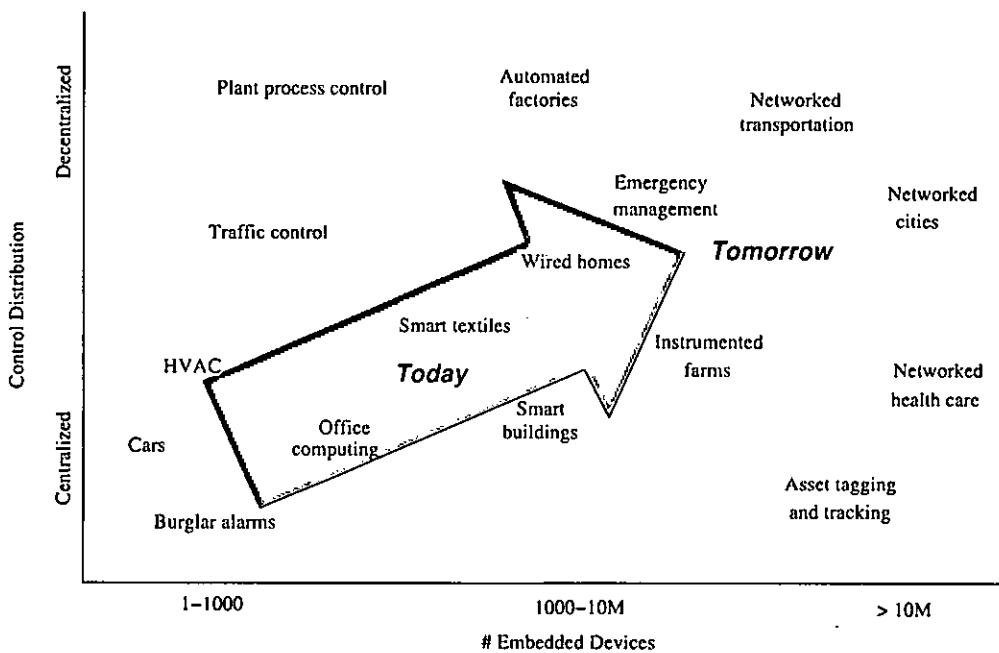


Figure 2.2: Sensor network application and the future trend. (Excerpted from *Wireless Sensor Networks: An Information Processing Approach* [99])

2.1.3 Factors of Sensor Network Design

The design of a sensor network is influenced by many factors which include *scalability*, *fault tolerance*, *hardware constraints*, *operating environment*, *network topology*, *transmis-*

Table 2.1: Applications of Sensor Networks.

<i>Area</i>	<i>Applications</i>	<i>Citations</i>
Environment	Weather monitoring and forecasting	[16, 22, 49, 91]
	Biocomplexity mapping of the environment	
	Flood detection	
	Forest fire detection	
	Pollution study	
	Habitant monitoring	
Health	Telemonitoring of human physiological data	[12, 65]
	Tracking and monitoring doctors and patients	
	Drug administration in hospital	
Home and Office	Smart environment	[28, 99]
	Vehicle tracking and detection	
	Security and surveillance	
	Asset and warehouse management	
Industry	Automotive	[99]
	Industrial process control	
Military	Monitoring forces, equipments and ammunition	[1, 99]
	Battlefield surveillance	
	Reconnaissance of opposing forces and terrain	
	Battle damage assessment	
	Biological and chemical attack detection	

sion media and *power consumption*. These factors are important because they serve as guidelines to design a protocol or algorithm for sensor networks. An elaborate description of these factors are presented in [1]. We give a brief outline of some factors below:

Scalability: The number of sensor nodes deployed in studying a phenomenon is in the order of hundreds or thousands. Depending on the application, the number can reach up to million. The density of nodes can range from few sensors to few hundreds per unit area or volume. The node density $\mu(R)$ can be calculated as:

$$\mu(R) = \frac{N\pi R^2}{A}$$

where N is the number of total sensors in a region of area A ; R is the radio transmission range. $\mu(R)$ gives the number of sensors within the transmission range of each sensor in region A . Node density can be 5-100 sensors per transmission range depending on the sensor application.

Fault tolerance: Node failure due to lack of power, physical damage or environmental interference is very frequent event in sensor networks. The failure of nodes should not affect the operation of sensor network operations. Fault tolerance is the ability to sustain network functionalities without any interruption due to sensor nodes failure [42, 80]. The reliability $R_k(t)$ or fault tolerance is modeled in [42] using Poisson distribution to compute a probability of not having a failure within the time interval $(0, t)$:

$$R_k(t) = \exp(-\lambda_k t)$$

where λ_k is the failure rate of sensor node k .

Sensor hardware: A sensor node is made up of four basic components as shown in Figure 2.3: a *sensing unit*, a *processing unit*, a *transceiver unit* and a *power unit*. *Sensing unit* is usually composed of two sub units: sensors and analog to digital converters (ADCs). The analog signals produced by the sensors based on the observed phenomenon are converted to digital signals by the ADC, and then fed into the *processing unit*. The *processing unit*, which is generally associated with a small storage unit, manages the procedures that make the sensor node collaborate with the other nodes to carry out the assigned sensing tasks. *Transceiver unit* connects the node to the network. *Power unit* supplies power to the node and may be supported by a power scavenging unit such as solar cells. There may be some application dependent sub units. Most of the sensor network routing techniques and sensing tasks require the knowledge of location with high accuracy. Thus, it is common that a sensor node has a *location finding system*. A *mobilizer* may sometimes be needed to move sensor nodes when it is required to do so.

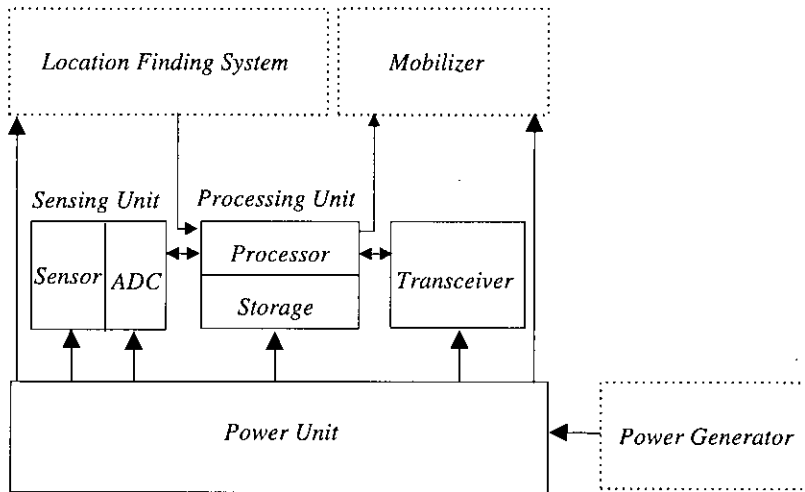


Figure 2.3: The components of a sensor node.

Transmission media: In sensor network, sensors are linked by wireless medium. Possible options for wireless media can be radio, infrared or optical. In case of radio links, ISM (Industrial, Scientific and Medical) bands can be used which are license-free in most countries. A small-size, low-cost, ultra-low powered transceiver is usually preferred. According to [71], due to hardware constraints and trade-off between antenna efficiency and power consumption, good choice of carrier frequency for transceiver is ultrahigh frequency (UHF) range. μ AMPS [81] uses Bluetooth-compatible 2.4GHz transceiver with an integrated frequency synthesizer. Another possible mode of inter node communication is infrared which is also license-free, robust to interference and cheap. Only problem is the requirement of line of sight (LOS) path between sender and receiver. Smart dust mote [46], a autonomous sensing and computing device, uses optical media for communication. Optical communication system has two transmission schemes [93], active communication with a laser diode and steerable mirrors and passive communication using a corner-cube retroreflector (CCR). Choice for transmission media is sometimes critical to application. For example, for marine application in aqueous environment, we may need long-wavelength radiation for penetrating the water surface, whereas in battlefield or inhospitable terrain, channels might be more error-prone and subject to interference.

Power consumption: The sensor nodes, being a micro-electronic device, can only be equipped with a limited power source (< 0.5 Ah, 1.2 V). In many application power replacement is impossible, so network life is solely dependent on battery lifetime. In a multihop ad hoc network, nodes play the role of both data originator and router and failure of nodes causes topology changes and necessitates rerouting of data packets. So energy conservation is crucial in sensor network. A sensor node consumes power in three domains: sensing, communication and data processing. Sensing power varies with the nature of application and physical properties to be sensed. Sporadic sensing might consume less power than continuous monitoring. Sensor nodes expend maximum energy in data communication. Both transmission and receipt of data packets dissipate energy. This energy consumption depends on the propagation model of the channel, radio parameters (amplification co-efficient, energy per bit, etc), transmission distance, data rate and packet length. Energy expenditure for data processing is much less compared to data communication. The experiment in [72] reveals that assuming Rayleigh fading and fourth power distance loss, the energy for transmitting 1KB data to a distance of 100m is approximately the same as for executing 3 million instructions by a 100 million instructions per second (MIPS)/W processor. So for an energy-efficient protocol design, reducing the energy for data communication can be of primary issue.

2.2 Research Issues in Sensor Networks

Sensor network is a very hot research area in recent years. Many researchers around the globe are conducting researches on various aspects of this type of networks. Depending on the objective of the works, researches on sensor networks can have two major directions: *communication architecture* - concentrating on layer based communication architecture for sensor networks and *information processing and operations* - focusing mainly on managing sensors' operation and their information gathering and processing activities subject to the constraints discussed earlier. Tables 2.2 and 2.3 present several significant works on these

Table 2.2: Research issues in sensor networks (Communication architecture).

<i>Communication Layer</i>	<i>Research Focus</i>	<i>Citations</i>
Application layer	Sensor management, task aiming and data advertisement, sensor query and data dissemination	[26, 27, 80, 81]
Transport layer	End-to-End communication, reliability, acknowledgment issues	[5, 9, 72, 74]
Network layer	Energy-efficient data routing, network level addressing, data centric routing, attribute based naming, route discovery and maintenance, stateless routing	[26, 28, 34, 36, 37, 45, 62, 76, 80, 84]
Data link layer	Medium access, channel assignment, energy-efficient protocols, MAC addressing, error control	[30, 50, 63, 81, 83, 84, 98, 100]
Physical layer	Signaling, encoding, modulation, propagation effect, fading, energy-efficient physical transmission media	[21, 23, 53, 54]

Table 2.3: Research issues in sensor networks (Information Processing and operation).

<i>Research topic</i>	<i>Research Focus</i>	<i>Citations</i>
Addressing	Dynamic address allocation to sensors, MAC addressing, attribute based naming	[3, 10, 66, 77, 90]
Routing	Energy-efficient data routing, network level addressing, data centric routing, attribute based naming, route discovery and maintenance, <i>stateless</i> routing	[37, 45, 52, 57, 59, 60, 67, 86]
Coordination	Coordination among sensors for collaboration, tracking events, data sharing	[32, 36]
Data aggregation	Aggregation of data at sensing nodes, energy-efficient data aggregation, data fusion	[32, 73]
Clustering	Dynamic cluster formation, cluster based data routing and data fusion	[37, 38, 56]
Coverage	Coverage and connectivity maintenance problems, area coverage by minimum sensors	[19, 31, 61, 92]
Protocol design	Application-specific protocol design, cross layer architecture and protocols	[17, 37, 38, 39, 41, 68, 79]

issues whereas Table 2.4 lists some ongoing research projects on sensor networks around the world.

Table 2.4: Current Research projects on sensor networks.

<i>Project name</i>	<i>Research area</i>	<i>URL</i>
SensoNet [2]	Transport, network, data link and physical layers, Power control, mobility and task management planes	http://www.ece.gatech.edu/research/labs/bwn/
WINS [28, 72]	Distributed network and Internet access to sensors, controls and processors	http://www.janet.ucla.edu/WINS/
SPIN [36]	Data dissemination protocols	http://nms.lcs.mit.edu/projects/leach
SPINS [70]	Security protocols	http://paris.cs.berkeley.edu/perrig/projects.html
SINA [80, 88]	Information networking architecture	http://www.eecis.udel.edu/cshen/
μ AMPS [81]	Framework for implementing adaptive energy-aware distributed microsensors	http://www-mtl.mit.edu/research/icsystems/uamps/
LEACH [37]	Cluster formation protocol	http://nms.lcs.mit.edu/projects/leach
Smart Dust [44, 46]	Laser communication from a cubic millimeter, mote delivery, micro-watt electronics power sources, macro motes (COTS Dust)	http://robotics.eecs.berkeley.edu/pister/SmartDust/
SCADDS [14]	Scalable coordination architectures for deeply distributed and dynamic systems	http://www.isi.edu/scadds/
Dynamic sensor networks [24]	Routing and power aware sensor management network services API	http://www.east.isi.edu/div10/dsn/
AwareHome [40]	Requisite technologies to create a home environment that can both perceive and assist its occupants	http://www.cc.gatech.edu/fce/ahri
COUGAR [12]	Distributed database and query processing	http://www.cs.cornell.edu/database/cougar/index.htm

2.3 Addressing and Routing Protocols for Sensor Networks

Addressing and routing are two major communication issues in sensor networks. Addressing deals with allocating unique addresses to sensor nodes in a network, whereas routing

finds the route (by selecting the next node in neighbor) for a packet to be delivered to a specific destination. In traditional networks, network layer addresses are used to make routing decision, but in sensor networks data packet may not be destined to a specific sensor by the address, rather it may be targeted by some attributes. In sensor application, data itself is more important rather than the identification of the sender. Still, some address based routing approaches appear in literature.

2.3.1 Addressing Protocols

Each node, in sensor network, is typically assigned a network wide unique global network address that is used for administrative tasks like configuration of the network, monitoring of individual sensors and downloading binary code or data aggregation descriptions to specific nodes [25]. The MAC address is used to identify the next-hop sensor node during packet routing. Each node, after receiving a packet, determines the next-hop MAC address by examining the local routing table, and updates the next-hop address of the packet. This process continues till the packet is delivered to the destination node specified by the destination address of the packet. Two major addressing schemes frequently occur in the literature: MAC or local addressing and network or global addressing. We give a short overview of various addressing schemes for sensor networks below.

Dynamic addressing scheme for wireless media access, proposed by Bharghavan in [10], is an early work on address allocation to wireless nodes where nodes do not have any preset global unique ID, rather they are assigned dynamic MAC address during the initialization phase of the network setup. The basic idea with the approach is that in wireless network every node in the network should not have unique IDs for their data communication. Since a packet sent by a node is seen by only its neighbors (nodes that lie within its transmission range), for identifying sender and receiver for a packet, these neighbors should have distinct addresses. So addresses can be spatially reused enabling two remote nodes to have same ID and it requires fewer bits than globally

unique addresses. Addresses can also be temporally reused when any node dies or leaves the system. Assuming maximum 10 nodes under a base station, the scheme uses 4-bit number to address nodes.

The network contains some static base stations and every node belongs to a suitable (possibly nearest) base station (Figure 2.4(a)). Each base station is assigned unique address during network setup and every node can uniquely identify its base station in its locality. A base station assigns a dynamic MAC address to a node when it first enters into the cell. This MAC address is chosen in such a way that it does not collide with other neighbors of this base station. A node can either send packet to its base station or receive packet from its base station. When a node tries to send data to a node under another base station, data transfer is handled by the respective base stations. Each packet bears the sender and receiver address along with the direction of packet transfer: *up* from node to base station and *down* from base station to node. Depending on sender and receiver address and the packet direction, a node or base station can determine whether it is destined for the packet or not.

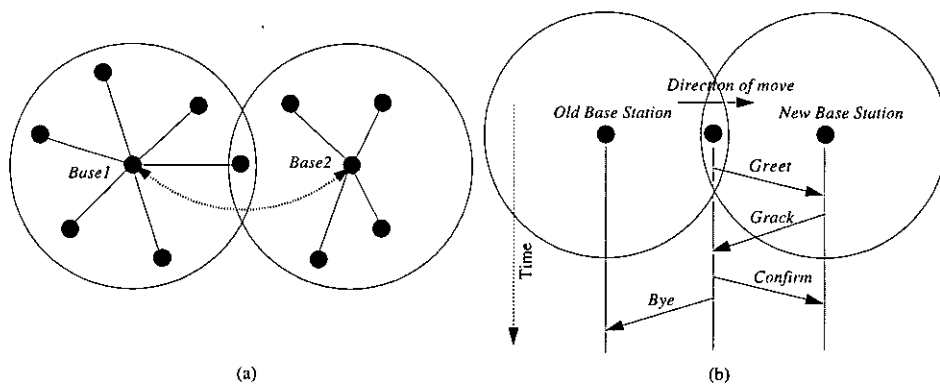


Figure 2.4: Dynamic addressing scheme in [10]. (a) Base stations (b) Handoff.

Another issues handled in [10] is handoff. When a mobile node leaves its cell and enters into another cell, handoff occurs. The node is then assigned a new address by its new base station and the earlier address is disposed and sent back to old base station for further reuse. During handoff, address adjustment is done by Greet, Grack (Greet

Acknowledge) and Confirm messages under the new base station and a Bye message is sent to old base station as shown in Figure 2.4(b).

RETRI, described by Elson *et. al* in [26], is a local addressing scheme for sensor networks. Since data centric routing hardly uses node address to identify nodes, nodes should not have any addresses and data packet should not carry any address field. However, packets may contain identifiers to denote a specific fragment of a datagram. Unlike dynamic addressing, RETRI does not assign spatially and temporally reusable addresses to nodes, rather it assigns Random, Ephemeral TRansaction Identifiers as packet identifier whenever a new data transmission is initiated. The identifiers are “probabilistically” unique along the neighbors, and can be spatially or temporally reused. Whenever a node sends a data packet, it chooses a random bit sequence as the identifier of that packet. When this packet is fragmented into smaller pieces, each fragment takes the same identifier. This technique is termed as address-free fragmentation (AFF). AFF uses fewer bits in address field in the packet than globally unique addresses.

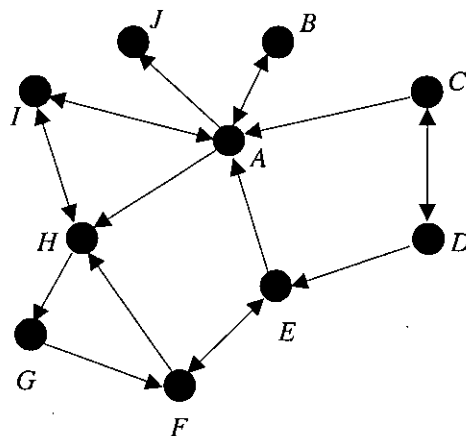


Figure 2.5: Distributed Assignment of MAC address scheme in [77]. For node A , B and I are bi-neighbors and C and E are in-neighbors. It should be ensured that B and I different address and C , E do not collide with them. But C and E may have same address.

Distributed assignment of MAC address is another local addressing technique, appeared in [77]. In this approach neighbors of a node are classified into three types: *in-neighbors* - neighbors from which the node has only incoming traffic, *out-neighbors* - nodes to which it can only send packets and *bi-neighbors*, to which node can communicate in both directions (Figure 2.5). A node can send data to and receive data from its bi-neighbors, whereas can only receive data from its in-neighbors and only send data to out-neighbors. Since nothing can be received from out-neighbors, their existence can hardly be perceived. So nodes only consider their in and bi-neighbors. Addressing scheme conforms the following principle - all bi-neighbors have distinct addresses and in-neighbors addresses differ from bi-neighbors, but two in-neighbor nodes might have same addresses. Addressing technique is quite simple. Initially all nodes do not have any address. Node selects a random address for itself and broadcast it (*broadcast_pck*). Each node keeps track of addresses of its neighbors. A node periodically broadcasts its address to its neighbors by *broadcast_pck*. If a node receiving a *broadcast_pck* detects an address conflict with one of its bi-neighbors, it sends *conflict_pck* to its bi-neighbor ordering it to re-allocate its address. If address conflict occurs between in-neighbors, no action is necessary. After receiving a *conflict_pck*, node reassigns its address and sends *broadcast_pck* informing that it have taken a new address and old one is freed up.

TreeCast [67] is a global addressing scheme for sensor networks. In this scheme nodes are organized in a tree structure and are assigned addresses according to their position in the tree. Parent nodes assign address to its descendant children and every node except the root bears an identity of its parent in its address. This scheme differs from other approaches in a way that here addresses are network addresses and they carry routing information with them. Routing based on assigned addresses is regarded as stateless routing since nodes do not maintain any state or routing history, and PalChaudhuri *et. al.* proposes first ever scheme of this type. The addressing scheme proposed in this thesis has some resemblance to *TreeCast*. Hence we describe *TreeCast* in detail in section 2.5.

Energy-efficient node address naming designed by [3], is a cluster based locally unique address assignment scheme for sensor networks. Nodes are grouped into some disjoint clusters and nodes in the same cluster are given local addresses that are unique within the cluster. Address reuse is possible among different clusters. Each cluster is assigned a unique address, and clusters are arranged logically in some hierarchy. When nodes communicate within a cluster, they use their local addresses, but when they communicate amongst clusters, cluster ID is appended with the local addresses. If all cluster IDs along with the local address are combined together for a node, a network wide unique address of that node is generated.

Distributed Global Identification is another global address assignment technique, proposed by Ould-Ahmed-Vall *et. al.* [66]. In this approach nodes are assumed to have no hardware ID or any other unique identification mark when they are deployed. The approach works in three successive phases: Phase 1 - tree building and temporary ID assignment, Phase 2 - collecting the sub-tree sizes and Phase 3 - final unique ID assignment. The scheme assigns unique ID to every node but the assigned addresses do not bear any routing information.

2.3.2 Routing Protocols

Routing is always a very important issue for any network. Routing finds route for a data packet to reach the destination that is far away from its source. Due to extreme resource constraints, traditional routing table based routing algorithm does not suitable for sensor networks. Numerous routing protocols are found in the literature. Routing protocols in sensor networks can be classified into two major classes. One is based on network structure and another is on protocol operations. Network structure based routing can be flat, hierarchical and location based, whereas protocol operations vary from negotiation, query, QoS and multi-path. In almost all routing protocols, energy limitation of sensor nodes is highly addressed and these protocols always try to preserve energy in all possible

ways. An excellent description of almost all known routing protocols for sensor networks appears in [4]. We give a short overview of some most referred routing protocols below.

Flooding and Gossiping [34] are two classical mechanisms to relay data in sensor network without the need for any routing algorithms and topology maintenance. In flooding, each sensor receiving a data packet broadcasts it to all of its neighbors and this process continues until the packet arrives at the destination or the maximum number of hops for the packet is reached. On the other hand, gossiping is a slightly enhanced version of flooding where the receiving node sends the packet to a randomly selected neighbor, which picks another random neighbor to forward the packet to and so on.

SPIN (Sensor Protocols for Information via Negotiation), proposed by Heinzelman *et. al.* in [36, 37], is a flat routing protocol. Here the sensor first broadcasts an advertisement ADV describing the event in brief rather than sending the complete data. The interested node sends REQ (request) messages to the source and then collect full data from the notifying sensor. SPIN's meta-data negotiation solves the problems of flooding, and thus achieving a lot of energy efficiency. SPIN is a 3-stage protocol as sensors use three types of messages ADV - to advertise new data, REQ - to request data and DATA - to communicate as shown in Figure 2.6. SPIN family includes many protocols, such as SPIN-1 (as described), SPIN-2 (incorporates threshold-based resource awareness), SPIN-BC (for broadcast channel), SPIN-PP (for point-to-point communication) , SPIN-EC (adds energy heuristics), SPIN-RL (for lossy channel). Further detail is referred to [36] and [52].

Directed diffusion. In [45], C. Intanagonwiwat *et. al.* proposes a data-centric routing protocol for sensor networks. In this case, node (usually sink) expresses its interest for data with specific attributes and diffuses the interest into the network. Eligible sensors that can observe the requested event, send their data on a return path to the node (Figure 2.7). Return paths are determined along the reverse direction of the interest paths. This

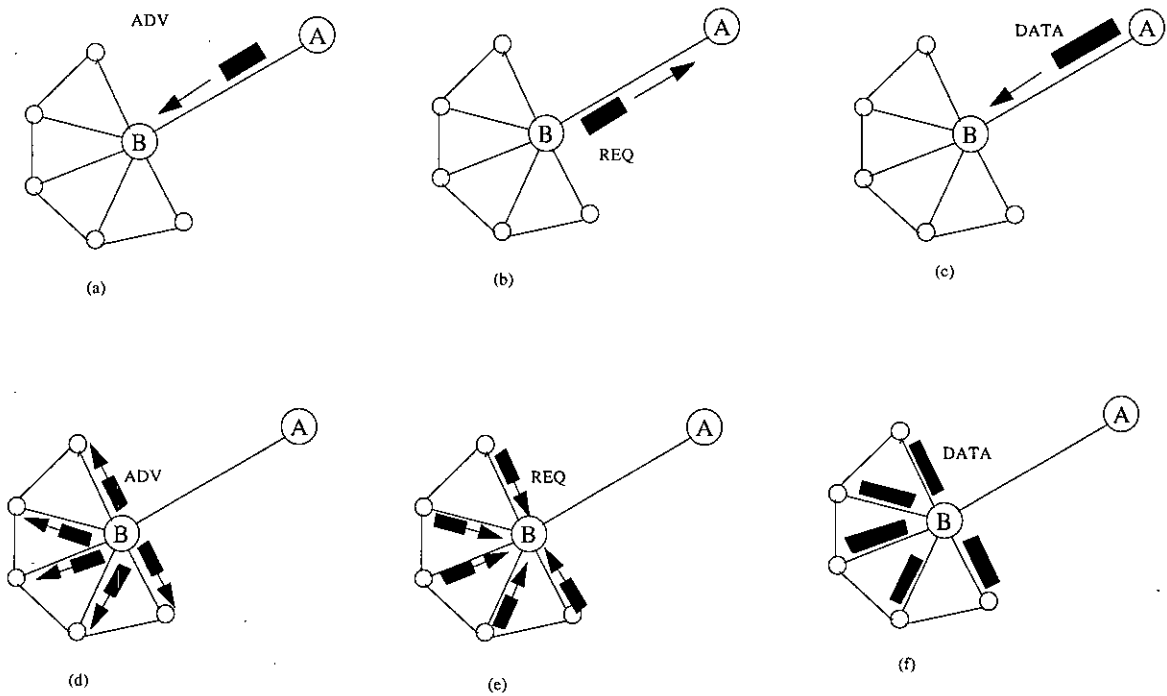


Figure 2.6: SPIN Protocol. (a) Node A starts by advertising its data to node B, (b) Node B responds by sending a request to node A, (c) Node B receives the requested data, (d) Node B sends out advertisements to its neighbors (e-f) Neighbors in turn send requests back to B.

process is called “gradient setup” in directed diffusion. Special arrangements are made to maintain and optimize the routes. The salient feature of directed diffusion is that it combines data coming from different sources by eliminating redundancy, minimizing the number of transmissions; thus saving network energy and prolonging its lifetime. Directed diffusion finds routes from multiple sources to a single destination that allows in-network consolidation of redundant data.

Rumor routing protocol [13] uses a set of long-lived *agents* to create paths that are directed toward the events they encounter. Whenever an *agent* crosses path with a path leading to an event that it has not encountered yet, it creates a path state that leads to the event. When the *agents* come across shorter paths or more efficient paths, they optimize the paths in routing tables accordingly. Each node maintains a list of its neighbors and

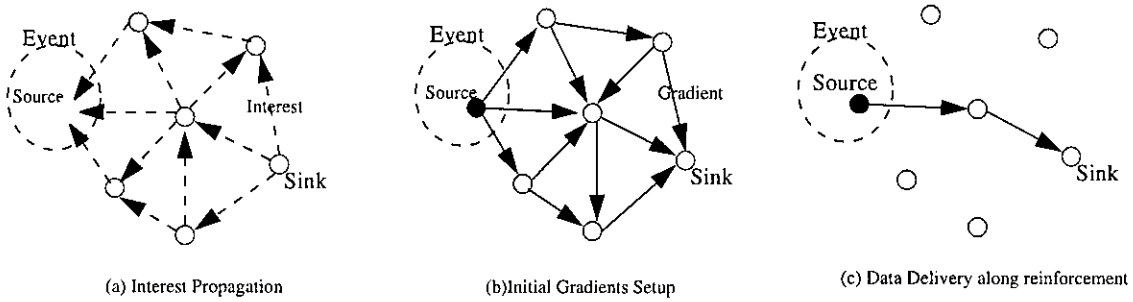


Figure 2.7: Directed-Diffusion protocol phases.

an event table that is updated whenever new events are encountered. Each node can also generate an *agent* in a probabilistic fashion. The *agent* has a lifetime of a certain number of hops after which it dies. A node does not generate a query unless it learns a route to the required event. If there is no route available, the node transmits a query in a random direction. Then, the node waits to know if the query reached the destination for a certain amount of time, after which the node floods the network if no response is heard from the destination.

Gradient-based routing GBR [78] is a variant of directed diffusion. The key idea in GBR is to memorize the number of hops when the interest is diffused through the whole network. As such, each node can calculate a parameter called the height of the node, which is the minimum number of hops to reach the base station. The difference between a node's height and that of its neighbor is considered the gradient on that link. A packet is forwarded on a link with the largest gradient. GBR uses some auxiliary techniques such as data aggregation and traffic spreading in order to uniformly divide the traffic load over the network. GLIDER, gradient landmark based distributed routing [29], is another gradient based routing protocol where a preprocessing phase discovers the global topology of the sensor field and, as a by product, partitions the nodes into some routable tiles - regions where the node placement is sufficiently dense and regular that enables local greedy methods to work well. A set of local coordinators are derived from the connectivity graph and certain landmark nodes are associated with a node's own and

neighboring tiles. Later, the protocol uses tile adjacency graph for global route planning and the local coordinators for realizing actual inter and intra-tile routes. Recently, another new gradient based routing protocol for sensor networks is proposed by Xia *et. al.* in [95].

LEACH, Low Energy Adaptive Clustering Hierarchy [37, 38], is a clustering based routing protocol for sensor networks. LEACH randomly selects some sensors as *clusterheads* (CHs) and rotates this role among sensors to distribute energy load evenly in the network. The operation of LEACH is broken into some *rounds*, each *round* consisting of three phases: advertisement, cluster set-up and schedule. In LEACH, the *clusterhead* (CH) nodes compress data arriving from nodes that belong to the respective cluster, and send an aggregated packet to the base station in order to reduce the data volume to be transmitted to the base station. Sensors inside a cluster communicate by TDMA scheme whereas *clusterheads* use CDMA technique. PEGASIS [57] is an enhancement of LEACH which applies better energy management among sensors to extend the lifetime of the network. Very recently Chang *et. al* introduces MECH (minimum energy cluster-head) [18], based on LEACH, that locates *clusterheads* more evenly than LEACH with uniform cluster size at each *round*.

Location based routing. In this kind of routing, sensor nodes are addressed by means of their locations. The distance between neighboring nodes can be estimated on the basis of incoming signal strengths. Relative coordinates of neighboring nodes can be obtained by exchanging such information between neighbors. Alternatively, the location of nodes may be available by GPS (Global Positioning System), if nodes are equipped with a small low power GPS receiver [96]. Some of the well known location based routing protocols are GAF (Geographic Adaptive Fidelity) [96], GOAFR (Greedy Other Adaptive Face Routing) [51], SPAN [20], GEDIR (Geographic Distance Routing) [87].

Stateless routing is a routing technique where nodes do not require to store any routing table, list of pre-destination states or previous routing history. Stateless routing

requires minimal memory requirement. Stateless routing for sensor networks does not occur frequently in the literature. SPEED, proposed by T. He *et. al.* [33], is a stateless protocol for real-time communication in sensor networks. It only maintains immediate neighbor information for data routing. SPEED constructs a set of neighbors NS for each node and a forwarding candidate set FS out of NS i.e., $FS \subseteq NS$. The nodes in FS lie closer to the destination than other nodes in NS . A packet is to send to a destination, it is forwarded to the best FS node. Greedy Perimeter Stateless Routing (GPSR) [47] is a routing protocol for wireless datagram networks that uses the positions of routers and the destination to make packet forwarding decisions. GPSR makes greedy forwarding decisions using only information about routes to immediate neighbors in the network topology. When a packet reaches a region where greedy forwarding is impossible, the algorithm recovers by routing around the perimeter of the region. By keeping state only about the local topology, GPSR scales better in per-router state than other routing protocols. Another stateless routing architecture in *TreeCast* [67]. *TreeCast* is described in section 2.5. PSGR, protocol for stateless geographic routing, is another location-aware stateless routing paradigm for sensor networks [97]. PSGR forms dynamic forwarding zone on the basis of density estimated on the fly.

2.4 Graph Related Terminologies

In this section, we provide definitions of some graph related terminologies that are frequently encountered in this dissertation. Graph is one of the very powerful graphical tool that is excessively used in presenting information of various types.

2.4.1 Graph

A graph presents relationship between objects. Graphically objects are represented as vertices (filled small circles) and relationship between any two objects is shown by an edge

(a line segment joining the corresponding vertices). Mathematically a graph $G(V, E)$ is defined to consist of a set of vertices V and a set of edges E such that if two vertices u and v ($u \in V$ and $v \in V$) have relationship between them, there would be an edge joining them in G and the edge (u, v) would belong to E , i.e., $(u, v) \in E$. Objects or vertices can represent anything in the physical world (people, material, computer, book, etc) and relation or edge can represent any perceptual relation between two objects. Suppose, a graph can be formed by a set of computers where computers are represented as vertices and edges may represent any relation between them such as connectivity. So two computers would have an edge in the graph, if they can communicate each other by a wired connection. Figure 2.8 shows a typical graph $G(V, E)$ with $V = \{a, b, c, d, e, f, g\}$ and $E = \{(a, b), (a, e), (b, c), (b, e), (c, d), (d, e), (d, g), (e, g), (e, f)\}$.

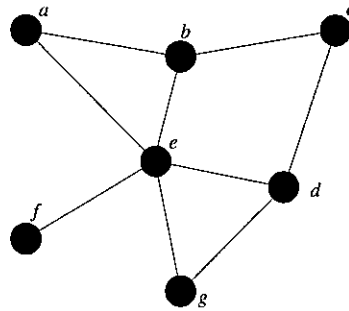


Figure 2.8: A Graph.

2.4.2 Connected Graph

Two vertices u and v is said to have a path between them, if there is a sequence of consecutive edges between them in the graph. For example, in Figure 2.8, a and d have a path in the form $a \rightarrow e \rightarrow d$ or $a \rightarrow b \rightarrow e \rightarrow d$ or $a \rightarrow b \rightarrow c \rightarrow d$. A graph is connected if any two vertices of it have a path between them. For example, Figure 2.8(a) is a connected graph. A connected graph indicates that from any vertex in the graph, we can reach to any other vertex along the edges. If a graph is not connected, it is called a disconnected graph. Figure 2.9 is an example of disconnected graph.

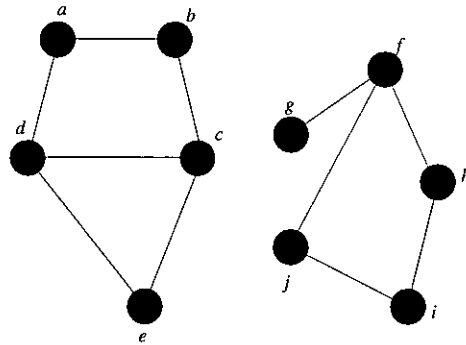


Figure 2.9: A disconnected graph

2.4.3 Unit Disk Graph

A unit disk graph consists of a set of vertices that are located in 2-D space and two vertices have an edge if and only if their euclidean distance is less than or equal to unit length. In a unit disk graph, a vertex v has edges with vertices that lie within the unit radius circle centered at v . Figure 2.10 shows a unit disk graph.

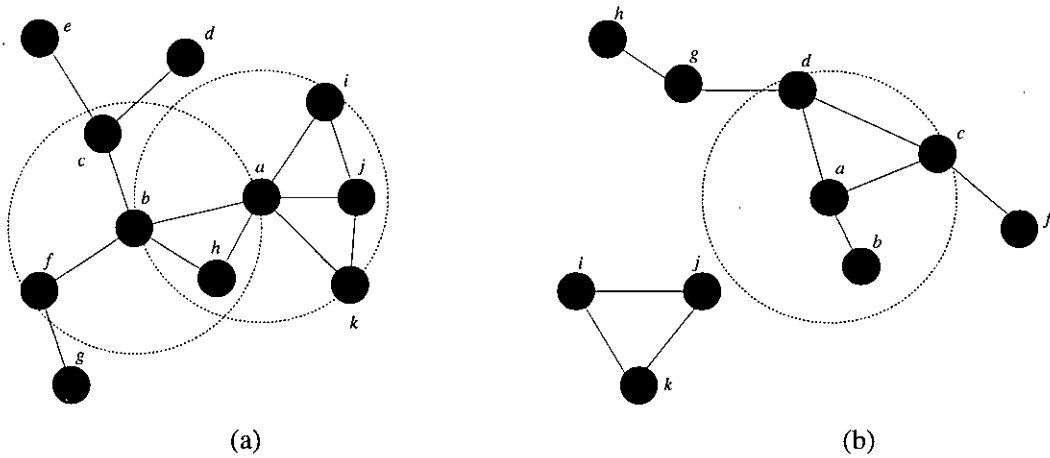


Figure 2.10: Unit disk graph (a) Connected (b) Disconnected (Circles are assumed to be of radius 1)

Unit disk graph frequently encounters in wireless communication system. Here hosts are represented as vertices and a host has edges with other hosts who can receive signal directly from the the host. In this case, unit length radius is the transmission range of

the host radius.

2.4.4 Tree

A cycle is defined to be a sequence of consecutive edges that starts and ends at the same vertex. For example, in the graph at Figure 2.8, $b \rightarrow c \rightarrow d \rightarrow e \rightarrow b$ is a cycle. A tree is a connected graph that has no cycle (Figure 2.11). A tree that contains every vertex of a graph is called a spanning tree (Figure 2.11(b)).

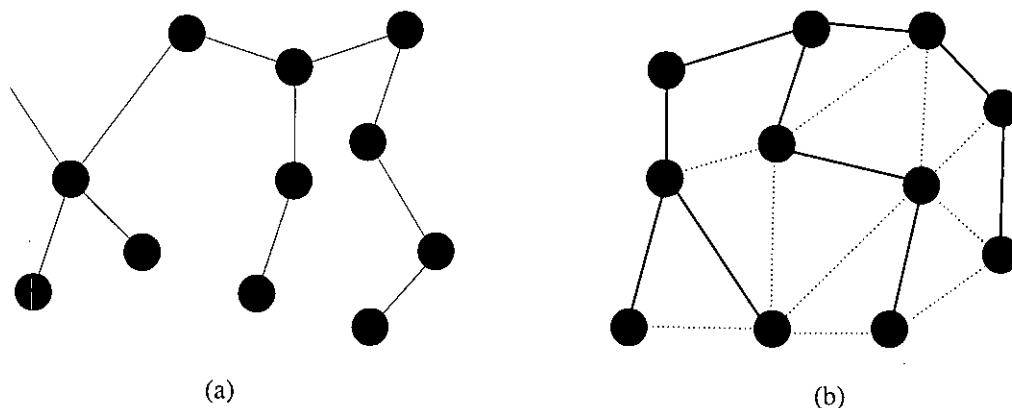


Figure 2.11: (a) Tree (b) Spanning tree of a graph. (Dark edges belong to spanning tree).

2.5 TreeCast

TreeCast [67] is a *stateless* addressing and routing architecture for sensor networks. The *TreeCast* architecture provides intelligent and automatic address assignment to the sensor nodes in the network and uses these addresses for *stateless* routing. The architecture has two main parts: address allocation and routing.

2.5.1 Address Allocation

In *TreeCast* the sensor nodes are organized into one or more trees where a sink node resides as the root of each tree. If a network has a single sink, a single tree is built rooted

by the sink node. For multiple sinks there would be multiple trees. A node with depth k (k nodes away from the sink) in the tree, gets an address with k levels of the form $(0|1)^{bk}$, where b is the number of bits assigned for each level of address. This b is determined by sink before assigning address to nodes depending on the node density of the network. For any node if the number of neighbors of that node is N , b is chosen such that $2^b \gg N$. A tree like structure of the nodes rooted at the sink (multiple tree for multiple sink nodes) is constructed, and each node is assigned an address in hierarchical fashion. A child node gets address from its parent, and address is formed by appending node's own address (selected randomly and locally unique) after its parent's address. Sink node, the top most parent, takes its own address as 1. Address hierarchy is shown in Figure 2.12.

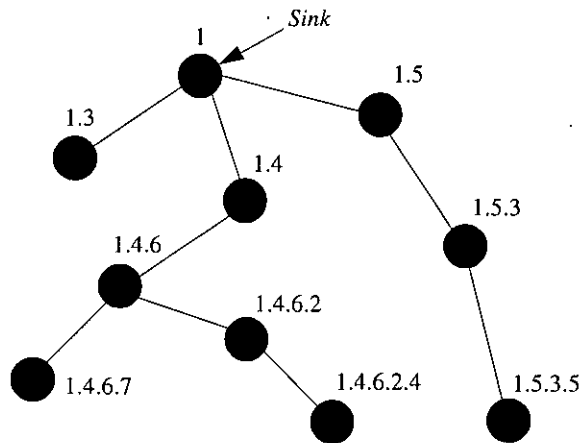


Figure 2.12: Address Allocation for nodes in *TreeCast*.

The address assignment protocol has three parts as shown in Figure 2.13: *CHOOSE PARENT*, *DO ALLOCATION* and *HEAR ALLOCATION*.

CHOOSE PARENT part of the address assignment protocol is done by each sensor when it does not have any parent associated with it. A *CONFIRM* packet signifies that a node has assigned a unique address. In this part, nodes do the following:

- 1) In level $(m+1)$, each node waits for a fixed multiple, say k , of T_{wait} time after receiving first *CONFIRM* packet from nodes at level m . This value of k is determined by examining the ratio of 2^b and N .

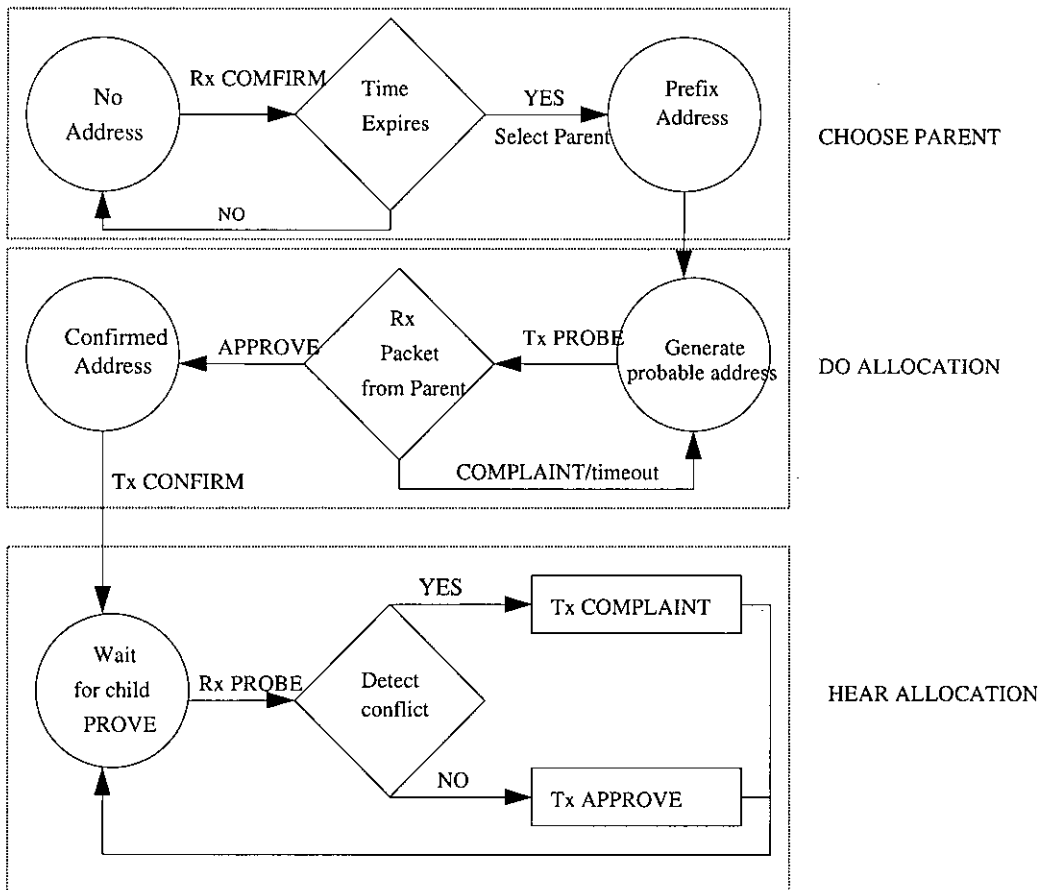


Figure 2.13: Address assignment algorithm of *TreeCast*.

- 2) Among the nodes at level m from which the nodes gets *CONFIRM* packet, the current node selects one of them (possibly someone felt with strong signal) randomly as its parent node. When a node selects its parent, parent's address becomes its prefix address.

In **DO ALLOCATION** part nodes which have selected their parent, choose a locally unique address for them and appends this address with their prefix address. The following actions are performed in **DO ALLOCATION** part:

- 1) The node generates a b -bit random number and concatenate it at the end of the prefix address it gets from its parent.
- 2) The node then broadcasts its address to its neighbors with a *PROBE* packet, after

a random delay in order to avoid collisions. The *PROBE* packet contains a random tag string to identify the packet.

- 3) The node waits for T_{wait} time to listen *APPROVE* or *COMPLAINT* packet from its parent. If no such packet comes within a specified time, step 2 is repeated.
- 4) If the node gets a *COMPLAINT* packet in response to its *PROBE*, it means that its chosen address collides with other ones. Then the node selects another b -bit random number, sends it to the parent via *PROBE* and waits for *APPROVE* or *COMPLAINT* from the parent.
- 5) If the node gets *APPROVE*, its address is approved and it keeps its address and sends *CONFIRM* packet to the parent containing its address.

HEAR ALLOCATION part of the address assignment is done by a node after it has confirmed its address. In this part, a node does the following tasks:

- 1) A node waits for *PROBE* packet from nodes whose probed addresses are direct children of itself.
- 2) If it receives a *PROBE*, it checks the address inside to see whether it heard and approved this address before. If the node detects this address has already been approved for any other node, it sends *COMPLAINT* to the probing node. If the probed address is heard new, an *APPROVE* packet is sent to the node. The addresses that a node approves are stored temporarily and deleted when *HEAR ALLOCATION* step is done.
- 3) If the child sends a *CONFIRM* packet in response to the *APPROVE* packet, the parent records the child's address in memory ensuring not to approve any other child with the same address.

Thus a tree (or multiple trees for more than one sink) is constructed rooted at sink node and each node is assigned a unique address depending on their level in the tree. The address assignment has two properties:

- 1) A node can identify its level by simply knowing its address and the constant b . If the node is at level k , its address contains $(k + 1)b$ bits.
- 2) A node, with the address of the form $(0|1)^{bk}$, has a parent bearing the address of the form $(0|1)^{b(k-1)}$, and the parent's address is the strict prefix of the node address.

2.5.2 Routing

Once address assignment is done for all sensor nodes, data can be routed among the sensors. Usually in a sensor network three types of routing messages are considered, Query messages, Response messages and Control messages.

Query Messages are sent to all sensor nodes by the sink node when it wants to know about certain type of event from the associated sensor nodes. Query message is broadcast into the entire network expressing the sink's interest for certain event. Associated sensors who trace the inquired events, send response message back to the sink. In *TreeCast* when a query is to be made, sink node broadcasts a query message to its neighbors. Each neighbor receives the message and forwards the message along its subtree. When the query message is forward to the children, the message contains the address of the forwarder. By observing the forwarder address in the query message, child nodes can detect whether the query message is forwarded from its parent or not. If a node at level $k + 1$ receives a forwarded query message from a node at level k and finds a complete match between its and the forwarder's address in all address levels except the last, the receiver is sure of getting the query message from its parent. In that case the child node forwards it further onto the subtree, otherwise it ignores the message. This strategy reduces the redundant transmission of same query message in the network.

Response Messages are sent by the sensor nodes in response to a query from the sink. In that case, the node sends the response message to its neighbors. If the sender of the

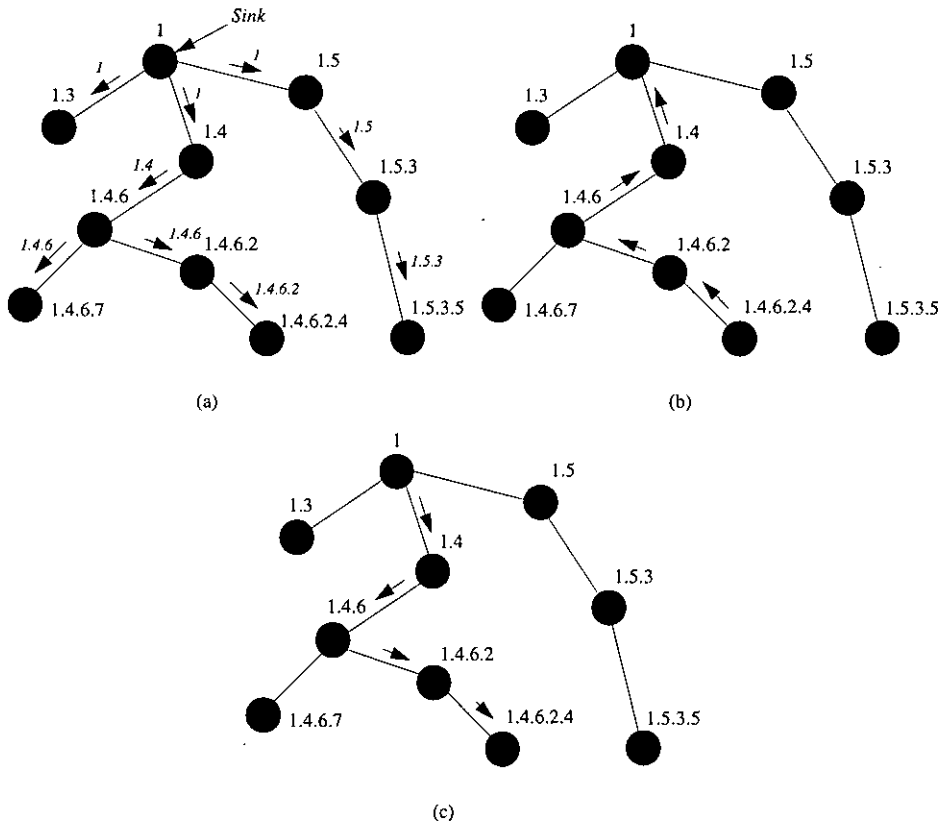


Figure 2.14: Routing in *TreeCast* (Arrow shows the direction of the message propagation).
 (a) Query message: message reaches every node, message bears the forwarder's address (b)
 Response message: Node 1.4.6.2.4 sends a response to the sink, message forwarded along the
 reverse prefix match path (c) Control message: sink sends to 1.4.6.2.4, message forwards along
 the prefix match path

response message is at k level, the message is received by a node at $k - 1$ level whose address is the perfect prefix of the sender. As the addressing suggests such a node always exists. This node accepts the message and broadcasts it again by placing its level in the message. Thus message traverse upward and finally reaches the sink.

Control Messages are issued by the sink node to a specific sensor node or a group of sensors for collaboration or in-network control and actuation. This sort of communication is required to manage certain tasks among the sensors. This is accomplished in the tree by

forwarding the message from the sink to the nodes till the perfect prefix match is found. A node at level k receives a control message from a node at $k + 1$ if the later becomes the destination or its address is the complete prefix of the destination. In that case the upper level node accepts the message and broadcasts it to its neighbors for further forwarding along the tree. In this way the message goes down the tree till the target sensor or group of sensors is reached.

2.5.3 Node Failure

Since *TreeCast* organizes nodes in a logical tree, failure of any node disconnects the tree. But the actual underlying neighborhood connectivity and reachability to the sink may not be damaged. If a parent node fails, child node can no longer sends their response messages to the sink via its parent. In *TreeCast*, whenever a child node sends a message to the parent to forward it to the sink, it overhears the transmission of the parent, listening whether the parent is transmitting its message further ahead or not. If child listens no response from its parent in this regard, it detects a parent-dead situation and the child becomes orphan.

When a node at level k becomes orphan, it can do any of the two following actions to forward its response to the sink:

- The node can make another neighbor node at level $k - 1$ to forward the response to the sink. This can be made by setting ANY PARENT flag in the packet to *true*. When the node wants only its parent to forward the packet, this flag is set to *false*.
- When no node at level $k - 1$ other than the parent take the responsibility to forward the packet of the orphan node, the node tries this time with LATERAL flag set. This flag enables any neighbor at the same level of the orphan node to participate in forwarding of the response message.

2.6 Sensor Network Simulator

In this section, we briefly describe some important simulators that are used in the simulations of sensor networks.

2.6.1 TOSSIM

TOSSIM [55], developed at UC Berkley, is a discrete event simulator for TinyOS sensor network. TinyOS is an operating system specially built for sensor nodes. Berkley sensor nodes run TinyOS and TinyOS based applications in them. Using TOSSIM, a developer can test the applications written on TinyOS for sensor network without going into the real deployment. TOSSIM scales to thousands of nodes, and compiles directly from TinyOS code; developers can test not only their algorithms, but also their implementations. TOSSIM simulates the TinyOS network stack at the bit level, allowing experimentation with low-level protocols in addition to top-level application systems. Users can connect to TOSSIM and interact with it using the same tools as one would for a real-world networking, making the transition between the two easy.

TOSSIM has a scripting language, which allows users to interact with running and paused simulations. This allows users to reproduce complex scenarios (of motion, for example) and construct tests of algorithmic corner cases. Additionally, having a full scripting language allows users to build scripting primitives, slowly creating a library of functions and test cases for TinyOS efforts.

TOSSIM also has a GUI tool, TinyViz, which can visualize and interact with running simulations. Using a simple plugin model, users can develop new visualizations and interfaces for TinyViz.

2.6.2 SensorSim

SensorSim [69] is a simulation framework that support many models and techniques for the design and analysis of sensor networks. SensorSim inherits the basic feature of traditional event driven network simulators, and builds up additional features that include ability to model power usage for sensor nodes, hybrid simulation that interacts between the real and simulated nodes, communication protocols and real time user interaction with graphical data display. SensorSim incorporates enhanced power model that includes battery model, CPU model, radio model, sensor device model and a middleware platform SensorWare to facilitate better simulation on sensor networks.

2.6.3 NS-2

NS-2 [64] is a discrete event network simulator that is used widely in network simulations. Initially intended for wired networks, the Monarch Group at CMU have extended NS-2 to support wireless networking. NS-2 supports numerous physical radio channel models, propagation models and wireless media protocols. NS-2's code source is split between C++ for its core engine and OTcl, an object oriented version of TCL for configuration and simulation scripts. The combination of the two languages offers an interesting compromise between performance and ease of use. Implementation and simulation under NS-2 consists of 4 steps:

- Step-1. Implementing the protocol by adding a combination of C++ and OTcl code to NS-2's source base;
- Step-2. Describing the simulation in an OTcl script;
- Step-3. Running the simulation;
- Step-4. Analyzing the generated trace files.

Implementing a new protocol in NS-2 typically requires adding C++ code for the protocol's functionality, as well as updating key NS-2 OTcl configuration files to recognize the new protocol and its default parameters. The C++ code also describes which parameters and methods are to be made available for OTcl scripting. The NS-2 architecture follows the OSI model closely. An agent in NS-2 terminology represents an endpoint where network packets are constructed, processed or consumed.

Some disadvantages of NS-2 stem from its open source nature. First, documentation is often limited and out of date with the current release of the simulator. Fortunately most problems may be solved by consulting the highly dynamic newsgroups and browsing the source code. Then code consistency is lacking at times in the code base and across releases. Finally, there is a lack of tools to describe simulation scenarios and analyze or visualize simulation trace files. These tools are often written with scripting languages. The lack of generalized analysis tools may lead to different people measuring different values for the same metric names. The learning curve for NS-2 is steep and debugging is difficult due to the dual C++/OTcl nature of the simulator. A more troublesome limitation of NS-2 is its large memory footprint and its lack of scalability as soon as simulations of a few hundred to a few thousand of nodes are undertaken.

2.6.4 PARSEC

PARSEC [6] (for PARallel Simulation Environment for Complex systems) is a C-based discrete-event simulation language and a package, as well. It adopts the process interaction approach to discrete-event simulation. An object (also referred to as a physical process) or set of objects in the physical system is represented by a logical process. Interactions among physical processes (events) are modeled by timestamped message exchanges among the corresponding logical processes. One of the important distinguishing features of PARSEC is its ability to execute a discrete-event simulation model using several different asynchronous parallel simulation protocols on a variety of parallel architectures.

PARSEC is designed to cleanly separate the description of a simulation model from the underlying simulation protocol, sequential or parallel, used to execute it. Thus, with few modifications, a PARSEC program may be executed using the traditional sequential (Global Event List) simulation protocol or one of many parallel optimistic or conservative protocols. In addition, PARSEC provides powerful message receiving constructs that result in shorter and more natural simulation programs.

2.6.5 GloMoSim

GloMoSim [7] is a scalable simulation environment for wireless and wired networks systems developed initially at UCLA Computing Laboratory. It is designed using the parallel discrete-event simulation capability provided by PARSEC. GloMoSim currently supports protocols for purely wireless networks. It is build using a layered approach. Standard APIs are used between the different layers. This allows the rapid integration of models developed at different layers by users. To specify the network characteristics, the user has to define specific scenarios in text configuration files: `app.conf` and `Config.in`. The first contains the description of the traffic to generate (application type, bit rate, etc.) and the second contains the description of the remainder parameters. The statistics collected can be either textual or graphical. In addition, GloMoSim provides various applications (CBR, ftp, telnet), transport protocols (TCP, UDP), routing protocols (AODV, flooding) and mobility schemes (random waypoint, random drunken).

Chapter 3

The HN-Addressing Protocol

In this chapter we describe our proposed addressing and routing scheme for wireless sensor networks. The addressing and routing scheme proposed here is *stateless* in a sense that it does not require any state keeping memory for its function. Nodes are completely stateless. They are assigned with an address and depending on these assigned address nodes route data to the desired destination. For data routing, nodes neither do store any routing table nor do keep any previous routing history.

We have seen earlier that *TreeCast* is a *stateless* addressing and routing architecture which organizes nodes in a tree like structure and assigns address to nodes depending on their depth in the tree. HN-Addressing differs from *TreeCast* in the following ways:

- Unlike *TreeCast*, we do not increase level of address in every depth of the tree, rather we make use of all available numbers in a certain address level.
- For a certain address level, *TreeCast* assigns addresses only to immediate neighbors keeping many unused addresses in that level. But in HN-Addressing, we go deep into the tree while addressing nodes in a certain address level, rather than confining it to only direct neighbors. When no more nodes can be assigned address in the current level, we raise the address level to the next level.

- HN-Addressing uses address bits in a level more efficiently and address level increases quite slowly. This gives well reduction of address bits, and nodes are assigned with addresses of shorter length. With fewer address bits, nodes can transfer data to sink or receive data from sink with lower communication energy.
- *TreeCast* does not restore the tree structure of node in case of any node failure. But HN-Addressing restores the tree structure when a node fails and reallocates addresses to the affected nodes.

3.1 Assumptions

Before going to detail description of our approach, we make the following assumptions for a sensor network:

- (1) *Single Sink node:* Among the sensors deployed in the network, there would be a sensor node that acts as a coordinator in the network. This node is termed as sink node. Sink connects sensor network with the existing wired network for the collaboration of sensing activities. Sink has good power supply, long range transmission radio and does not drain out of energy in short time. Normally there could be more than one such sinks in a sensor network. In our model we assume there would be a single sink, although our technique can be extended for multiple sinks.
- (2) *Unique ID of nodes:* Each node in the network must have a unique ID for its identification. Possibly this ID can be the hardware ID that is inscribed with their hardware when manufactured. If nodes do not have such ID for their own, a randomly generated tag string (possibly of 48 bits or more) can be used as temporary ID for that node. For each node, this temporary ID must be different from its neighbor nodes. Nodes within the same transmission radius should not contain the same ID. For a moderate sensor density, using considerably long bit sequence can achieve this.

- (3) *Zero mobility:* In our architecture we assume that nodes remain stationary in their position and hardly move throughout their entire life.
- (4) *Connected network:* Network is assumed to be always connected and nodes are always reachable from the sink by multihop transmission. So a tree is always maintained and no network partition is ever built.
- (5) *Broadcast channel:* The wireless channel is assumed to be a broadcast channel. That is, whenever a node sends a packet to a specific neighbor node, the packet goes to all nodes lying within the transmission range of the sender. A node listens every transmission of its neighbors.
- (6) *Communication scenario:* In our approach, we assume that all communications are sink-oriented, i. e., sensor nodes receive data from the sink or they send data to the sink. Sensor nodes do not send/receive data among themselves. They act as data originator or router. Though our approach can support packet communication with any arbitrary destination apart from the sink, it might not have strong application in sensor networks.
- (7) *Node fails, not link:* We consider the failure of nodes only, not the links. That is, nodes may sometimes fail due to hardware fault, drain-out of battery power or any other reason, but the wireless link among the active nodes never fails. If two nodes are active, there is no way to place a hindrance between their communication. No messages or packets ever lost due to channel error. Although there can be some situations like blocking by object, extreme noise or fading in wireless channel that may cause a link to malfunction, we ignore these effects in our model.

3.2 Design Goals

We identify the following design goals for the proposed address and routing architecture for sensor networks:

- *Efficiency*: The address allocation scheme would be efficient in terms of number of address bits. The scheme would accommodate all nodes by shorter address length.
- *Accuracy*: The address must be accurate and should not collide globally. No two nodes in the network should ever get the same address. It should not only guarantee the collision free addressing at allocation time, but should also maintain the collision free state in the long run in case of node addition/deletion.
- *Scalability*: The addressing scheme should scale the size of the network implying that widespread periodic broadcasts are not desirable.
- *Routing enabled addressing*: The addressing of nodes should enable routing of data packets based on the assigned addresses.

3.3 Basic Idea and Methodology of HN-Addressing

Before we go the actual addressing algorithm of HN-Addressing, we explain the basic idea and methodology of HN-Addressing with illustrative examples in this section. The formal treatment of the technique is produced in the successive sections.

3.3.1 Addressing without Hierarchical Levels

In a sensor network, nodes are normally deployed in a random fashion without any previous planning for sensor locations ¹. The scattered sensors form a graph in the region where sensors are represented as vertices and two vertices have an edge if and only if they are within the transmission range of each other. This type of graph is known as unit disk graph, defined as the graph obtained from a set of vertices and edges between any two vertices if two circles of unit radius at the center of these two vertices contain each other. In our case, the unit radius is the transmission range of the sensor radio. We

¹In Smart dust project [46], people plan to spread sensors on the target area from an aeroplane.

call this graph as communication graph. On the communication graph, we conduct two operations:

- *Finding spanning a tree rooted at the sink.* We find a spanning tree from the communication graph that spans all nodes in the network. In the process of finding spanning tree, each node selects a neighbor node as its parent and the sink becomes the supreme grand parent of all nodes. Nodes that are not parent of any other node, remain as *leaf*.
- *Numbering of nodes according to the traversal of the tree from the root.* Then, a *preorder* traversal of nodes along the tree is made starting from the root, and each node is assigned a number according to its visit order, whenever being visited for the first time. In the *preorder* numbering of nodes of the tree, a node is first numbered, then it conducts numbering of each of its subtrees one after another in a certain sequence. In this process, sink is the first node to be numbered, and when the sink ends up with numbering of all of its subtrees, entire numbering process is completed and the whole network is addressed. It can be easily verified that every node gets the different number in this process. The assigned number of a node acts as the address of that node. Figure 3.1 shows the addressing steps.

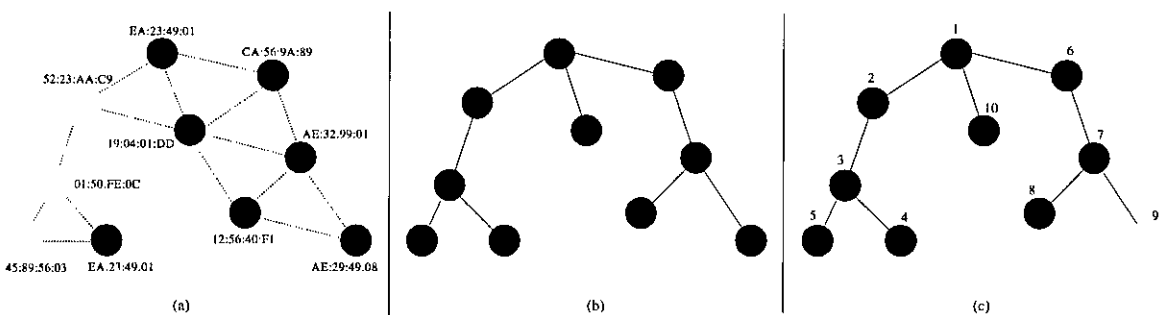


Figure 3.1: Addressing steps in HN-Addressing. (a) Underlying communication graph (Hex numbers are hardware ID) (b) Formation of spanning tree (c) *preorder* numbering of nodes.

Observation 3.3.1 *The preorder traversal scheme assigns addresses to all nodes and no node gets duplicate address.*

Proof. Since the communication graph is assumed to be connected, there exists a spanning tree and thereby all nodes are reachable from the sink. Hence in the process of *preorder* visit of the tree, every node would be visited at least once. So all nodes get the addresses. Again nodes are visited in an ordered fashion and number of ordering is always incremented by 1 after each new visit of nodes. Since no two nodes can be visited at the same time, their visit order should be different. So addresses are unique. \square

We define *routing number* as the number that is assigned to a node in the *preorder* traversal phase. And this number acts as the address for the associated node. Along this *routing number*, every node keeps another number, called *subordinate number*, that is the maximum number taken by any node in the subtree of the corresponding node. For example, in Figure 3.2, node with *routing number* 6 bears the *subordinate number* 9 that appears as the largest number in the subtree of node 6. These two numbers (i.e., 6(9)) form the routing entity for the corresponding node. *Routing number* is shown as bare number whereas *subordinate number* appears within the parenthesis.

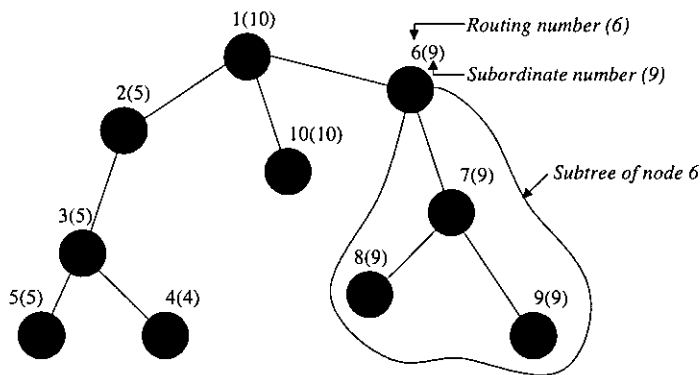


Figure 3.2: Addressing of nodes without levels with *routing* and *subordinate number*.

Data Routing

HN-Addressing supports data routing based on assigned addresses of nodes. Three types of packets are usually encountered in sensor applications: *Query* - sink node floods to all nodes, *Request* - sent by the sink to specific node and *Response* - send by a sensor node to the sink. When sink node wants to send a request packet to a specific destination sensor, it broadcasts the packet among its neighbors. Neighbor nodes receive the packet, and check whether the destination is within the subtree of the node. If the destination appears in the subtree, the node forwards the packet to amongst its neighbors. Thus the packet is reached to the destination. The situation is shown in Figure 3.3. The sink (1) sends a packet to node 8. Sink broadcasts the packet to its neighbors and node 2, 6 and 9 receive the packet. Then each node determines whether node 8 can be reachable via itself, i.e., node 8 resides in their respective subtrees. The subtree of 2 contains nodes from 3-5, whereas for node 10, it contains only itself, but nodes 7-9 belong to the subtree of node 6. So node 6 takes the packet while others drop it. Then, node 6 makes another broadcast and packet is similarly received by node 7 because of the same reason. Finally packet is received by node 8 at the next broadcast. When node 7 broadcasts the packet, it eventually reaches the node 6 again, but this time node 6 discards the packet.

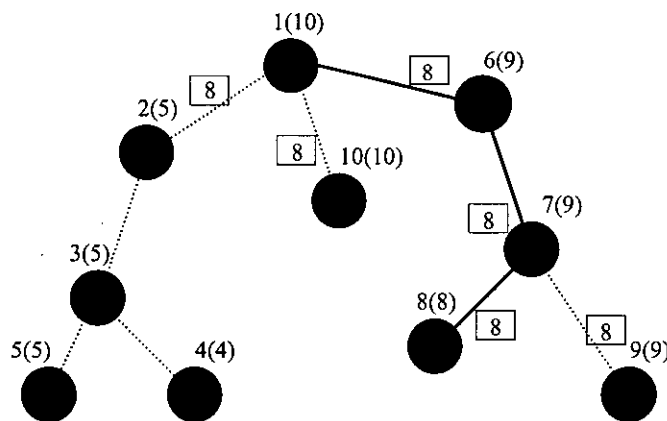


Figure 3.3: Routing packet to node 8 from the sink. (Dark lines show the path.)

Similar approach is adopted for the packet destined to the sink and query packets.

We describe routing matters more formally in section 3.8.1 of this chapter.

3.3.2 New Hierarchical Addressing Architecture

As we see, in HN-Addressing nodes get their addresses in the form of number dispensed from the root to the leaf nodes gradually. Each time a node is assigned a number the node stores this number as its *routing number* and when all nodes in its subtrees have been allocated their addresses, it gets the maximum number assigned to a node in its subtree as its *subordinate number*. As the previous description says, numbers that are assigned to the nodes are getting higher and higher as more nodes are getting their addresses. But due to the address bit constraint, this number cannot grow indefinitely.

How many address bits are required to allocate address to the entire network? If we know (or guess roughly) about the size of the network (number of nodes), we can calculate the minimum number of bits required to assign distinct addresses to all nodes to be $\lceil \log_2(N) \rceil$ for a network of size N . But in sensor applications, number of nodes may not be known earlier or it may vary due to failure of nodes now and then or addition of new nodes in the network. Keeping address length arbitrarily large leads to under utilization of address bits against the network size, and it increases the energy consumption as well for large address overhead in packet. So address allocation should adjust the address length according to the necessity of the network. This introduces the hierarchical levels in address.

The number that is assigned to nodes in *preorder* visit of the tree, cannot grow beyond some pre-specified limit. Whenever a node has to be assigned with a larger number, address level is raised. In that case address no longer remains in the form of a single whole number, say 5, rather it becomes something like 5.1 in second level or 5.1.4 in third level and so on. The largest number that can be assigned to a node in any address level is bounded by the value of bits per level (*bpl*). Figure 3.4 shows the addresses with hierarchical levels.

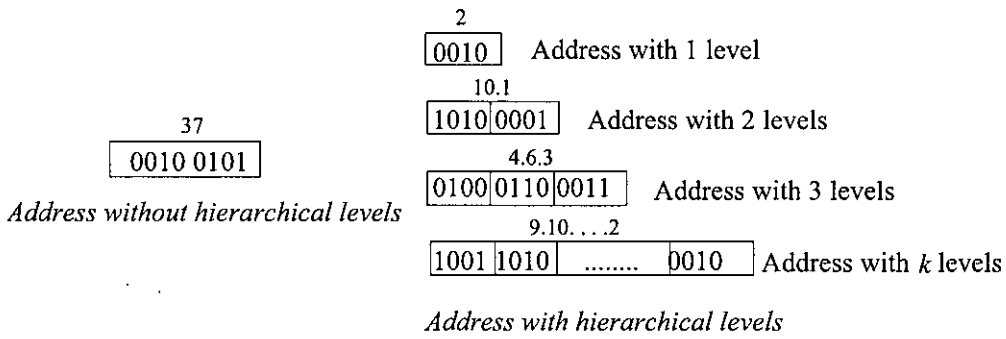


Figure 3.4: Addresses with hierarchical levels for $bpl = 4$.

Bits per level (bpl) is a critical system parameter and is decided long before the sensors deployment. Keeping bpl small rises the address levels as well as the address length, whereas large value of bpl may require less address level but each level takes more bits. The choice for bpl value depends on the number of total nodes in the network and their density. For $bpl = k$, the largest number can appear in any level is $2^k - 1$. For example, if bpl is set to 3, no node can take number larger than 7 in any address level. Whenever assigning number requires to be higher than the largest value in current address level, address level is increased by one and node takes number from the next level. We name this phenomenon as *hierarchical leveling*.

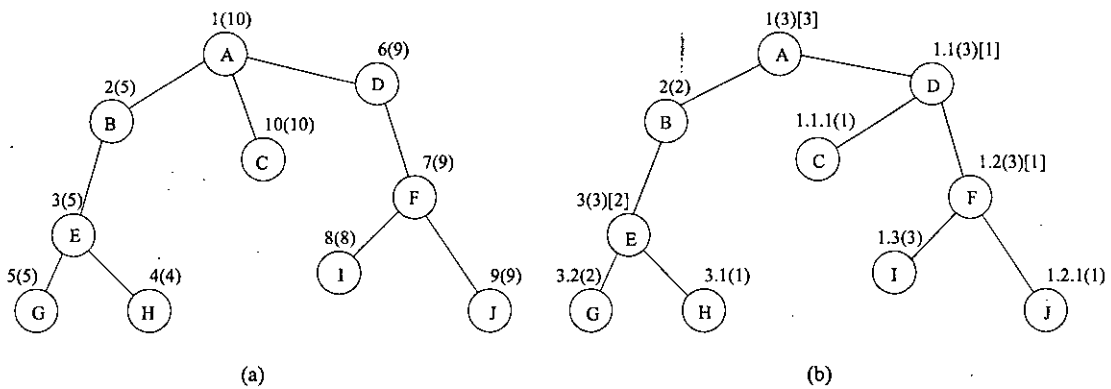


Figure 3.5: Comparison of two Addressing schemes. (a) Without level (b) With level for $bpl = 2$.

3. Node H , which gets number 4 by addressing without level, is assigned with address 3.1, an address of two levels. And similarly G is assigned 3.2 in lieu of 5. Along with storing *routing number* and *subordinate number* as before, node E now keeps another number 2 as *auxiliary number* that counts the number of nodes that are assigned addresses in the next address level of E . Similar things happen when node A goes for address allocation for node D . D takes address 1.1 and other two nodes F and G in the subtree get addresses 1.2 and 1.3 respectfully. Other two nodes C and J get addresses of three address levels. Number in the square bracket represents the *auxiliary number*. If a parent node does not assign any child node with elevated address level, the *auxiliary number* for the associated parent node is zero and this is not shown in the address notation (as for node B).

Data Routing

When hierarchical leveling of addresses are active in address allocation process and nodes get addresses with more than one address level, data routing depending on address becomes little tricky. In that case, data is forwarded level by level. Suppose for Figure 3.5 (b), the sink (A) tries to send data to G (3.2). Here the destination has two address levels, so two successive routing is combined. First data is forwarded to the node with address 3 (E in figure) which appears in the first address level of the destination. After E is reached, routing is made on next address level. Now data is forwarded to node 3.2 and eventually reaches the destination G . So the routing path is $A \rightarrow B \rightarrow E \rightarrow G$. So we can say that to reach a node, say 5.2.4.1, from the sink 1, the path to be followed is $1 \rightsquigarrow 5 \rightsquigarrow 5.2 \rightsquigarrow 5.2.4 \rightsquigarrow 5.2.4.1$.

3.4 Address Allocation Technique

In this section, we narrate in details how addresses are assigned in HN-Addressing. HN-Addressing is a distributed message passing protocol where participant nodes pass messages amongst themselves in finding the spanning tree and visiting nodes in *preorder*

manner thereafter. We describe the allocation process with hierarchical levels and without levels separately.

3.4.1 Address Allocation without Hierarchical Levels

In our address allocation approach, nodes are first organized into a tree like structure rooted at the sink node. Then nodes are assigned addresses by their respective parent node in the tree according to the *preorder* traversal of the tree. When a parent node starts allocating addresses to its children, it first picks one of its child (from a *ready* queue, as we would see subsequently) and assigns an address to it. The child node in turns assigns address to its children, and it goes below to the leaves of the tree. Until a child has not completely assigned addresses to all descendant children in the subtree, the parent node does not start address allocation for its next child. In this way a *preorder* traversal of nodes is made and nodes are addressed one after another. While conducting the visit of nodes in this fashion, nodes keep necessary information that later enable them to data routing.

The sink node initiates the addressing process and the protocol uses four control messages (actually six, we describe other two later on) for address allocation:

- **ALERT:** A node sends ALERT message to its neighbors, and sink is the first node that sends ALERT at the beginning. Upon receiving the ALERT message from any neighbor node for the first time, the recipient node selects the ALERT sender as the parent node and then sends ALERT message to its neighbors. Thus ALERT message propagates into the network and a spanning tree is readily constructed.
- **READY:** When a node selects its parent after receiving ALERT, it sends READY message back to the parent. When parent gets READY message from its child, it puts the child's ID ² in the *ready* queue, a list of child nodes that are candidate for

²Sometimes nodes do not have any hardware ID, in that case any long random tag string can be used as ID.

address allocation by the parent.

- **NUMBER**: Parent node picks one child node from the *ready* queue and sends a NUMBER message to it as a step of assigning address to the child.
- **DONE**: When a node is finished with addressing of all nodes in its subtree, it sends DONE message back to the parent directing the parent to send NUMBER message to another *ready* child node and to initiate address allocation in the corresponding subtree.

In the course of allocation process, each node remains at any of the following four states:

- **NOT-NUMBERED** Initial state for all nodes except the sink. In this state, a node has not yet selected its parent and waits for an ALERT message to come from one of its neighbors.
- **READY** The node has selected its parent and now is ready to collect address from the parent.
- **NUMBERING** A node remains in this state, if address allocation of nodes in its subtree is going on. This state is initiated by the receipt of a NUMBER message from the parent. In this state, the node picks a node from its *ready* queue and sends a NUMBER message to the child dictating the child to assign address in the child subtree and wait for DONE message back from the same child. Upon the receipt of DONE, it picks another *ready* child and the same procedure is continued. Initially the sink node remains in this state.
- **DONE** When a NUMBERING node gets its *ready* queue empty identifying that it has finished address allocation in its subtree, it sends the DONE message back to the parent. Now its state becomes DONE and it remains here subsequently.

Figure 3.6 shows the state diagram representing the transitions among states on receipt of various messages.

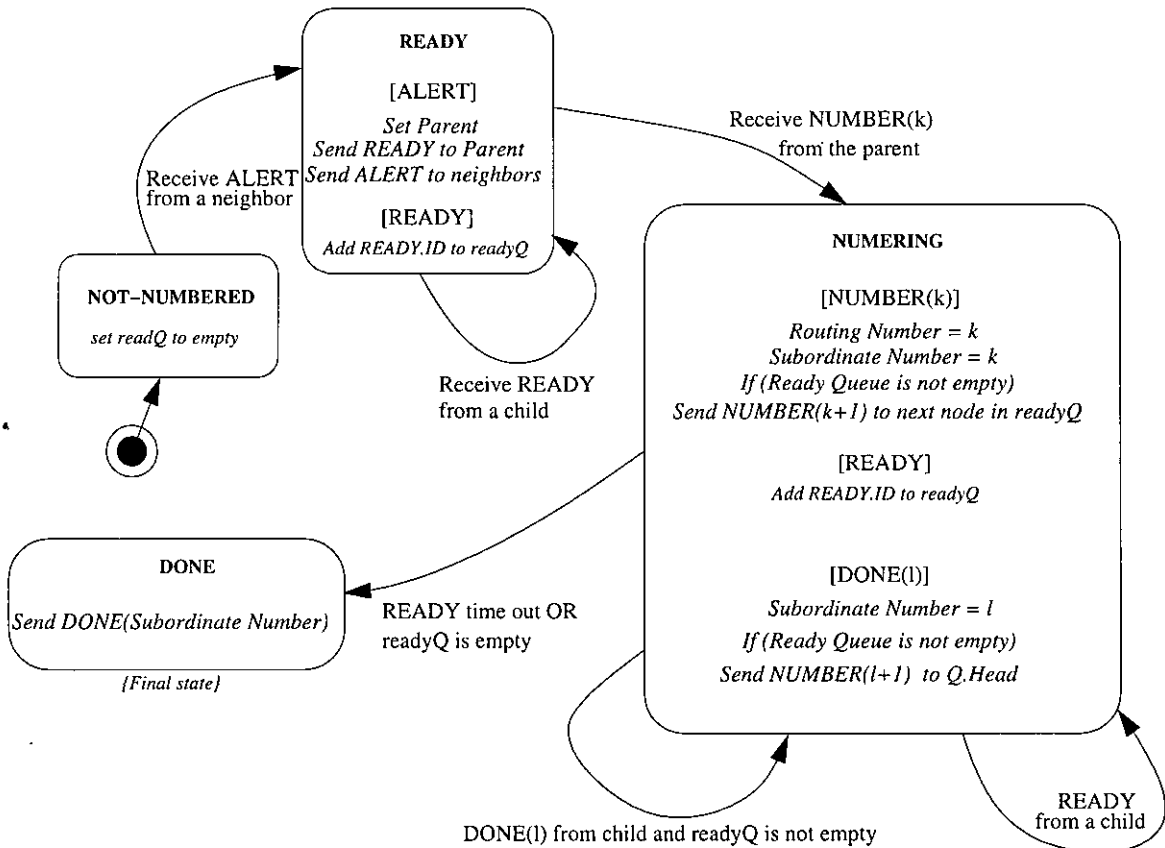


Figure 3.6: Transitions of states of node during address allocation.

Numbering of nodes

When a node in *NUMBERING* state (as the sink node at the beginning) receives first *READY* message from one of its neighbors, it starts assigning address to the child. It picks the child node's ID from its *ready* queue and sends *NUMBER(k)* message to the child. Sink node is the first node to be in *NUMBERING* state and assigns its address as 1. It then picks one of its child from the *ready* queue and sends *NUMBER* message with $k = 2$. The value of k indicates the next available address for the descendant nodes beneath the parent node in the tree. The value of k is updated each time a *DONE* message

is received from the corresponding child.

A node switches from *READY* state to *NUMBERING* state when it gets $\text{NUMBER}(k)$ message from its parent. On the receipt of $\text{NUMBER}(k)$ message from the parent, the node assigns k as its *routing number* as well as its address. Then it picks one of its *ready* child from the *ready* queue (if there is any), sends $\text{NUMBER}(k + 1)$ to the child and waits for *DONE* message from the child node. The child node after completing assigning address to descendant nodes returns *DONE* message to the parent. *DONE* message indicates that all nodes under the subtree rooted at the child node have been visited. When the parent node gets $\text{DONE}(l)$ message from the child, it updates the value of k as $k = l$ and picks the next child node from the *ready* queue and repeats the process by sending $\text{NUMBER}(k + 1)$ to the child. If there is no child left in *ready* queue, the node sends $\text{DONE}(k)$ to its parent.

The last value of k , that is returned to the parent by $\text{DONE}(k)$, is stored as *subordinate number* of the node. These two numbers, *routing* and *subordinate number* of a node form the routing entity for that node. A node with *routing number* = r and *subordinate number* = s contains those nodes in its subtree whose *routing number* lies within r and s , i.e., $r < \text{routing number} \leq s$. When sink node gets *DONE* message from all of its children, address allocation for all nodes in the network is completed. The sequence of message passing and corresponding state update of nodes is presented in Figure 3.7 for a small network.

3.4.2 Address Allocation with Hierarchical Levels

To support the hierarchical address levels, there would be some changes in the technique of address allocation presented earlier. When bits per level is incorporated, before sending the $\text{NUMBER}(k + 1)$ message to a *ready* child the parent node should check whether the next address assignment exceeds the largest number for the current address level. If $k < 2^{bpl} - 1$, $\text{NUMBER}(k + 1)$ can be sent to the child node, otherwise the parent has

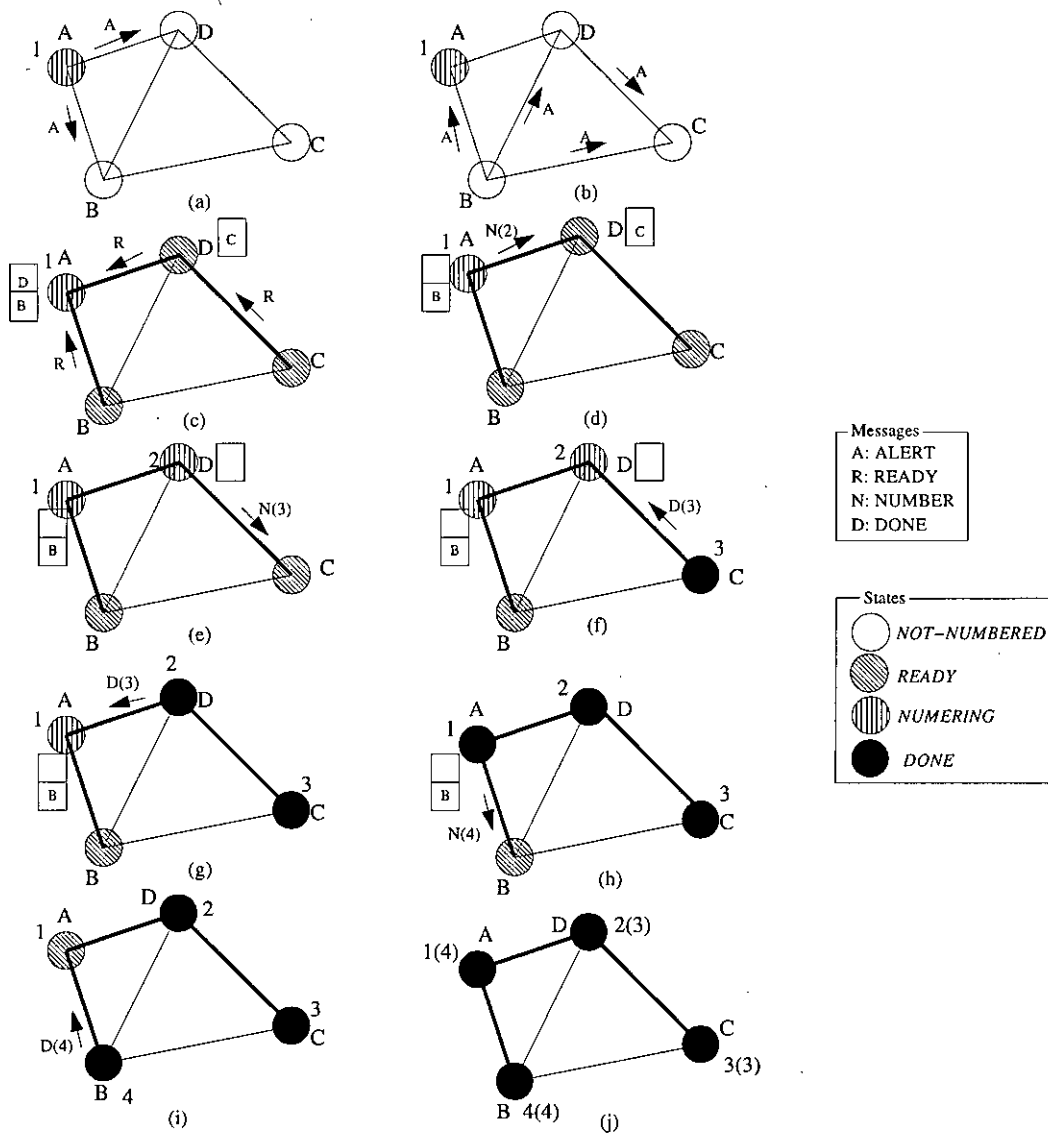


Figure 3.7: Sequence of message passing in address allocation of HN-Addressing.

to increment the address level by one. In later case, parent flags to next level addressing and issues a $NUMBER(prefix, 1)$ message to the child where $prefix$ is the address of the parent node. In next level addressing, when a child node receives the $NUMBER(prefix, k)$ message, it appends prefix address (address of the parent node) with k to construct its own address. For single level address allocation, prefix address in NUMBER message is nil. In that case only the received number forms the address.

When the parent node gets $DONE(l)$ message back from the child node while ad-

addressing in next level, it stores l as its *auxiliary number*. *Auxiliary number* counts the numbers of nodes in the subtree that are assigned addresses at the next address level. In multilevel addressing, node address is represented as decimal octet with the last octet being the *routing number* whereas *subordinate number* is shown in parentheses and *auxiliary number* in the square bracket. Each octet is bpl -bit integer greater than 0. For example, node address 1.3.4(7)[5] contains *routing number* 4, *subordinate number* 7 and *auxiliary number* 5. In section 3.6, we give the complete address allocation algorithm.

3.5 Node Failure and Address Reallocation

In our previous description we assume that no node fails or becomes inactive during the whole period of address allocation procedure. So nodes never loss any message and get their addresses comfortably. But in a sensor network failure of nodes or their inactiveness is very common and a frequent phenomena. So our addressing technique should function amidst of this failures. We differentiate node failure from its inactiveness as follows. When nodes fail, they are dead and never reappear in the scene, whereas an inactive node (say, due to going to sleep shutting down its radio for energy saving) remains dead for sometimes, but rise up again in some future time. Whatever may the cause (dead or sleep), we identify the event as node failure. A revived node is always treated as a fresh node forgetting its previous address parameters, and is assigned with new address.

We propose an address reallocation scheme in our architecture in case of node failure. In our architecture a tree like structure of nodes rooted at the sink node is always maintained. When an internal node fails, the tree structure is breached. Our reallocation scheme restores the tree structure reallocating new addresses for the affected nodes. If we are quite unlucky of getting one or more disconnected components due to node failure and tree structure cannot be maintained, we give up and there is hardly anything to do. Unless the network becomes disconnected from the sink, it can be shown that our reallocation approach works successfully.

Two types of node failure may impact the address allocation. First type of failure incorporates to nodes failure at the very beginning of network life when all nodes have not got their addresses yet. Second failure may be at the time of usual event sensing tasks after the initial address assignment for the whole network has been done. We consider both types of node failures in our address allocation scheme. One thing is to note that we consider only node failure, rather than link failure. That is, if a node is active, we assume that it never misses any message destined to it.

3.5.1 Node Failure at Address Allocation Time

Since address allocation is done at the very beginning of network deployment, it is safe to assume that nodes are full of battery power at that time. So nodes rarely become dead in that time and they might not be scheduled for go to sleep so early. Still there may be failures due to catastrophic disorder of sensor radio or unexpected malfunction of circuitries inside the sensor body. Anyway, if any node fails, it may be any of four states where nodes remain in address allocation period, namely *NOT-NUMBERED*, *READY*, *NUMBERING* and *DONE*, *DONE* being the final state. Failure at *DONE* state actually corresponds failure at event sensing time.

Failure in *NOT-NUMBERED* state: If a node fails in *NOT-NUMBERED* state before receiving any *ALERT* message from any of its neighbors, it makes no sense in our address allocation procedure. Address allocation completes smoothly leaving the dead node. If the node resumes later on, it waits to hear *ALERT* from any of its neighbors. If no *ALERT* arrives within some specific time, its broadcasts *ASSIGN-ADDRESS* message among its neighbors as a solicitation of seeking address from neighbors. Any *DONE* node may respond against this *ASSIGN-ADDRESS* message by sending *ALERT* message to the node. The node receives the *ALERT* and chooses the parent thereby. Now the revived node interacts with its new parent according to the usual address allocation procedure. After transferring a series of *READY*, *NUMBER* and *DONE* messages, the new node gets

the address. When a node gets address in this fashion, the parent node assigns address at the next address level.

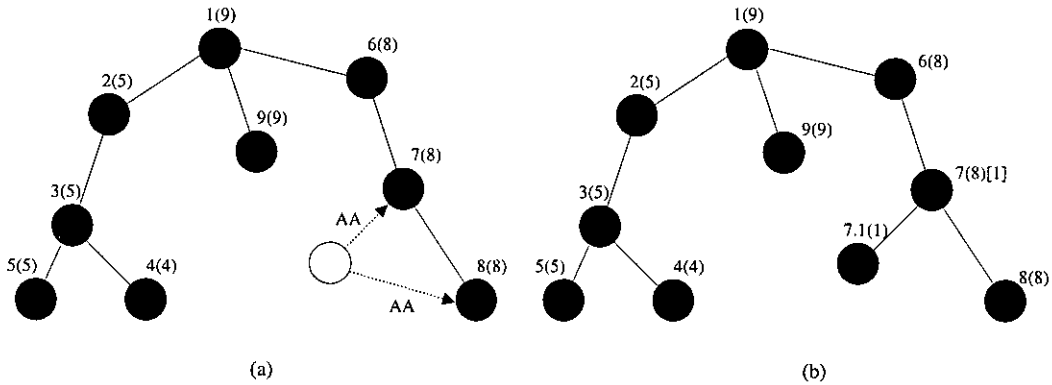


Figure 3.8: Reallocation of address. (a) node sends ASSIGN-ADDRESS message to neighbors (b) obtains address from 7.

102850

Failure in *READY* state: In *READY* state, node selects its parent and sends *READY* message to the parent. Its ID is stored in the *ready* queue of its parent node for future address assignment. But if node fails in this state, parent may not trace this failure. In course of address assignment one child after another, parent node may send *NUMBER* message to this node and wait for a *DONE* message to be returned from it. Since the node is dead, it neither receives *NUMBER* message nor sends *DONE*. The parent does not get *DONE* within a predefined time, a *DONE-TIMEOUT* is triggered and the parent leaves the child and goes for another *ready* child from its *ready* queue if there is any. If this dead node becomes alive in some future time, it becomes *NOT-NUMBERED* and obtains address as described earlier.

Failure in *NUMBERING* state: This case is pretty hazardous. *Ready* queue is destroyed and no child gets *NUMBER* message from dead parent in time. After timeout child leaves the parent, initiates the whole process by switching to *NOT-NUMBERED* state and gets address from other alive node by issuing *ASSIGN-ADDRESS* message as described earlier. A node leaving its parent sends *DISCARD-ADDRESS* message to its

children. When a node gets DISCARD-ADDRESS message from its parent, it leaves its parent, forwards DISCARD-ADDRESS to its children and tries to get address from another DONE node. A dead *NUMBERING* node fails to send DONE message back to its parent, in that case parent node is interrupted by a DONE-TIMEOUT event and it proceeds for next child.

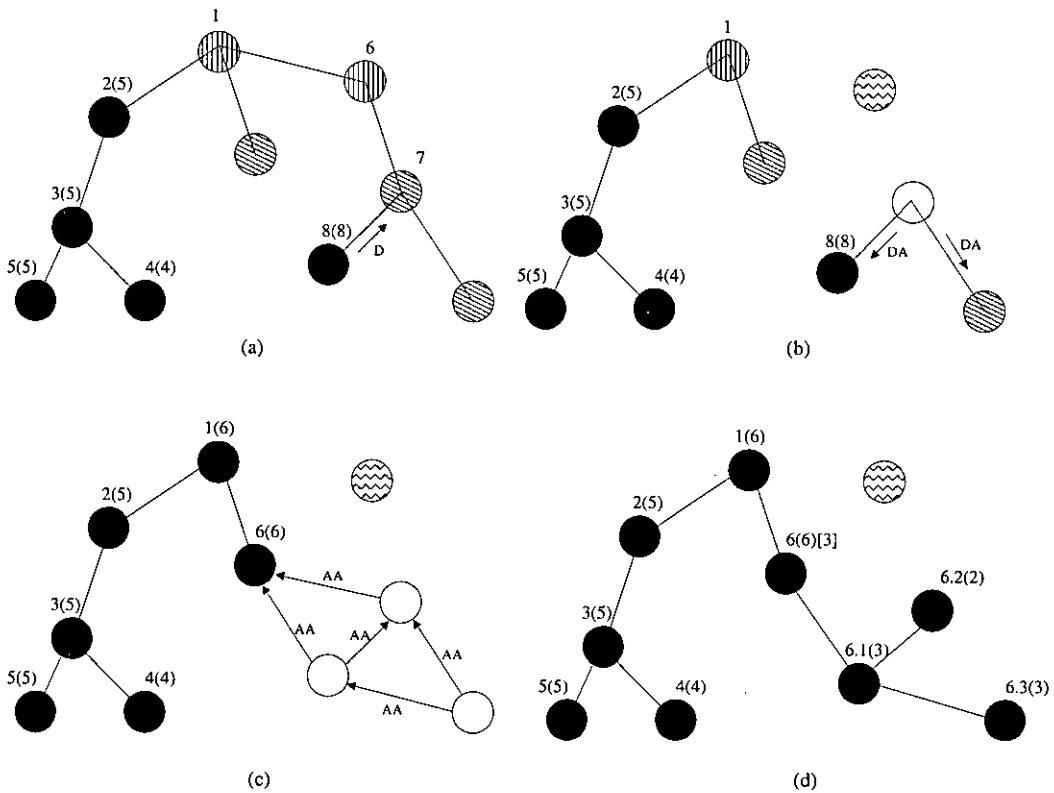


Figure 3.9: Reallocation of address. (a) Node 6 fails, (b) Node 7 detects parent failure and sends DISCARD-ADDRESS to neighbors, (c) Sink timeouts for DONE from 6 and assigns address to another child, whereas nodes leave their parent and seek address by ASSIGN-ADDRESS, (d) Nodes get addresses from 6.

3.5.2 Node Failure at Event Sensing Time

When a node fails during event sensing and data routing time long after the initial address allocation has been done, it is handled separately. If the dead node be a parent of other nodes, then its children cannot send data through it to the sink. So reporting

of event response by the sensors to the sink halts. When a leaf node fails, it actually matters nothing. If the leaf node wakes up later, it seeks address from others by ASSIGN-ADDRESS solicitation. But the death of parent matters for its children and subsequent nodes in its subtree.

The failure of the parent node can be detected in a way where neighbor nodes periodically pass some small ALIVE packets among themselves stating their status. If no ALIVE packet comes from a specific node, the node can be assumed to be dead. Another approach can be, according to *TreeCast* in [67], is to overhear the parent's transmission. When child node sends a packet to the parent to forward it onward, child can listen whether the parent is sending its packet. If no transmission is sensed, parent can be considered dead. Another way can be sending a solicitation message to the parent, if node receive no packets from the parent for long time. If solicitation is not acknowledged for certain times, child detects a dead parent.

Having detected a dead parent, child node does the following. It first leaves its parent, sends DISCARD-ADDRESS message to its children telling them to forget their current parent, goes back to *NOT-NUMBERED* state and broadcasts ASSIGN-ADDRESS to neighbors. If there is a candidate node eligible for assigning address to this node, it responds with ALERT message, and address allocation procedure begins. When this dead node wakes up later on and if it can preserve its earlier address parameters, it can resume its task as before with its previous address by little adjusting its *subordinate number* (setting it to its *routing number*) and *auxiliary number* (set it to 0). If earlier address values cannot be maintained during the failure period, it starts over again seeking address from neighbors. Figure 3.10 shows a scenario of address reallocation.

3.6 The Address Allocation Algorithm

In this section we present the algorithm of address allocation process. Figure 3.11 presents the state diagram of the entire procedure. The six messages with their fields that are used

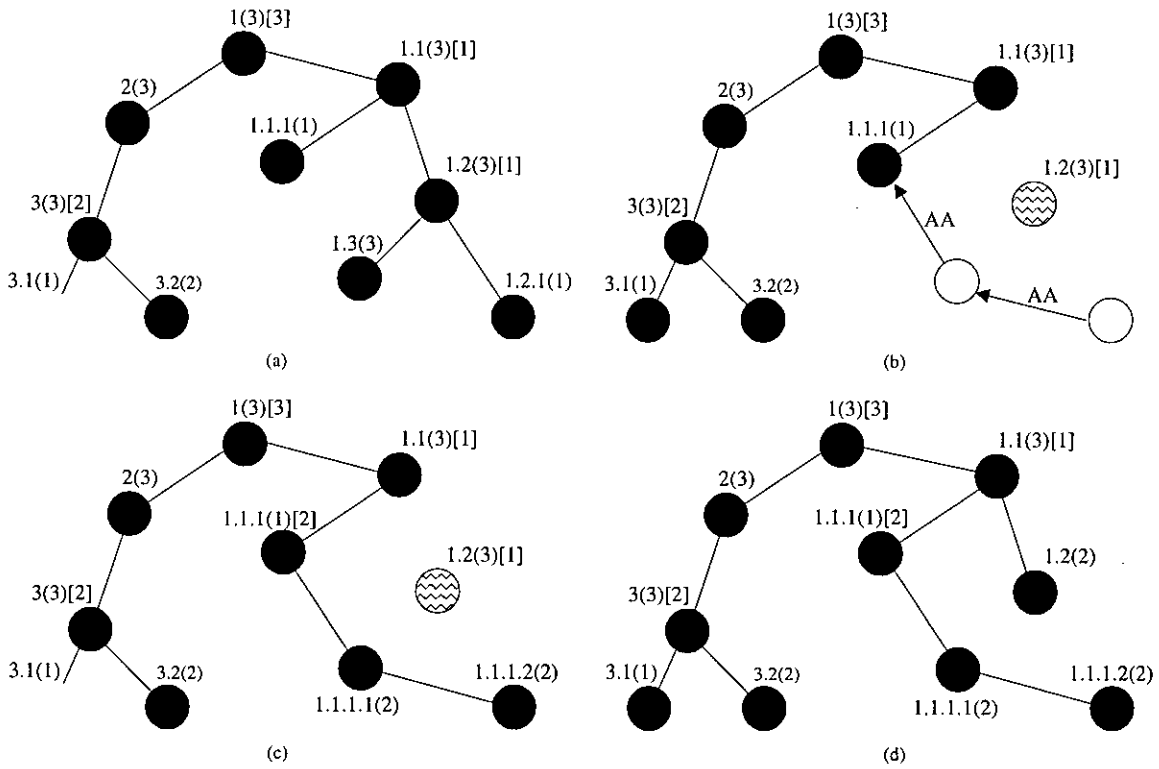


Figure 3.10: Reallocation of address. (a) Addressing for $bpl = 2$ (b) Node 1.2 fails and children get orphan (c) Orphan nodes get address from 1.1.1 (d) Node 1.2 revives and adjusts its address.

in address allocation are:

```
ALERT {
    src: Source node id who is sending ALERT
    depth: Depth of the sender node
}

READY {
    src: Source node ID
    dst: Destination ID of the message
}

NUMBER {
    src: Source node ID
    dst: Destination ID
```

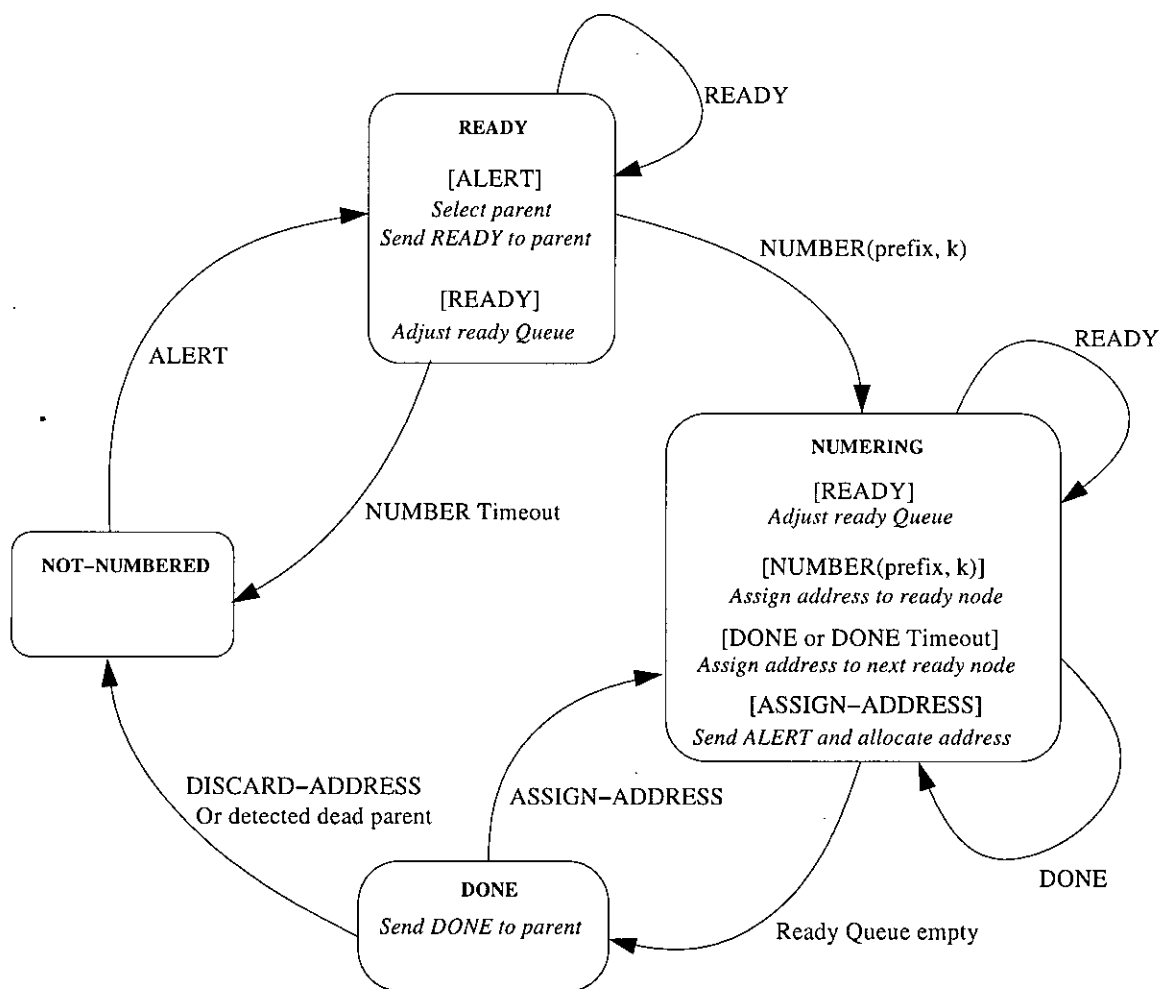


Figure 3.11: State diagram of entire address allocation procedure in HN-Addressing.

prefix: Prefix address assigned by parent

number: Number to be assigned to child

}

DONE {

src: Source node ID

dst: Destination ID

number: Routing number of sender

}

ASSIGN-ADDRESS {

```

    src: Source node ID who is requesting for address assignment
}
DISCARD-ADDRESS {
    src: Source node ID who is requesting for address discard
}

```

We adopt the following message passing syntax from Parsec [6]:

- “**Send Message**(f_1, f_2, \dots, f_k) to ID” means sending message to specific node and the Message is formed by the message fields f_1, f_2, \dots, f_k .
- “**Broadcast Message**(f_1, f_2, \dots, f_k) to neighbors” means sending message to all neighbors of the sender. Here also the Message is formed by the message fields f_1, f_2, \dots, f_k .
- “**Receive** (Message m)[condition]” means if Message m is received and the condition is true. Condition part is optional. This is statement is blocking, i.e., following statements do not execute until the specified message is received. For a message, if the condition is false, the message is dropped.
- “**Timeout** (Message) **after** Message-Timeout” means if certain type of message does not arrive within the specified time Message-Timeout. This statement should be followed by an appropriate **Receive** statement with the same Message.

Now, we give the complete algorithm of our address allocation that runs at each sensor node as a process of address allocation.

Address-Allocation-algorithm

Each node keeps the following information:

ID: Node identification number (may be hardware ID or temporarily generated)

parent: ID of the parent node

depth: Depth of the node

address: Address assigned to this node

prefix: Prefix address as assigned by the parent

rn: *Routing number* of the node

sub: *Subordinate number* of the node

aux: *Auxiliary number* of the node

state: Current state of address allocation process

Q: Ready queue, storing ID of child nodes awaiting for address

[*state*=NOT-NUMBERED] /* *The actions are in NOT-NUMBERED state */*

Receive (ALERT *a*)//* *Receiving an ALERT message */*

/* *ALERT received; choose parent */*

parent ← *a.src*;

depth ← *a.depth* + 1; /* *Node is one hop away from the parent */*

Send READY(*ID*, *a.src*) to *a.src*; /* *Send READY to parent */*

state = READY;

Broadcast ALERT(*ID*, *depth*) to neighbors;

Timeout (ALERT) after ALERT-TIMEOUT

/* *ALERT does not come in time, seek address from neighbors */*

Broadcast ASSIGN-ADDRESS(*ID*) to neighbors;

[*state*=READY]

Receive (READY *r*)[*r.dst* = *ID*] /* *Ready destined to this node */*

Q ← *r.src*; /* *Put sender ID in ready queue */*

Receive (NUMBER *n*)[*n.src* = *parent*] /* *NUMBER destined to this node */*

rn ← *n.number*;

sub ← *n.number*;

aux ← 0;

prefix ← *n.prefix*;

```

address ← concat(prefix, rn); /* Appends rn after prefix */
state = NUMBERING;

```

Timeout (NUMBER) after NUMBER-TIMEOUT

```

/* NUMBER message fails, initialize allocation */
parent ← NIL;
state=NOT-NUMBERED;

```

[state=NUMBERING]

while (Q is not empty) **do**

```

nodeId ←  $Q$ ; /* Pick a node from the ready queue */

```

if ($sub < 2^{bpl} - 1$) **then**

```

/* Assign number in current address level */

```

```

Next-Level ← false ;

```

```

Send NUMBER( $ID, nodeId, prefix, sub + 1$ ) to  $nodeId$ ;

```

else

if ($aux < 2^{bpl} - 1$) **then**

```

/* Assign number in next address level, adjust sub or aux number */

```

```

Next-Level ← true ;

```

```

Send NUMBER( $ID, nodeId, address, aux + 1$ ) to  $nodeId$ ;

```

endif

endif

Receive (DONE d) [$d.dst = ID \wedge d.src = nodeId$]

```

/* DONE received from the child */

```

```

if (Next-Level) then  $sub \leftarrow d.number$  else  $aux \leftarrow d.number$ ;

```

Timeout (DONE) after DONE-TIMEOUT

```

/* DONE times out, leave the current child */

```

```

Drop current allocation, Go for next ready node;

```

end do

if (Q is empty) **then**

/ No child left in ready queue, return DONE to parent */*

Send $DONE(ID, parent, r)$ to $parent$;

$state = DONE$;

endif

[$state=DONE$]

if ($parent$ is detected dead) **then**

$parent \leftarrow Nil$;

Broadcast $DISCARD-ADDRESS(ID)$ to neighbors;

$state = NOT-NUMBERED$;

Broadcast $ASSIGN-ADDRESS(ID)$ to neighbors; */* Seek address */*

Goto Address-allocation task at [$state=NOT-NUMBERED$];

endif

Receive ($DISCARD-ADDRESS\ dis$)[$dis.src = parent$]

/ Parent tells to forget the address, such is told to children too */*

$parent \leftarrow Nil$;

Broadcast $DISCARD-ADDRESS(ID)$ to neighbors;

$state = NOT-NUMBERED$;

Broadcast $ASSIGN-ADDRESS(ID)$ to neighbors;

Goto Address-allocation task at [$state=NOT-NUMBERED$];

Receive ($ASSIGN-ADDRESS\ asgn$)

/ Someone is seeking address */*

Broadcast $ALERT(ID, depth)$ to $asgn.src$;

Receive ($READY\ r$)[$r.src = asgn.src$]; */* READY from the alerted node */*

$Q \leftarrow r.src$;

$Next-Level \leftarrow true$;

```
state=NUMBERING;
```

```
Goto Address-allocation task at [state=NUMBERING];
```

Timeout (READY) after READY-TIMEOUT

```
/* No trace of the new node */
```

```
Ignore asgn;
```

end Address-Allocation-algorithm

3.7 Address Parameters

To analyze the address allocation scheme more formally we define the following address parameters for a sensor node s :

- $A(s)$: Address of s , say 5.1.3.4. In short, can be denoted as A_s .
- $l(s)$: Number of level in address $A(s)$. For example, $A(s) = 5.1.3.2$ has address level $l(s) = 4$.
- $A(s)[i]$: i th octet of $A(s)$, $1 \leq i \leq l(s)$. As for example of $A(s) = 5.1.3.2$, $A(s)[1]$ is 5 and $A(s)[3]$ is 3. $A(s)[i]$ can be sometimes represented as $A_s[i]$ as shorthand notation.
- $r(s)$: *Routing number* of s , last octet of address $A(s)$. Obviously $A_s[l(s)] = r(s)$.
- $d(s)$: Depth of node s , number of nodes away from the sink node.
- $sub(s)$: *Subordinate number* of s . As the addressing indicates $r(s) \leq sub(s) < 2^{bpl}$.
- $aux(s)$: *Auxiliary number* of s . We can verify that $1 \leq aux(s) < 2^{bpl}$.

Let us define a function *match*: Address \times Address $\rightarrow Z^+$ which counts the largest prefix match in two addresses. For example *match*(1.5.2.3, 1.5.4.2) gives 2 whereas

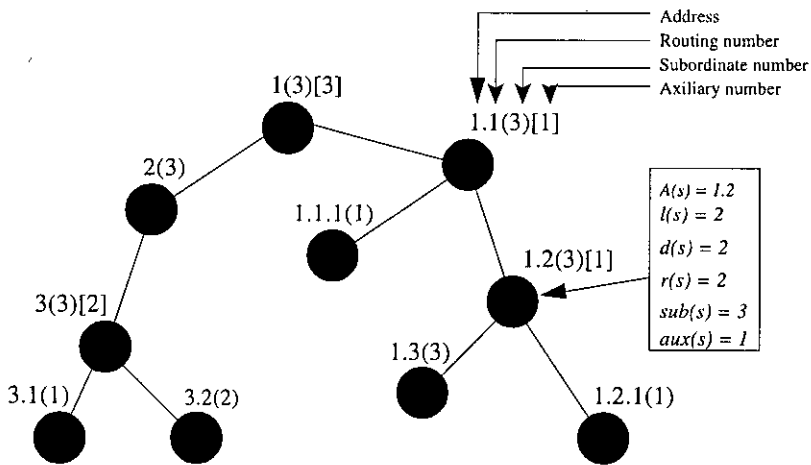


Figure 3.12: Various address parameters in HN-Addressing.

$match(6.2.1, 1.2)$ gives 0. The function $match$ for addresses of two nodes x and y can be defined as:

$$match(A_x, A_y) = \min\{i : A_x[i] \neq A_y[i]\} - 1 \text{ for } 1 \leq i \leq \max(l(s_1), l(s_2))$$

If addresses are equal i.e., $A_x = A_y$, then $match(A_x, A_y) = l(x)$.

We have the following properties of address regarding parameters of nodes:

Property 3.7.1 *If node x is the parent of node y , then $d(y) = d(x) + 1$.*

Proof. It simply follows from the address allocation scheme. When a node receives the ALERT message from its parent, it computes its depth by adding one more with parent's depth. So child node is exactly one hop away from the parent. \square

Property 3.7.2 *For any node y , $l(y) \leq d(y) + 1$.*

Proof. In the worst case of address length, address level is increased by one at every depth of the tree. In that case, number of levels in address for the node is exactly one more than its depth. Number of address levels cannot go beyond this depth limit. \square

Property 3.7.3 *If node x is the parent of node y , then $l(y) = l(x)$ or $l(y) = l(x) + 1$.*

Proof. Whenever a parent node assigns address to its child (by sending NUMBER message), it may either assign number in the current address level (if $k < 2^{bpl} - 1$) or increase the address level by one. In former case, number of address levels of both parent and child is same ($l(y) = l(x)$), but for later case, child's address level is one more than the parent's one ($l(y) = l(x) + 1$). Such is the claim of the property. \square

Property 3.7.4 Detection of Child. *A node y is the child of node x (reversely, x is parent of y) if and only if $d(y) = d(x) + 1$ and any of the following two conditions is satisfied by their address parameters for $m = \text{match}(A(x), A(y))$:*

I. $l(x) = l(y) = m + 1$ and $r(x) < r(y) \leq \text{sub}(x)$.

II. $l(x) = l(y) - 1 = m$ and $1 \leq r(y) \leq \text{aux}(x)$.

Proof. The $d(y) = d(x) + 1$ condition is due to Property 3.7.1.

To the prove the property, we are to show that if x be the parent of y , any of the two mentioned conditions holds, and on the contrary if any of the two conditions holds, x should be parent of y .

The proof is derived from two address allocation situations in HN-Addressing. When a parent node assigns address to one of its child by sending NUMBER message, one of two things can happen. Parent does assign number to the child in the last address level (if $k < 2^{bpl} - 1$) or increase the address level by one and assigns number in next address level. The proof can be constructed as follows for the two cases:

Case I Addressing in last level. Let x is the parent of y . In this case both x and y have the same number of address level. So $l(x) = l(y)$. Since y receives address from x , a NUMBER(*prefix*, k) message is issued to y from x where *prefix* is the address of x without the last octet. y appends k after *prefix* to form its address. So A_x and A_y differs only in last octet resulting $m = \text{match}(A_x, A_y) = l(x) - 1$. Again $\text{sub}(x)$ denotes the maximum number that is obtained by any node in the subtree of x in the last address

level of $A(x)$, so y 's routing number should lie within $r(x)$ and $sub(x)$. Hence we get $r(x) < r(y) \leq sub(x)$, and condition I is satisfied.

Again we assume $l(x) = l(y) = m + 1$ and $r(x) < r(y) \leq sub(x)$. We are to show that x is parent of y . Now, $A(y)$ differs from $A(x)$ exactly in the last octet and y 's routing number lies within $r(x)$ and $sub(x)$. Again $d(y) = d(x) + 1$ makes y exactly one hop away from x . Hence y must have received address from x . So x is the parent node.

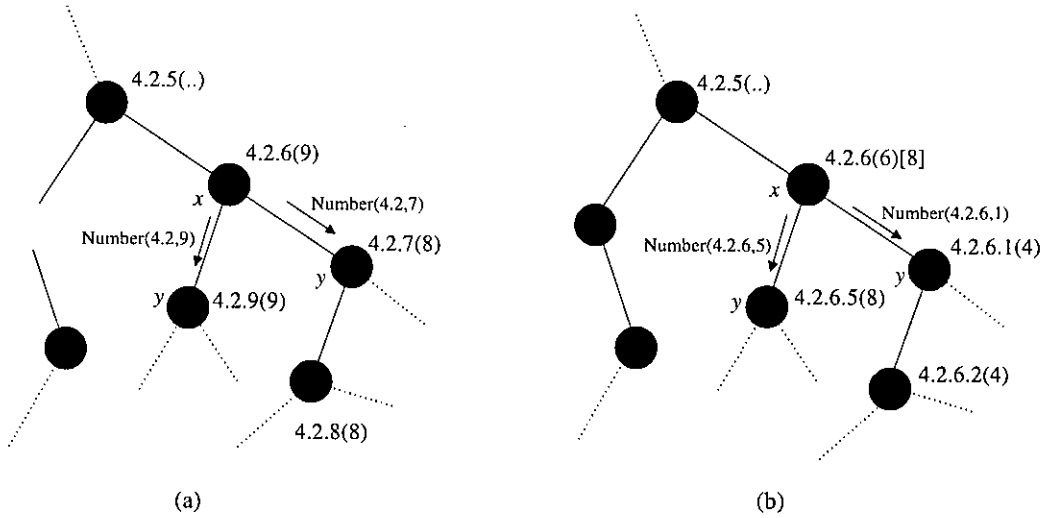


Figure 3.13: Demonstration of property 3.7.4. (a) Condition I: y gets address from x in the last address level (b) Condition II: y gets address from x in the next address level

Case II Addressing in next level. Let x is the parent of y . In this case, y has one more level in its address than x 's and x 's address is complete prefix of y 's address. So $l(x) = l(y) - 1$ and $m = l(x)$. In our address allocation procedure y must have been assigned a number between 1 and $aux(x)$ by the parent x and this number appears as y 's routing number. So $1 \leq r(y) \leq aux(x)$.

Again if $1 \leq r(y) \leq aux(x)$ holds and y 's address has complete prefix match with x (being $m = l(y) - 1 = l(x)$) and y is one hop away from x ($d(y) = d(x) + 1$), y must have taken address from x . So x is the parent of y . □

Due to Property 3.7.4, parent node can determine its child from its address and similarly child nodes can determine its parent. As we see Figure 3.14, nodes can determine their parent depending the address parameters of the nodes in the network.

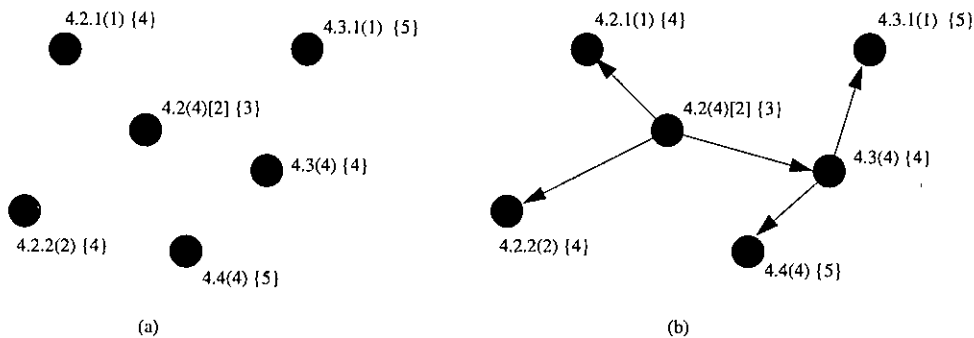


Figure 3.14: Detecting parent nodes (a) Bare nodes (b) Underlying tree (Number inside {} means depth).

Property 3.7.5 Detection of subtree node. For two nodes x and y with $d(y) \geq d(x)$ and $m = match(A_x, A_y)$, the node y lies in the subtree of x in the tree, if it satisfies any of the two following conditions:

I. $m = l(x) - 1$ and $r(x) < A_y[m + 1] \leq sub(x)$.

II. $m = l(x)$ and $1 \leq A_y[m + 1] \leq aux(x)$.

Proof. The property directly follows from the address allocation technique. When a parent assigns address to its children, it either assigns address in its last address level or starts from 1 in the next address level. And this continues beneath the tree up to the leaves. Let x be the root of a subtree. Two cases can happen.

Case I. When the subtree nodes get address of the same address level of x , their addresses differ only in the last octet and other octets are equal to $A(x)$'s. In this case, if y be a subtree node, we have $m = match(A_x, A_y) = l(x) - 1$. So $A_y[m + 1]$ denotes the last octet in A_y . According to our addressing scheme this last octet is bounded by

$r(x) < A_y[m + 1] \leq sub(x)$. When y in turn assigns address to its children (y acts as parent) with longer address level, additional octets are added after $m + 1$ level, keeping first m address levels exactly same. Those nodes are also included in the subtree of x .

Case II. When nodes are assigned address with one additional address level of their parent, they take a new level and their addresses exactly match with the parent in all octets except the last. The number in additional address level is bounded by the *auxiliary number* of the parent. If x be a parent and y be a subtree node, we have $m = match(A_x, A_y) = l(x)$. So $A_y[m + 1]$ denotes number in additional address level, it is bounded by $1 \leq A_y[m + 1] \leq aux(x)$. When this y assigns nodes with longer address levels, those nodes also appear in the subtree of x . □

Due to the Property 3.7.5, any node can determine whether a destination node resides in its subtree by examining the destination address. In that case, node participates in forwarding of the packet enroute to the destination. This is how HN-Addressing makes *stateless routing* based on assigned address.

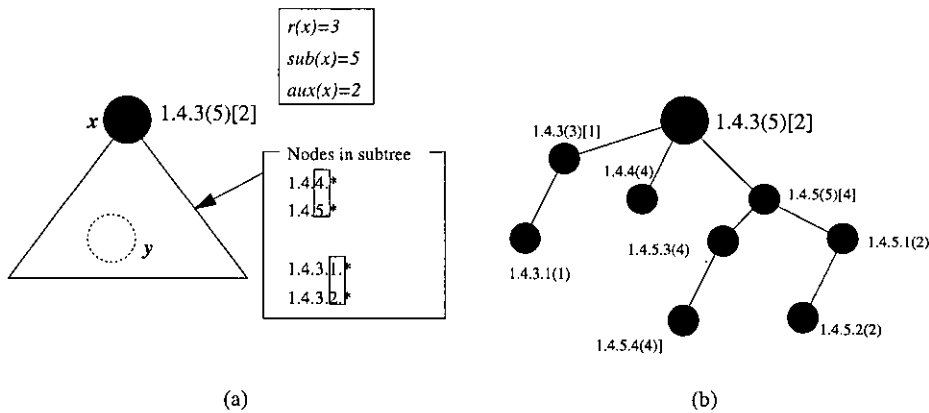


Figure 3.15: Illustration of Property 3.7.5. (b) Possible nodes in the subtree of 1.4.3(5)[2].

3.8 Routing

Once each node of the network has been assigned with address, every node in the network contains the following information:

- *Address*: The address that is assigned to it by the address allocation process.
- *Depth*: Count of number of hop away from the sink node.
- *Routing Number*: The address number itself for single level address and last octet of the address in case of multi-level address.
- *Subordinate Number*: The maximum *routing number* as obtained by any node in current address level in the subtree of the node.
- *Auxiliary Number*: The maximum *routing number* as obtained by any node in the next address level in the subtree of the node.

These information form the routing entity of the node. Any routing decision can be made depending in these numbers. Nodes do not require any precomputed routing table or previous history of destination route. Hence the routing technique is called *stateless*. In *stateless* routing nodes only know where to send a packet (i.e., the address of the destination node), but they do not know anything about the current position of the node or possible route to the destination. Rather they simply broadcast the packet to their neighbors and neighbor node depending on the address of the destination relays the packet en route to the destination. Address allocation scheme assigns address to the nodes in such a way that exactly one node from the neighbors is ensured to participate in forwarding of data packet to a destination node.

We define three types of packets that are normally encountered in sensor networks:

- I. *Query packet*: This packet is issued by the sink node and is flooded into the entire network. All nodes should receive the packet and forward it accordingly.

- II. *Request packet*: Sometimes the sink passes some specific control data to any specific sensor node. This packet contains the address of the destination node. If packet cannot be reached by direct broadcast by the sink, other nodes in the middle forwards the packet to reach the destination.
- III. *Response packet*: In response to the sink's query on any specific event, a node can send response packets to the sink if it gathers relevant data of the queried event. Response packet is destined to the sink and it contains the source node's address along with the event data. If packet cannot be reached to the sink directly, other nodes route it to the sink. In the application of periodic observation, sensors might send response packet with their observed data to the sink, without the prior query from the sink.

For a packet p of each type, we define the following packet fields:

Query packet

- $A(p)$: Address of the forwarder node of the query. Initially it is sink's address.
- $sub(p)$: *Subordinate number* of the forwarder node of the query packet.
- $aux(p)$: *Auxiliary number* of the forwarder node of the query packet.
- $d(p)$: Depth of the forwarder node.
- $query(p)$: Query data of interest.

Request packet

- $D(p)$: Address of the destination node to which request is to be reached.
- $d(p)$: Depth of packet forwarder node. This field is updated each time the packet forwarded by a node.
- $data(p)$: Requested data of interest.

Response packet

- $S(p)$: Address of the node that generates the response packet destined to the sink.
- $d(p)$: Depth of packet forwarder node. This field is updated each time the packet forwarded by a node.
- $data(p)$: Response data of the observed event.

3.8.1 Routing Rules

Below we discuss the routing paradigm in HN-Addressing for the above three types of packets that most frequently appear in sensor networks.

Query Packet: Query is one type of network layer broadcast of a packet by the sink node. When the sink node wants a specific type of information of certain event, it makes a query to every node in the network flooding its interest into the entire network. When a node gathers the announced data from the surrounding environment, it makes a response to the sink sending the observed data. Query packets are originated from the sink. When the query packet q is first issued it is constructed with the packet fields as $A(q) = A(sink)$, $sub(q) = sub(sink)$, $aux(q) = aux(sink)$ and $d = 0$ and arbitrary query data of interest in $query(q)$. The packet is then broadcast to its neighbors and neighbors in turn broadcast to their neighbors and so on. One node can receive query packet from multiple sources, but it forwards the packet if the packet is sent from its parent. A node forwards a query if it receives it from its parent, otherwise it drops it. So for a node x , we can have the following query routing rule.

Routing Rule: Query Packet at [Node x]

Receive (Query q)

If (node $A(q)$ is the parent of x)

- Accept and record q ;

- Set $A(q) = A(x)$, $sub(q) = sub(x)$, $aux(q) = aux(x)$ and $d(q) = d(x)$;

- Broadcast q to neighbors;

Else

Drop q ;

End Rule

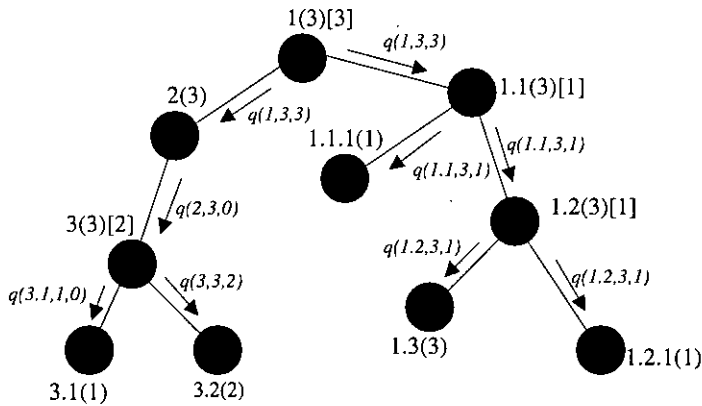


Figure 3.16: Flooding query packet in the entire network.

Request Packet: Request packets are issued by the sink node to pass specific control, management or query message to a specific sensor node. Packet is reached to the destination forwarded by some intermediate nodes, if the destination is not directly reachable from the sink. The request packet p is constructed by the sink by setting packet fields as $D(p) =$ destination address, $d(p) = 0$ and $data(p) =$ Control data to pass. Sink broadcasts the packet to its neighbors. Among the neighbor nodes, the node whose subtree contains the destination node, forwards the packet onward and other drops the packet. As our addressing scheme ensures, there is only such node in the neighborhood of the forwarder.

Routing Rule: Request Packet at [Node x]

Receive (Request p)

If ($D(p) = A(p)$)

- Destination reached;
- Accept and record p ;

Else If $(d(x)=d(p)+1$ and $D(p)$ lies in the subtree of x)

- Accept p ;
- Set $d(p) = d(x)$, keep other packet fields unchanged;
- Broadcast p to neighbors;

Else

Drop p ;

End Rule

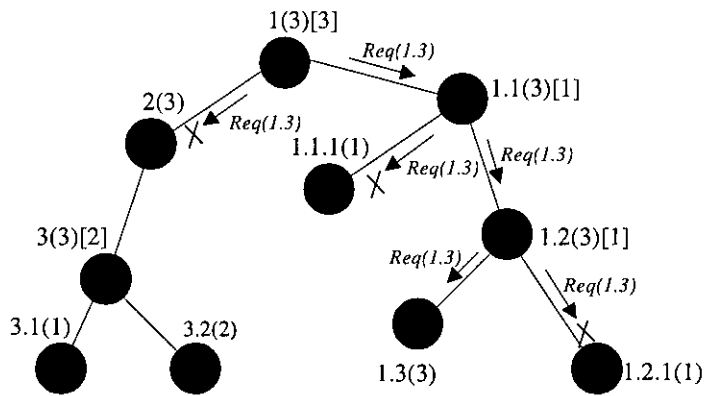


Figure 3.17: Sink sending request packet to node 1.3. (Packet does forward after the node indicated by cross)

Response Packet: When a node gathers any data about an event that the sink expressed interest for by query packet, the node sends the event data by response packets destined to the sink. The response making node prepares the response packet p by putting packet fields as $S(p) =$ node address, $d(p) =$ depth of the node and $data(p) =$ event data to be passed to the sink. Then it broadcast the response packet to its neighbors. The parent of this node accepts the packet, forwards it again. Thus packet finally reaches to the sink. Since a node has only one parent, only one node participates in forwarding

of packet backward to the destination. We build the following routing rule for this sort packet.

Routing Rule: Response Packet at [Node x]

Receive (Response p)

If (x = sink)

- Destination reached;
- Accept and record p;

Else If ($d(x)=d(p)-1$ and $S(p)$ lies in the subtree of x)

- Accept p;
- Set $d(p) = d(x)$, keep other packet fields unchanged;
- Broadcast p to neighbors;

Else

Drop p;

End Rule

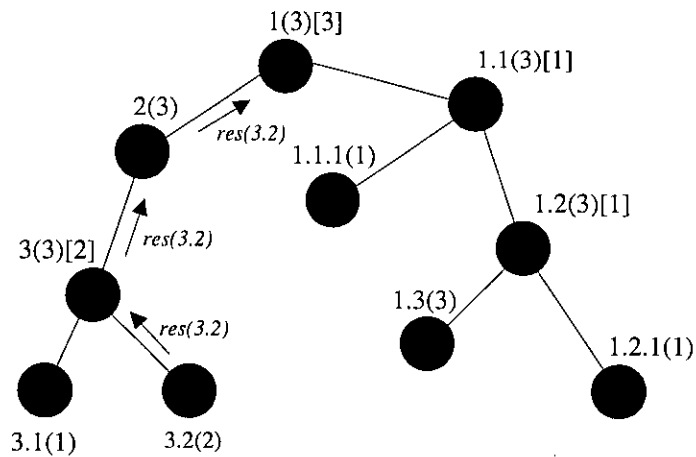


Figure 3.18: Node 3.2 sends response packet to the sink.

Chapter 4

Simulation Results

In this chapter we describe the simulation results of HN-Addressing, the proposed addressing and routing architecture for sensor network. Through simulation we study the behavior of our approach and evaluate its performance based on some performance metrics. We also compare the performance of our approach to *TreeCast* described in [67].

4.1 Simulation Setting

We simulate HN-Addressing using Parsec [6], a C-based parallel discrete event simulation language developed in UCLA Parallel Computing Laboratory. The main objective of the simulation is to understand the impact of various parameters on the performance issues of the protocol. We also simulate *TreeCast* to make a comparison between our approach and *TreeCast* on various metrics. In the following sub sections we describe the various settings of our simulation. Table 4.1 presents the entire simulation setting.

4.1.1 Simulation Environment and Parameters

We run our simulation for sensor networks of size ranging from 100 to 500 nodes which are placed uniformly in a two dimensional area of 1000m \times 1000m (from $x = 0$ to $x = 1000$

Table 4.1: Simulation Setting.

Parameter	Value
General Settings	
Target Area	1000m × 1000m
Number of nodes (N)	50 - 300
Usual number of nodes	200
Sink location	(500, 500)
Transmission radius (R)	125m
Node density $\mu(R)$	10 (Approx.)
Bits per Level (bpl)	3 - 10
Number of runs	5
Channel Characteristics	
Propagation model	Friis free space (r^2 path loss)
Signal transmission range	Fixed (125m)
Channel	Symmetric
Energy Consumption	
Radio model	First Order Radio Model
Energy for Transmitter Electronics ($E_{Tx-elec}$)	50 nJ/bit
Energy for Receiver Electronics ($E_{Rx-elec}$)	50 nJ/bit
Energy for Transmitter Amplifier (ϵ_{amp})	100 pJ/bit/m ²
Energy/bit	≈ 1.56 J/bit
Event Reporting	
Reporting type	Discrete Event-driven
Packet type	Request, Response, Query
Event generation process	Poisson
Inter-arrival of events	Exponential (mean 1000 STU*)
Event generator node selection	Uniform random
Total packet transmission	100
Node Failure Model	
Failure Model	Continuous and Consecutive
Time gap between two failures (Continuous)	Exponential (mean 1000 STU)
Sleeping time (Consecutive)	Exponential (mean 100 STU)
Failing node selection	Uniform random
*STU means Simulation Time Unit	

and $y = 0$ to $y = 1000$ in X-Y plane). In most cases, if the performance evaluation is to be made on parameters other than number of nodes, we keep number of sensor nodes as 200. The choice for this number is due to the simulation of *TreeCast* [67] which also uses the same number of nodes. In all cases, we assume the location of the sink is in the center of the scene at (500, 500). Each node is assumed to have a transmission radius of 125 m. This gives node density i.e., the average number of nodes in the transmission area of a node to be $\frac{200}{1000 \times 1000} \times (\pi \times 125^2) \approx 10$ nodes per transmission area. This node density signifies that every node has approximately 10 neighbor nodes on the average. For computing average values, we run the simulation for five times with the same setting, but different scene of node deployment. Figure 4.1 shows a typical node deployment scene with 200 nodes and right figure shows the generated tree by HN-Addressing.

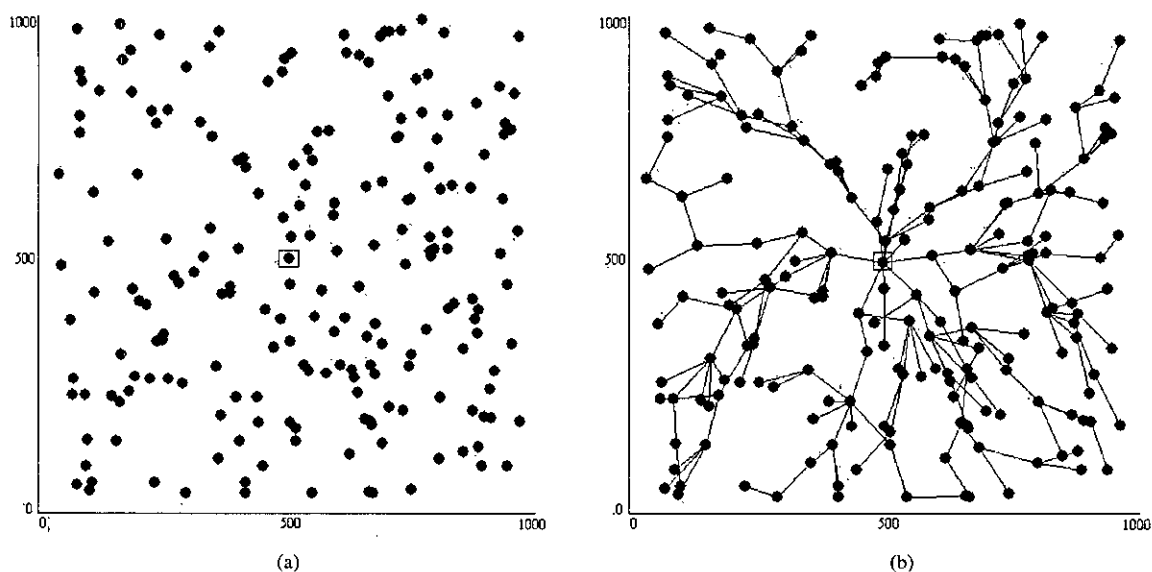


Figure 4.1: Scenario of 200 nodes in 1000m \times 1000m area. (a) Nodes are scattered randomly in the area (b) Generated Tree for the scene (Squared node indicates *sink*).

4.1.2 Energy Consumption Model

In our simulation we consider the following radio and energy consumption model for the sensor network. We only consider the energy consumption due to data communication, no power consumption for processing event data is considered. In computation of communication energy, we mainly concentrate on energy consumption related to address overhead in packet.

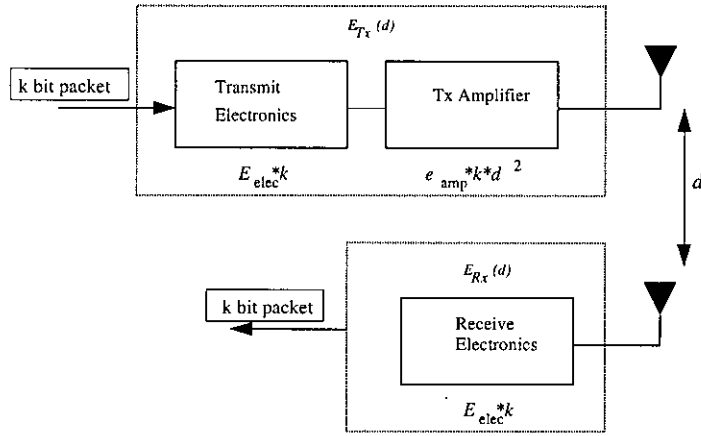


Figure 4.2: First Order Radio Model for sensor node.

Radio Model. In our simulation we assume a simple first order radio model for the sensor node, as suggested by Heinzelman in [37], where the radio in each sensor consists of a transmitter and receiver as shown in Figure 4.2. Both transmitter and receiver circuitry dissipate energy whenever transmitting and receiving packets. For transmitting or receiving a packet of same length, the energy dissipation for the corresponding circuitry is assume to be the same. This energy is called *Electronics Energy* and denoted by E_{elec} . While transmitting packet, transmitter amplifier amplifies the signal and it requires ϵ_{amp} energy per bit for this. The model also assumes Frii free space propagation model for radio channel where signal strength falls in square order of distance. Thus, to transmit a k -bit message a distance d , energy dissipation by the radio is given by:

$$E_{Tx}(k, d) = E_{Tx-elec}(k) + E_{Tx-amp}(k, d)$$

$$E_{Tx}(k, d) = E_{elec} \times k + \epsilon_{amp} \times k \times d^2$$

and to receive the same length message, the radio expends:

$$E_{Rx}(k, d) = E_{Rx-elec}(k)$$

$$E_{Rx}(k, d) = E_{elec} \times k$$

Detail technical treatment of this radio model is due to [38]. We assume that the sensor radio dissipates $E_{elec} = E_{Tx-elec} = E_{Rx-elec} = 50$ nJ/bit (50×10^{-9} J/bit) to run the transmitter and receiver circuitry and $\epsilon_{amp} = 100$ pJ/bit/m² (100×10^{-6} J/bit/m²) for the transmitter amplifier to achieve an acceptable $\frac{E_b}{N_0}$. According to these values, a sensor node takes nearly 1.56J^1 to transmit a single bit to the at most 125m away neighbor.

Symmetric Channel. We make the assumption that the radio channel is symmetric such that the energy required to transmit a message from node A to node B is the same as the energy required to transmit the message of same length from node B to node A for a given SNR (signal-to-noise ratio).

Data Reporting. In the application of sensor networks, sensor nodes usually senses event or measure some physical condition (like temperature) in their vicinity. Two types of event reporting can be possible [37]: continuous and discrete. In continuous reporting sensors periodically send their observed data to the sink in some regular interval and this normally happens when sensors are deployed to observe some ambient condition of the target environment. But in discrete reporting, sensors send data to the sink whenever a certain event has been occurred in the sensing area and a sensor detects it. In our simulation we conduct a discrete type “event-driven” data reporting technique where

¹ $E = E_{elec} + \epsilon_{amp} \times d^2 = 50 \times 10^{-9} + 100 \times 10^{-6} \times 125^2 \approx \frac{125^2}{10^4} \approx 1.56\text{J}$

sensors are to send data to the sink at some random time, and the sink sends request packet to some random node at some random time. Source node (for sending response packet) or target node (for sending request packet) are selected using a uniform distribution over the network nodes. Event reporting is assumed to be a Poisson process with a mean inter-arrival time of 1000 simulation time. Each data packet is assumed to contain fixed data of size 16 bits irrespective of request, response and query. The choice for this is due to [3] which argues that sensor's data is normally small and varies from 16 to 48 bits.

4.1.3 Node Failure Model

Node failure is a very frequent phenomenon in sensor networks. Now and then nodes can fail because of mechanical or electrical malfunction or complete drain out of stored battery power. Node failure can be identified by the inactivity of corresponding node in sending and receiving messages. An inactive node (or failed thereby) can neither send nor receive messages. A node can be inactive in two ways: being dead or going into sleep for energy saving. In case of death, node dies permanently and can never be revived into the scene later on. In case of sleeping, nodes turn off their radio and become inactive for extended period of time. In later case, node resumes its operation after some pre-scheduled period. Whatever may be the case, we identify node's inactivity as node failure. In our simulation, we adopt two failure models:

Continuous failure. Nodes fall to death and never resume afterward. In simulation of this model, nodes are selected randomly to die and the time gap between two successive death of node is exponentially distributed with a specified mean time. Our model uses this mean time as 1000 in simulation time unit and we chose randomly the next node to die. In continuous failure model, number of active nodes gradually decreases and at one time the network becomes disconnected.

Consecutive failure. Nodes go into sleep and wake up later on after some predefined time. During this sleeping period, radio of the node remains inactive. In this model, the sleeping period of a node is assumed to be exponentially distributed with mean 1000 simulation time unit and the selection of sleeping node is made randomly (uniform) from all active nodes.

4.1.4 Performance Metrics

The performance metrics that we consider in our simulation are *Message Cost for Addressing*, *Address Length* and *Communication Energy*.

Message Cost for Addressing is the number of messages required to assign addresses to all nodes. In address allocation phase, nodes transfer *ALTET*, *READY*, *NUMBER* and *DONE* messages to themselves to allocate address. *Address Length* is the average address length in bit that nodes take in address allocation phase. The protocol assigns addresses with different levels to nodes at different depth. Address with more level generates longer address in bit. For example, if bits per level (*bpl*) is 4, address with level 3 becomes 12 (4×3) bits in length. One of the main concentration of our protocol is to keep this address length shorter. Another performance metric is *Communication Energy* that measures the total energy dissipation by sensors in sending and receiving data packets. Simulation shows the impact of address length on packet communication. It is observed that communication energy is reduced for shorter address length.

4.2 Performance Evaluation of HN-Addressing

In simulation, we vary number of nodes (N) and bits per level (*bpl*) to observe the behavior of some performance metrics like *Message Cost for Addressing*, *Address Length* and *Communication Energy*. To limit the illustrations, we depict only some particular simulation-based scenarios, where subtle changes in various parameters (e.g., number of

nodes, bits per level, etc.) result in salient differences in the behavior of our performance metrics. We also consider failure of nodes and evaluate the performance thereafter accordingly.

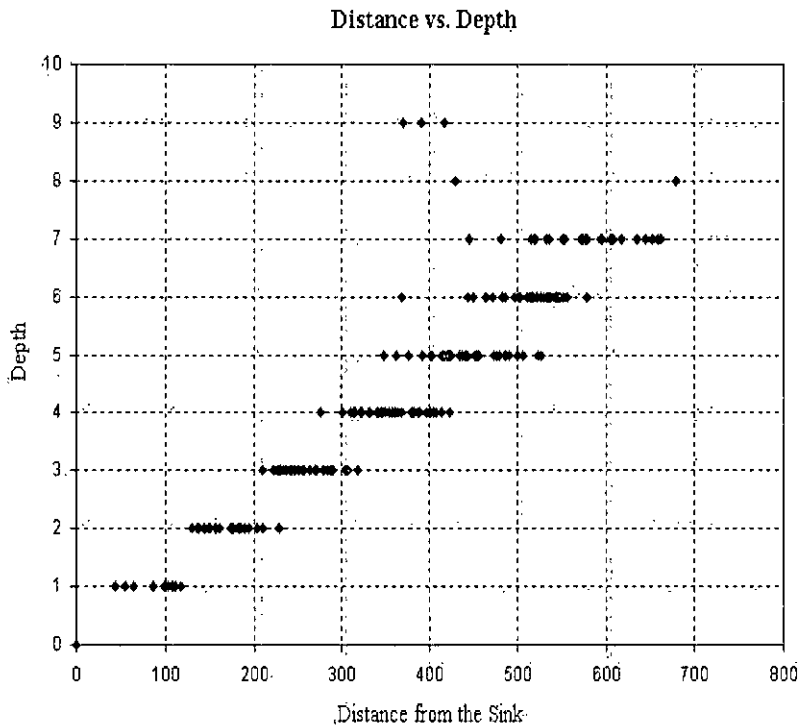


Figure 4.3: Depth of nodes against the distance from the *sink*.

4.2.1 Tree Formation

As we saw earlier, in HN-Addressing protocol sensor nodes are organized in a tree rooted by the *sink*. In the tree, each node is given a depth value that counts the number hops required to reach the *sink* from that node. This tree formed by sending and receiving of *ALERT* and *READY* messages by the nodes. Figure 4.1(a) illustrates the scattered position of 200 nodes in the environment and 4.1(b) shows the corresponding tree. To show explicitly the quality of the tree formed by the algorithm, we produce Figure 4.3, which shows the Euclidean distance of a node from the *sink* on the X-axis, and the depth of the node on the Y-axis. It is clear that most nodes are likely to be in the smallest

depth according to the distance from the sink, and the rests are within 1 more depth of the smallest possible depth. Sometimes nodes do receive address allocation messages from neighbor of higher depth rather than from one of lower depth. This makes multiple depths to exist at the same distance for different nodes (e.g., at a distance of 500m from the sink).

4.2.2 Message Cost for Addressing

This metric corresponds to the count of total messages required to allocate addresses to all nodes in the network at the initialization phase. We know that four messages are mainly used in address allocation algorithm: *ALERT*, *READY*, *NUMBER* and *DONE*. There are also two other messages *DISCARD-ADDRESS* and *ASSIGN-ADDRESS* that are used when a node fails or does not receive necessary messages from its parent. Figure 4.4 shows the message count for addressing by HN-Addressing. The impact of two parameters, number of nodes and bits per level (*bpl*) is discussed below:

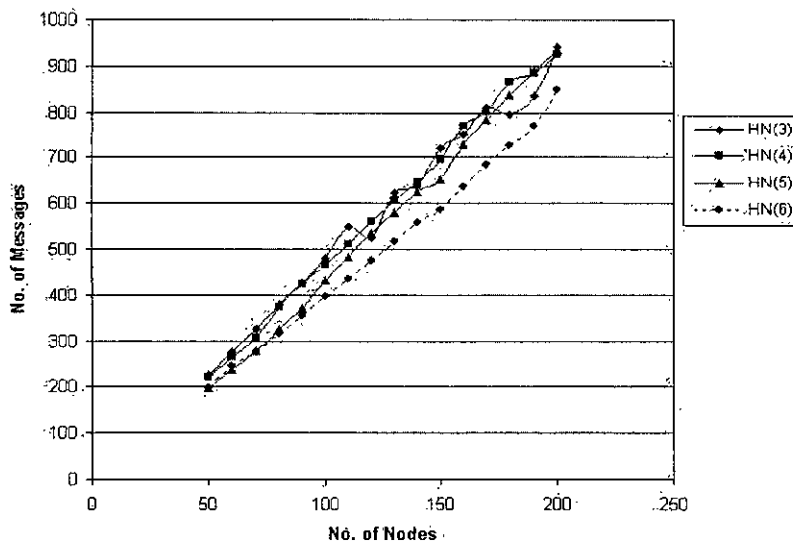


Figure 4.4: Message Cost for Addressing in HN-Addressing. HN(x) indicates HN-Addressing with $bpl = x$.

Impact of number of nodes. As Figure 4.4 reveals message cost for addressing nodes is linearly dependent on number of total nodes. From the curves in figure, it can be observed that message count is nearly four times of the number of nodes, i.e., Message Cost = $O(4N)$, where N is the number of total nodes in the network. This four times is due to the fact that if no node ever fails and all events go smooth, every node sends *ALERT*, *READY* and *DONE* messages exactly once and receives *NUMBER* message once. In some cases, when a node in *READY* state does not receive *NUMBER* message from its parent within time and a *READY-TIMEOUT* occurs, the child node leaves the parent and collect number from other suitable node. In that case more messages are required to assign addresses to nodes.

Impact of bits per level (bpl). The number of message count is slightly greater than four times of number of nodes. The count of extra message over $4N$ is dependent on the value of bits per level (bpl). Smaller value of bpl requires more messages than the larger one. Extra messages are required when a *ready* node cannot collect number from its parent and has to select another parent and collect number from the same. A parent node may not be able to assign number to its children when it runs out of its all possible numbers of the same level and the next level. It happens when the parent's *subordinate number* and *auxiliary number* reaches their maximum possible value, i.e., *subordinate number* = $2^{bpl} - 1$ and *auxiliary number* = $2^{bpl} - 1$. In that case the node cannot assign any new number to any of its child node. Figure 4.5 depicts the scenario. This event occurs more frequently for smaller value of bpl , when the number space becomes narrower. As we see in figure 4.4, message count is little higher for $bpl = 3$ and it gradually gets lower for higher value of bpl , as for $bpl = 4, 5$ and 6 . In Figure for $bpl = 6$, message count becomes fewest.

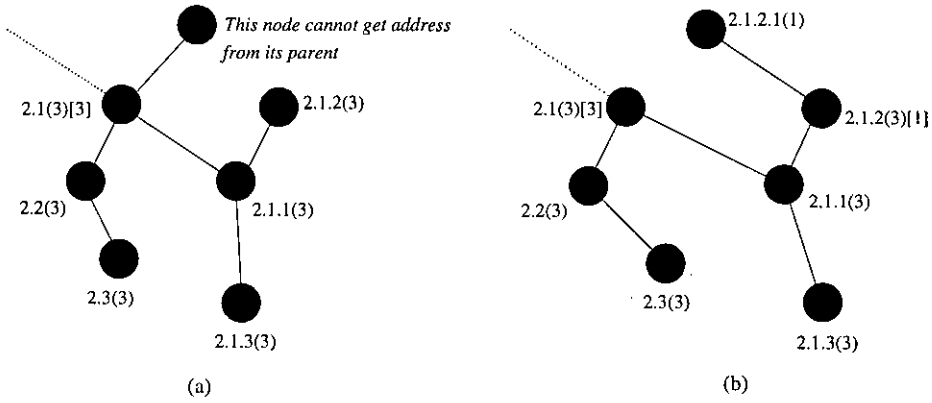


Figure 4.5: Parent fails to assign address to child. (a) Parent 2.1 cannot give address to one of its child node, (b) child changes the parent and gets address 2.1.2.1 from 2.1.2.

4.2.3 Address Length

Address length is the principal performance metric of our simulation. It measures the average length of address of all sensor nodes in the network. For any node, address length is computed as product of bits per level and address level of that node. For example, a node having address level 3 with an address, say 1.3.4, would have address length 12 ($= 3 \times 4$), if $bpl = 4$. Since nodes at different depth get different address level, address length varies from node to node. Since address makes overhead in packet communication, reducing average address bits is one of the primary objectives of our protocol.

Recalling the notation from earlier chapter, we compute the average address length as obtained by each node in the network, as follows:

$$\text{Average address length} = \frac{1}{N} \sum_x (l(x) \times bpl)$$

bits where $l(x)$ is the number of address level of node x and N is the number of total nodes in the network. If nodes are numbered without level hierarchy, i.e., all nodes are assigned address in the single level, number of address bits should be at least $\lceil \log_2(N) \rceil$.

The value of bits per level (bpl) has a subtle impact on average address length. bpl sets a restriction on the maximum number of nodes that can be accommodated in the same address level. Higher bpl value makes address length longer, but it accommodates more sensor

nodes in the same address level, resulting fewer number of address levels. On the other hand, smaller *bpl* value makes address length shorter, but fewer number of nodes can be allocated in the same address level requiring an increment in the number of address levels. It is hard to predict what value of *bpl* makes the overall address length minimum. Table 4.2 shows a computation of average address length for the associated scene to Figure 4.6. The computation results that average address length for scene (a) is 3.8, whereas it is 3.9 for scene (b). But if we use flat addressing without hierarchical levels, it would require 4 bits ($4 = \lceil \log_2(10) \rceil$) to address 10 nodes.

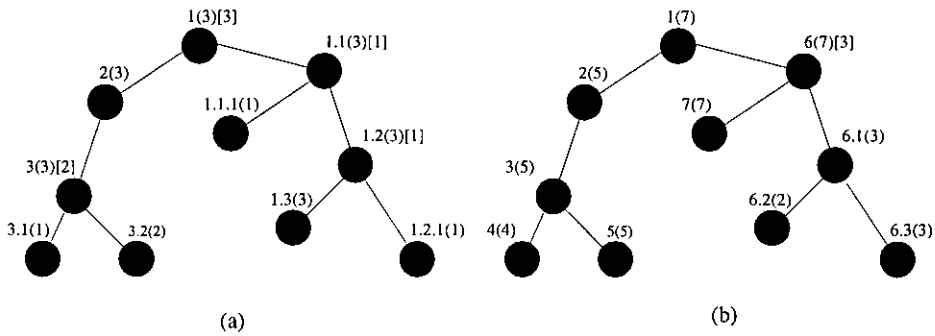


Figure 4.6: Address level scenario for 10 nodes. (a) *bpl* = 2 (b) *bpl* = 3.

Table 4.2: Average address length calculation for Figure 4.6.

Scene of Fig. 4.6 (a)		Scene of Fig. 4.6 (b)	
# of addr. level	# of Nodes	# of addr. level	# of Nodes
1	3	1	7
2	5	2	3
3	2	3	0
Total Address level = 19		Total Address level = 13	
Total Nodes = 10		Total Nodes = 10	
Avg. Addr. level = 1.9		Avg. Addr. level = 1.3	
Bits per level = 2		Bits per level = 3	
Avg. Address length = 3.8 bits		Avg. Address length = 3.9 bits	
Minimum Address length without levels = 4 bits			

Node distribution against depth/level In Figure 4.7, we show the depth/level wise distribution of 200 nodes for the scenario in Figure 4.1. Depth of a node counts the number of hops that the node is away from the sink, whereas level measures the number of address level in the address of a node. A node with higher depth value is located farther away from the sink than a node of lower depth value. Again, with the increase of address level, address gets longer. Figure 4.7 shows the number of nodes that belong to certain depth and level for bits per level (*bpl*) 3 and 5 for a network of size 200 nodes. It is seen that the distribution of nodes against the depth remains almost same in both cases when $bpl = 3$ and $bpl = 5$. Since nodes are randomly (uniformly distributed) scattered in the target area, it is expected that node density in the mid depth would be high, and we observe the same in Figure 4.7. The bar chart unveils that distribution of nodes against level is somewhat left skewed than that of against depth, and with the rise of value of *bpl* node distribution becomes more skewed to the left. It also shows that for larger *bpl* value more nodes get lower address level, resulting in a reduction in average address length. Figure 4.8 plots the distribution of nodes in respect of depth and level for various values of *bpl*.

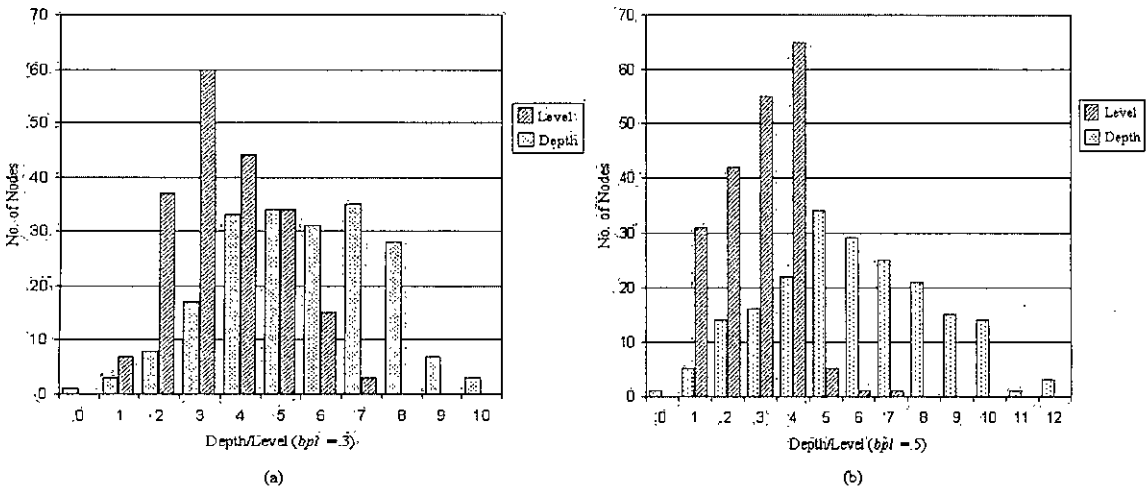


Figure 4.7: Depth vs. Level for 200 nodes. (a) $bpl = 3$ (b) $bpl = 5$.

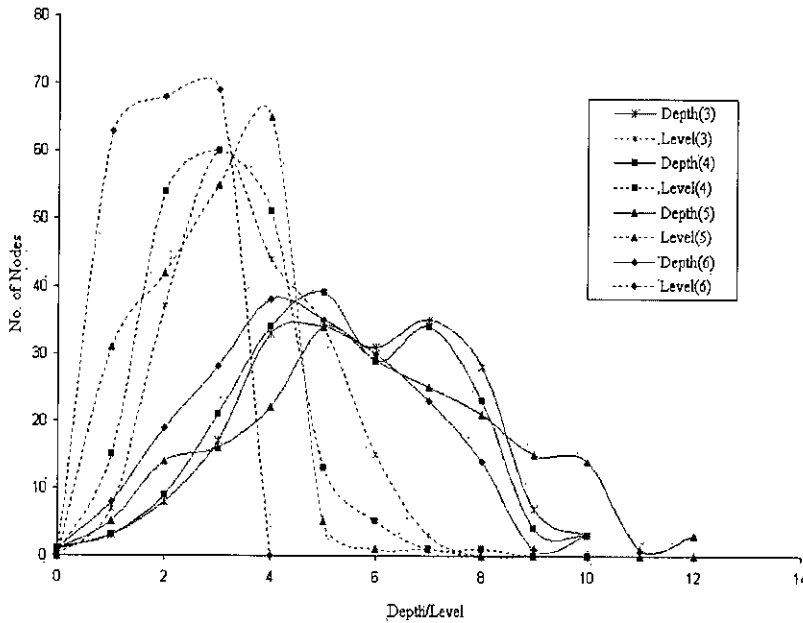


Figure 4.8: Distribution of nodes against depth and level. (In legend, number in parenthesis indicates the bpl value).

Impact of bits per level It is previously shown that bits per level bpl has critical impact on the average address length for a specific number of sensor nodes. Figure 4.9 demonstrates the simulation results on this regard for a network of size varying from 50 to 300. In Figure 4.10 the same impact is shown for a network size of 200 nodes. We list the following observations regarding the impact bpl on average address length :

- For smaller value of bpl , say 3 or 4, address length is quite high for any size of network.
- As bpl rises, address length gradually declines and at some point it gets the minimum value, then again goes up. The value at which address length obtains its minimum value is somewhat near to $\lceil \log_2(N) \rceil$ where N is the number of nodes in the network. A network of size N requires at least $\lceil \log_2(N) \rceil$ bits to address all nodes in a single address level. For example, network with 50 nodes achieves minimum address length at $bpl = 6$, whereas network with 300 nodes reaches the minimum at $bpl = 9$.

- When $bpl > \lceil \log_2(N) \rceil$, address length becomes larger and it then consistently climbs up (upward tail in Figure 4.9. In this case, hierarchical levels occur quite seldom and almost all addresses are in the very single level producing address length to be equal to bpl . Very few nodes may take multiple levels that makes address length slightly greater than bpl , as observed in Figure 4.10.
- Experiment shows that address with multiple level does not necessarily decrease the address length. Addressing requires minimum space if no level is created and address length is fixed by $\lceil \log_2(N) \rceil$.

Impact of network size Figure 4.11 portrays the average address length against the network size as well as the impact of bpl . It unveils that for a certain value of bpl , smaller network has shorter addresses than that of a larger network, and address length gradually rises against the network size.

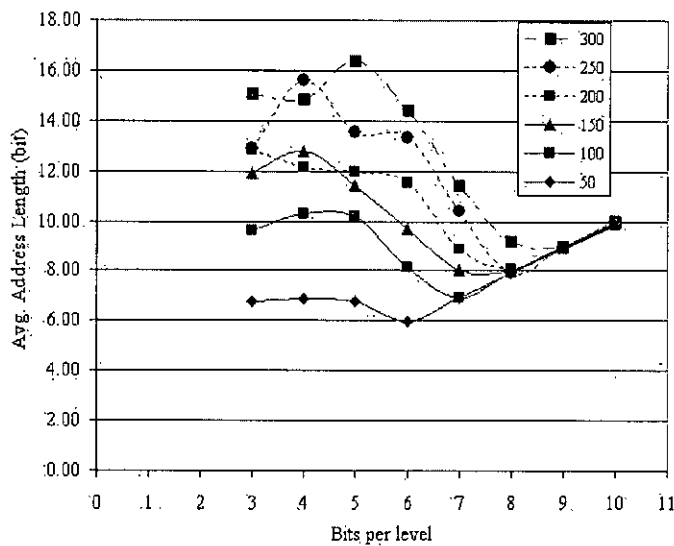


Figure 4.9: Impact of bits per level (bpl) on average address length on varying network size.

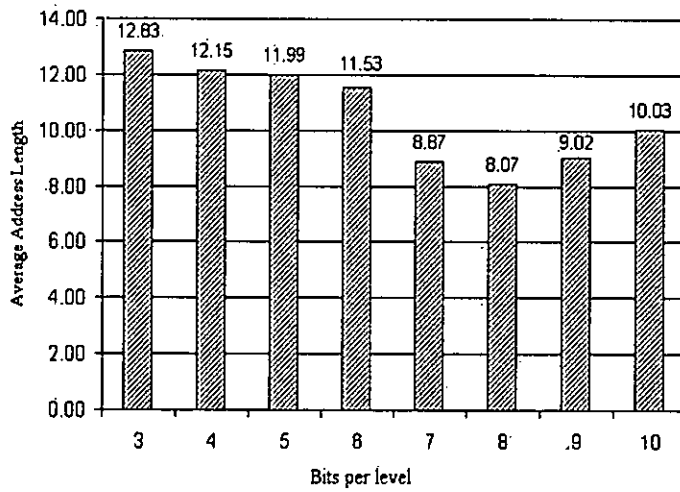


Figure 4.10: Impact of bits per level (*bpl*) on average address length for a network of size 200.

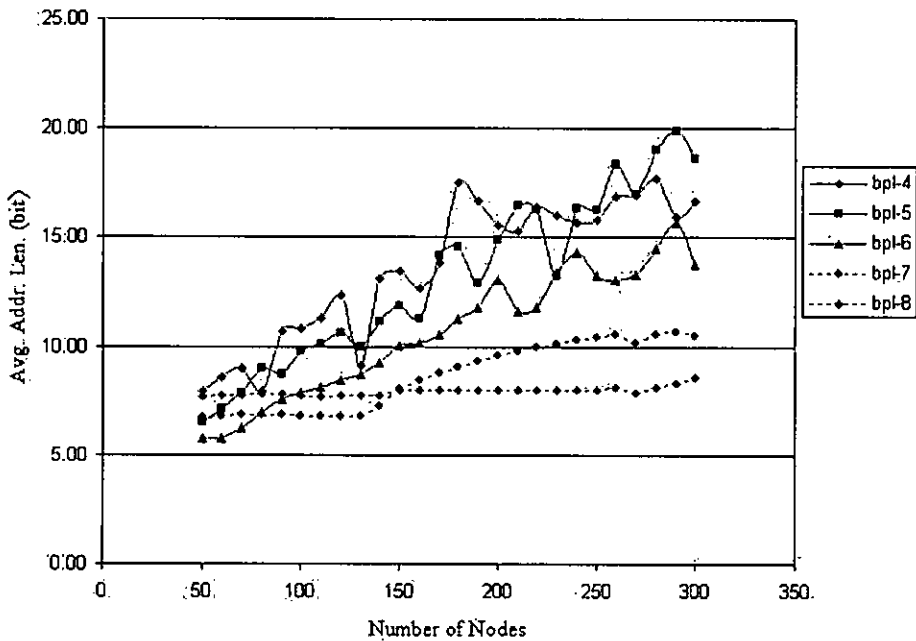


Figure 4.11: Impact of number of nodes on average address length.

4.2.4 Communication Energy

Whenever sensor nodes send their event data to the sink or the sink sends special control message to sensor nodes, it consumes energy. In our simulation we are interested to

measure the energy dissipation for carrying address in data packet. Since each data packet contains source or destination address, packet transmission has some address overhead in terms of energy consumption. We make the following assumptions in our simulation to analyze the communication energy:

- *Fixed data length.* We assume that all data packets (query, request or response), contains same length of data packet. In our simulation we assume it to be 16 bits. The choice is due to the fact that sensors' reading or event data are normally small. In fact any size of data length can be assumed, but in order to trace the impact the address overhead precisely (to compare the observation against *TreeCast* as well), we consider constant data length rather than choosing it randomly.
- *Packet formation.* The packets are formed with the fields as shown in Figure 4.12. We assume depth and number of levels are 4-bits numbers allowing a total of 16 depth values and 15 address levels which are quite sufficient for a network smaller than 1000 nodes deployed in 1000m×1000m. In our experiments, we haven't got any depth/level greater than 16. For wide area deployment, depth field can be few bits longer than level.

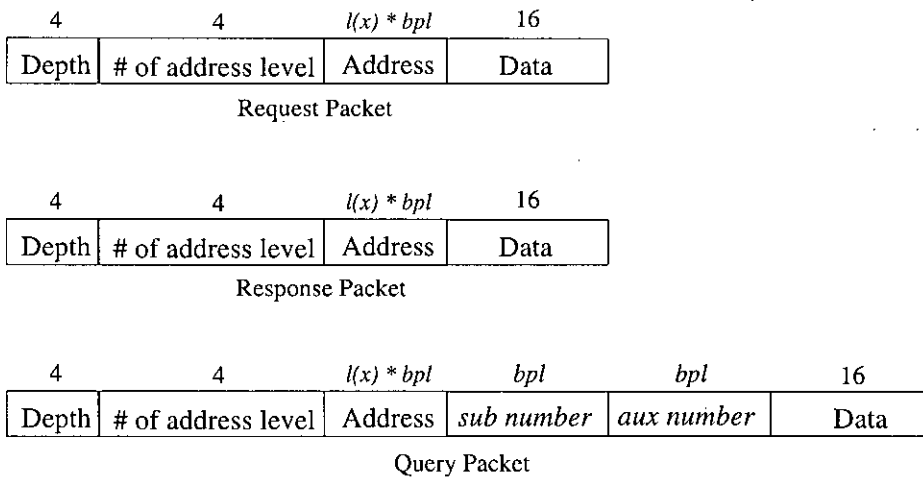


Figure 4.12: Packet fields of HN-Addressing (field lengths are measured in bits).

Communication cost

From the earlier chapter, we know that when a request packet is issued to a specific sensor by the sink or a response packet to sink by a sensor, the packet contains the address of associated node. In former case, packet contains the target address and in later, it contains the source address. If we consider packet with fixed length data (say, 16 bits), energy consumption may be impacted by the address length in packet. Again when a packet is destined to the sink or to any specific node, the packet traverses several hops to reach the destination. As a result the depth of a node has also some effects on total energy consumption for data communication.

We define a quantity, average communication cost for request and response packet as follows:

$$\text{Average Communication Cost (for request or response)} = \frac{1}{N} \sum_x l(x) \times d(x) \times bpl \text{ bits}$$

where $l(x)$ is the number of address level of node x , $d(x)$ is the depth of the node x and N is the number of total sensor nodes. Average communication cost measures the average transmission of address bits in the network, if each sensor sends or receives packet exactly once. Since every bit transmission takes nearly same amount of energy, this measurement closely estimates the total communication energy due to address transmission.

But in case of query packet, the communication cost is little different, because while forwarding a query packet to all nodes in the network each sensor puts its own address along with its *subordinate number* and *auxiliary number* in the packet. So in this case, communication cost for query packet can be computed as,

$$\text{Communication Cost (for query)} = \sum_x (l(x) + 2) \times bpl \text{ bits}$$

Table 4.3 shows a computation of average communication cost for the scenario in Figure 4.6.

In the simulation we select randomly a active node and make it to send a response packet to to the sink. Since content of both request and response packet is same and they

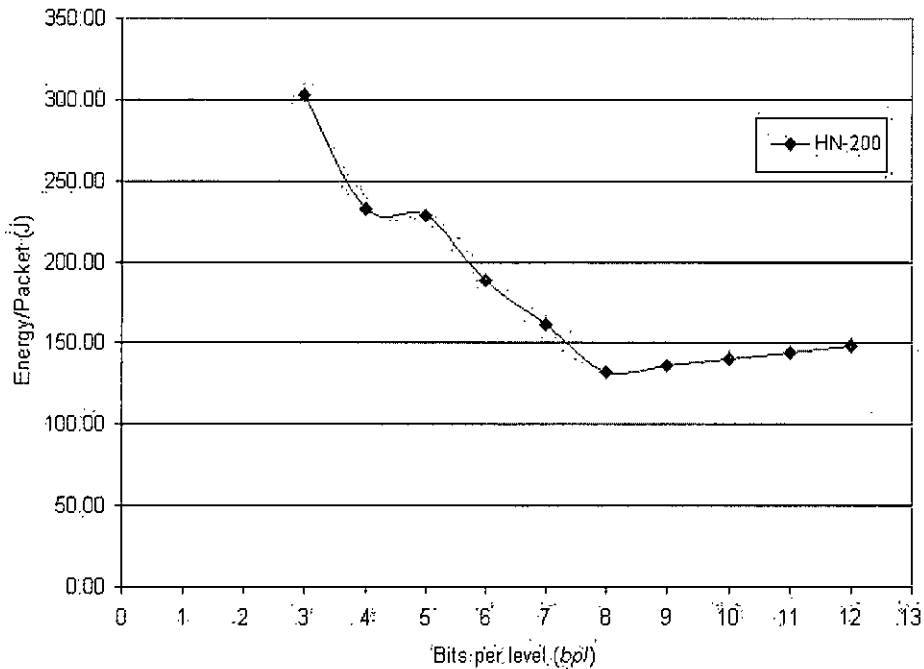


Figure 4.14: Energy consumption of packet routing in HN-Addressing for a network of size 200.

in Figure 4.13:

- Energy per packet is declined with the increase of bits per level (bpl). This is due to the fact that as the value of bpl rises, average address length of nodes gradually falls which causes the reduction of energy/packet.
- Energy/packet gradually falls against the rise of bpl upto a certain value of bpl , but afterward energy consumption again goes up. It can be predicted that the point after which the rise of bpl results in a linear increase of energy/packet, is fairly around $\lceil \log_2(N) \rceil$. $\lceil \log_2(N) \rceil$ is the minimum bit requirement for addressing a network of size N . For the value of bpl greater than $\lceil \log_2(N) \rceil$, average address length gradually rises resulting more energy dissipation per packet transmission. We find in Figure 4.14 that for 200 nodes we get the minimum energy/packet at $bpl = 8 = \lceil \log_2(200) \rceil$ and after that point, energy/packet again escalates.
- Network size has some, though little and somewhat irregular, impact on energy/packet.

Table 4.3: Calculation of average communication cost for the scenario at Figure 4.6.

Scene of Fig. 4.6 (a)	Scene of Fig. 4.6 (b)
$\sum_x l(x)d(x) = 54$	$\sum_x l(x)d(x) = 29$
$\sum_x (l(x) + 2) = 39$	$\sum_x l(x) + 2 = 33$
Total Nodes = 10	Total Nodes = 10
Bits per level = 2	Bits per level = 3
Communication Cost	
Req/Res = 10.8 bits	Req/Res = 8.7 bits
Qry = 78 bits	Qry = 99 bits

follow the same route to the target, we consider only response packet. We generate 100 such events for a specific sensor deployment and get an average over them. To get the final average value of energy, we take average of 5 observations from 5 different scenes. The simulation output is shown in Figure 4.13.

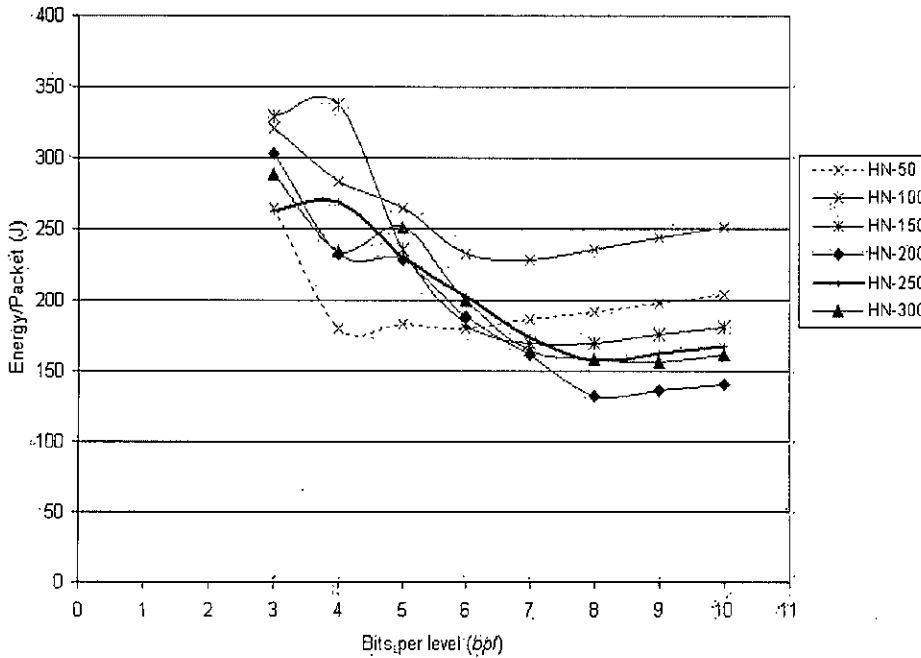


Figure 4.13: Energy consumption of packet routing in HN-Addressing with varying network size (b) for network of size 200.

We can list the following observations regarding energy consumption curve as shown

Major contributions in communication energy is due to address length and depth of nodes. As nodes are uniformly deployed in the area, distribution of nodes against depth may remain quite similar for varying network size. But average address length of nodes becomes slightly longer for larger network. *bpl* plays a intricate impact on energy/packet against network size. Smaller network may not be able to make complete utilization of address bits at all levels, whereas larger network can make better usage of address bits by optimal selection of *bpl*.

4.3 Performance on Node Failure

HN-Addressing uses address reallocation approach in case of node failure. When a node fails and consequently some nodes become orphan, we conduct an address reallocation procedure that is initiated by the address solicitation messages from the affected node. In this course tree structure is rebuilt and orphan nodes are assigned with newer addresses, this time with additional address levels than their previous assignment. So, it signs a rise of average address length owing to node failure.

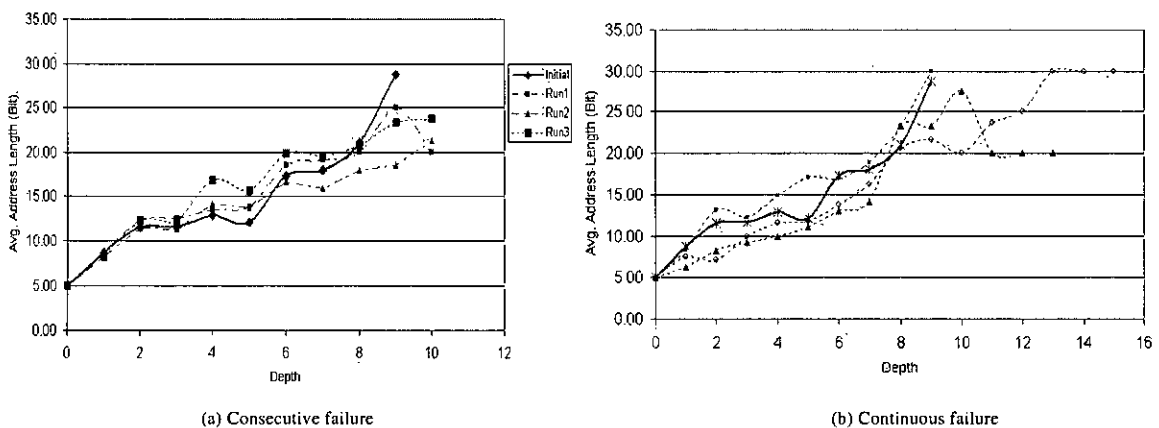


Figure 4.15: Average address length against node depth in case of node failure.

We consider two types of node failure models in our simulation: continuous and consecutive failure. Simulation observations are presented in Figure 4.15 for both continuous

and consecutive failure of nodes. In each failure model, we make 20% nodes to fail randomly one after another keeping a random time gap between two successive failures. In Figure 4.15, we show the effects for three independent simulation runs on the same network (200 nodes). The curves indicate that address length does not deteriorate very much in case of node failure and address length roams around some initial address curve. Again, in case of consecutive failure address length is slightly worse than continuous failure.

4.4 Comparison with *TreeCast*

In this section, we show the results of our comparative study between our proposed addressing and routing approach with *TreeCast* on the basis on several performance metrics. We consider the following comparison metrics:

1. Message cost for addressing
2. Average address length
3. Communication energy
4. Address allocation time

4.4.1 Message Cost for Addressing

Both HN-Addressing and *TreeCast* use distributed message passing to allocate address to nodes in the network. *TreeCast* uses four types of messages namely *ALLOCATE*, *APPROVE*, *COMPLAINT* and *CONFIRM* in its address allocation phase. HN-Addressing also passes four types of messages like *ALERT*, *READY*, *NUMBER* and *DONE* in its addressing process. HN-Addressing also passes two other messages *ASSIGN-ADDRESS* and *DISCARD-ADDRESS* to tackle the failure of nodes in the middle of address allocation process. Figure 4.16 presents the simulation output of initial message cost for addressing

in both schemes. As Figure reveals HN-Addressing takes slightly more messages than *TreeCast* keeping the trend of the number of messages against network size exactly the same. In both cases, message count is approximately four times of the network size.

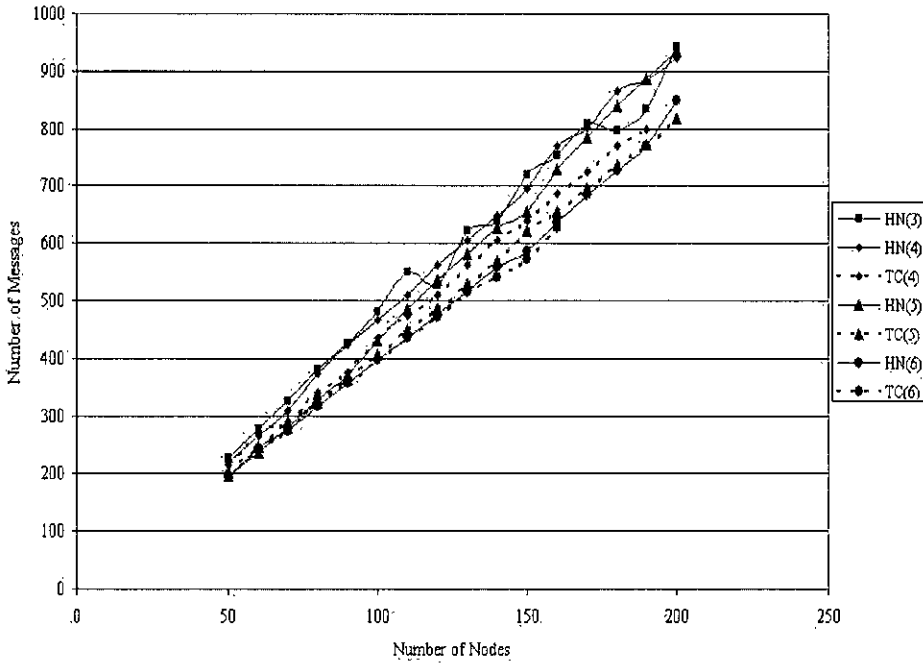


Figure 4.16: Initial message cost for address allocation in HN-Addressing and *TreeCast*.

4.4.2 Average Address Length

Average address length is one of the most important comparison metric. Unlike *TreeCast*, HN-Addressing does not increase address level at every depth of the tree, rather it utilizes all addresses in certain address level bounded by the bits per level (*bpl*) and switches to the next address level when all addresses in certain level are exhausted. This makes better utilization of address bits in HN-Addressing and average address length of nodes is smaller than that of *TreeCast*. Table 4.4 shows the computation of average address length for the scenario of Figure 4.17.

In respect of address length, HN-Addressing outperforms *TreeCast* by a big margin. Figure 4.18 shows the simulation result for 200 nodes in this regard. As the bar chart

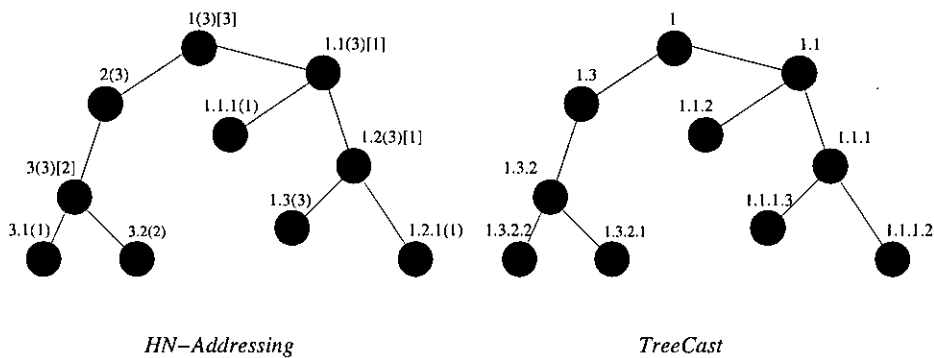


Figure 4.17: Address allocation scenario for 10 nodes by HN-Addressing and *TreeCast*.

Table 4.4: Average address length calculation for Figure 4.17

HN-Addressing		TreeCast	
# of addr. level	# of Nodes	# of addr. level	# of Nodes
1	3	1	1
2	5	2	2
3	2	3	2
4	0	4	4
Total Address level = 19		Total Address level = 27	
Total Nodes = 10		Total Nodes = 10	
Avg. Addr. level = 1.9		Avg. Addr. level = 2.7	
Bits per level = 2		Bits per level = 2	
Avg. Address length = 3.8 bits		Avg. Address length = 5.4 bits	

indicates, with the increase of bits per level, HN-Addressing gradually reduces the address bits whereas *TreeCast* grows it up. In Figure 4.18, we show the average address length for 200 nodes by varying *bpl* from 3 to 7. No *TreeCast* observation has been found for *bpl* value 3, because by setting *bpl* value to 3, we limit the number of children of a parent node to be at most 7 ($= 2^3 - 1$). As we are running our simulation for node density 10, there may be some nodes that cannot be assigned addresses from their parents due to run out of possible all addresses. By setting *bpl* value to 3, we have not got a network of size 200 that can be assigned with addresses to all of its nodes. So, we left the bar empty for *TreeCast* in the case of *bpl* = 3. It is also observed that minimum address length of

TreeCast is higher than HN-Addressing.

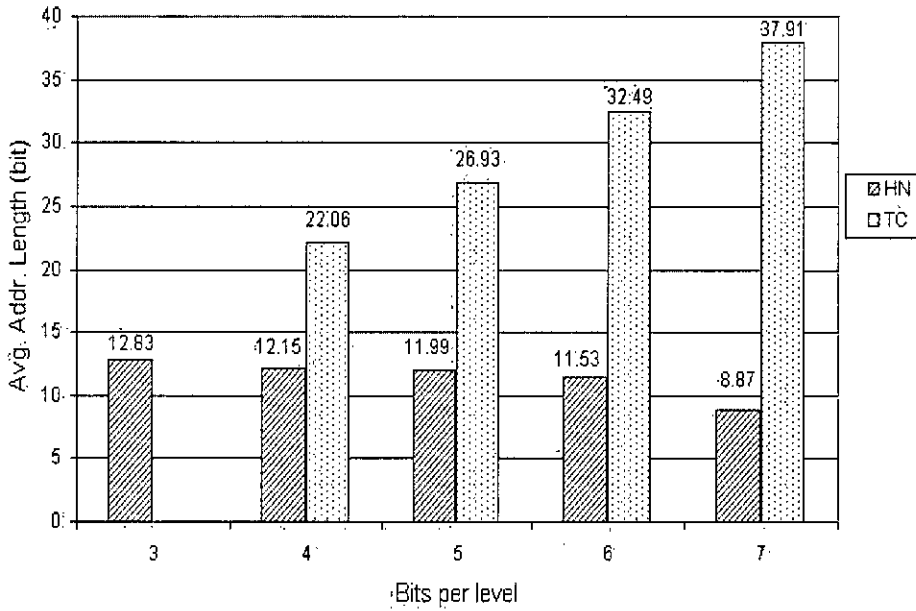


Figure 4.18: Average address length in HN-Addressing and *TreeCast*.

4.4.3 Communication Energy

We compare our proposed routing technique against *TreeCast* in terms of energy consumptions in routing events. We assume that data packet contains target or source address and a fixed data of size 16 bits. Since data field is fixed, address length plays a dominating role in energy dissipation by the sensors in packet sending and receiving. In Figure 4.19, we show the average energy consumption per packet for request packet from the sink or response packet to the sink. The bar chart demonstrates that in terms of energy per packet HN-Addressing is far better than *TreeCast*. This is due to the following two facts:

- HN-Addressing keeps address shorter and hence reduces the address overhead in the data packet resulting a reduction of communication energy. But in case of *TreeCast*, address length is much longer than HN-Addressing which makes energy consumption per packet greater.

- What makes *TreeCast* worse is that in *TreeCast* with the increase of depth (or level), node address gets longer. A node far away from the sink with a larger depth value is assigned an address with an address level equal to its depth which makes the address length pretty longer. So long addresses are to traverse long way to the sink and vice versa. This contributes a double impact on energy/packet (longer address passes many hops) that makes the communication energy quite large. And with the increase of *bpl*, *TreeCast* rises address length and accordingly the energy/packet in the right way.

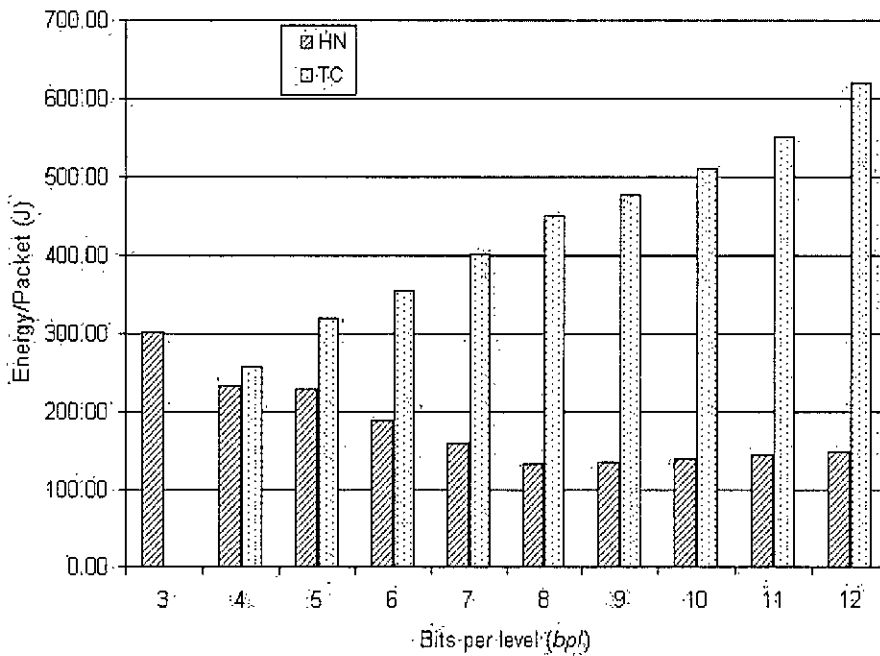


Figure 4.19: Energy/packet for HN-Addressing and *TreeCast*.

4.4.4 Address Allocation Time

This metric measures how much time the addressing protocol takes to assign address to all nodes in the network. The simulation result is shown in Figure 4.20. It depicts that HN-Addressing has clear defeat by *TreeCast* in this metric. *TreeCast* assigns addresses to the whole network quite faster than HN-Addressing. *TreeCast* takes almost logarithmic

order of time on the size of network (number of nodes) to address whole network, whereas the HN-Addressing takes linear time.

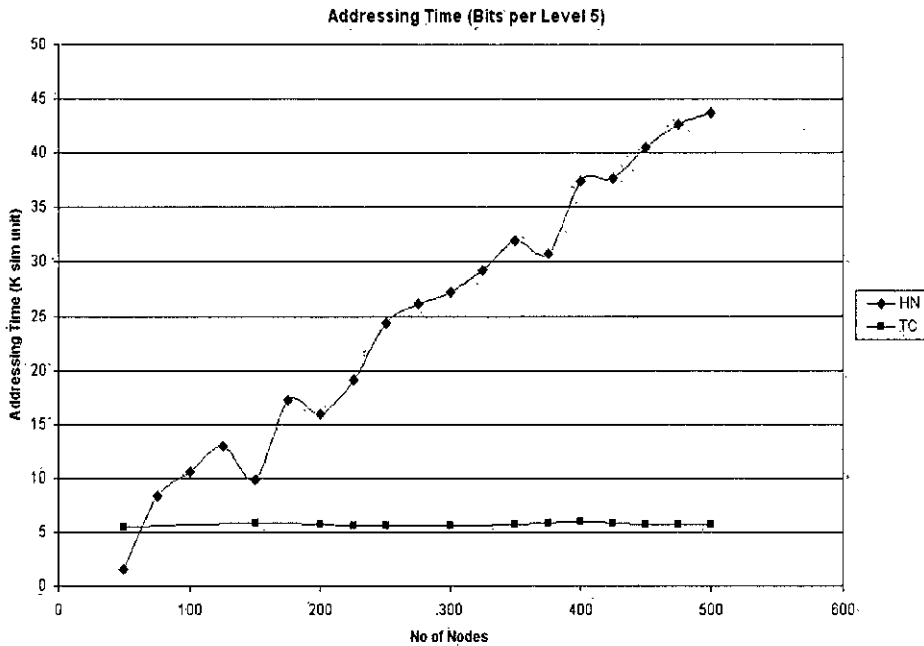


Figure 4.20: Address allocation time for a network of size 200 with $bpl = 5$.

More time requirement of HN-Addressing to allocate address to all nodes is due to the fact that HN-Addressing assigns address to each node individually, one node at a time. No two nodes in the network are assigned address at the same time. Nodes are assigned address sequentially one after another. But in *TreeCast*, several nodes can actively participate in the address allocation process at the same time. When a node is confirmed with its local address, it can safely go for addressing the nodes in its subtree. So address allocation in subtrees can go in parallel, whereas in case of HN-Addressing, only after finishing address allocation to all nodes in a subtree, a parent node can go for another child node or subtree. No two subtrees can be active in address allocation process simultaneously, so no parallel allocation process is possible. And this is why the address allocation time is large for HN-Addressing. *TreeCast* takes allocation time in the order of depth of the tree, whereas HN-Addressing takes time in the order of number of nodes. Figure 4.21 illustrates the scenario.

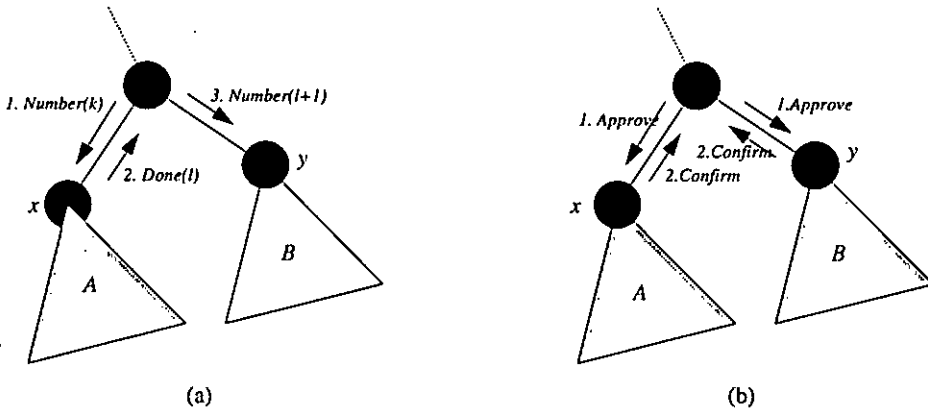


Figure 4.21: Address allocation. (a) HN-Addressing: address allocation is going on in subtree *A* under *x*, until node *x* returns *DONE(l)* message to its parent, no node in subtree *B* under node *y* can have address, (b) *TreeCast*: if both *x* and *y* are approved with their addresses, both subtree *A* and *B* can be active in address allocation.

4.5 Summary

In this chapter, we produce the simulation results in support of performance of our proposed addressing and routing scheme for sensor networks. We analyze various performance metrics against some defined system parameters. We also simulate another similar approach *TreeCast* and compare our approach against it. In comparison between HN-Addressing and *TreeCast*, we can draw the following summary table:

Table 4.5: Summary of comparison between HN-Addressing and *TreeCast*.

Parameters	HN-Addressing	TreeCast
Message count	Nearly $4 \times n$	Nearly $4 \times n$
Address length	Good (logarithmic)	Poor (Linear)
Data routing	Yes	Yes
Packet overhead	Small	Large
Handling node failure	Adjust address	Alternate Route
Communication energy	Better	Worse
Addressing Time	Poor (Linear)	Good (Logarithmic)

Chapter 5

Conclusion

In this last chapter, we draw the conclusion of our thesis by describing the major contributions made by the research works associated with the thesis followed by some directions for future research over the issue.

5.1 Major Contributions

The contributions that have been made in this thesis can be enumerated as follows:

- In this thesis, we focus on addressing and routing architecture for sensor networks. To our best knowledge, this topic on sensor networks has been examined little so far. If we haven't explored the literature poorly, we can say that our task is the second of this type. The first architecture is *TreeCast* [67].
- We design a dynamic and distributed global address assignment technique for sensor networks, HN-Addressing. In this regard, we propose a mechanism to assign distinct address to each individual sensor in the network so that each sensor can be uniquely identified by the sink. The addressing scheme devised here is self-configurable and can endure the common constraints of sensor networks like energy shortage, multihop communication, node failure and so on.

- Not only the addressing, in our work we build a routing paradigm based on our address assignment technique such that nodes can take routing decisions whenever routing data packets to destination depending on the assigned addresses. It enables data routing that requires neither routing table nor the earlier routing history. In this sense, we actually develop a *stateless* routing technique for sensor networks.
- We formulate a theoretical study of our approach and produce a formal verification of the underlying routing principles in the architecture.
- We not only develop the protocol for addressing and routing, but also simulate the protocol by Parsec to make closer observations into the protocol. We make a rigorous simulation based study of various performance issues of the proposed approach, and analyze the simulation output against the expected behavior. The performance of the scheme in terms of message cost, address length, communication energy and others are analyzed.
- We also make a simulation of our counter scheme, *TreeCast*, to compare our approach with it. Simulation reveals that HN-Addressing outperforms *TreeCast* with a very sharp margin in various important aspects like average address length and communication energy, that are considered to be very critical metrics for energy-efficient operation of sensor networks.
- And at last, If we place two *stateless* routing approaches head to head, we can have the concluding remarks as follows. *TreeCast* uses hierarchical leveling in assigning addresses to nodes along to the depth of the network tree. Our observation finds that this makes the address length quite longer. So we make an alternate approach. As numbering policy, we use a graph theoretic concept of pre-order traversal of the network tree. We combine the pre-order leveling of nodes along with the multi-level hierarchy in our address assignment technique. As the simulation ascertains, our technique makes the address length quite shorter and makes data routing more energy preserving.

5.2 Future Research Directions

No research work can ever end. Any research on any topic always makes a way to further research on the way. Ours is not an exception. Since address based *stateless* routing has been explored little, more research can be possible on this topic. Based on our current design and the results of simulations presented in this thesis, we can investigate the extension of our works in future in the following directions:

1. The message cost for initial addressing of the whole network can be reduced. Although the initial message cost for addressing nodes has little impact on the long run operation of the network because of the zero-mobility of sensor nodes requiring rare reallocation of addresses (due to failure, wake up from sleep) once after the network has been addresses completely, still there may be some scopes to reducing the number of messages. In our rough guess, we can try the address allocation with three messages instead of currently proposed four messages leaving the *ALERT* message altogether. *ALERT* message is used to select parent node out of the neighbors. This parent node later assigns address to the child node by a *NUMBER* message. Instead of waiting for an *ALERT* message, a node may rather overhear the *DONE* message in its neighborhood to select its parent. Further investigation is required to determine whether the technique could be applicable or not.
2. Due to severe energy constraint, while designing any protocol for sensor network, everyone always try to save energy, whatever small the quantity of saving may be. As we show in a computation in the earlier chapter, energy consumption for reporting data packets to the sink, or request from the sink or network wide flooding depends on the address length and depth of nodes (since data field is assumed to be of constant length). To require less energy consumption in packet communication, it is desirable that longer packets traverse shorter paths and shorter packets traverse longer paths to reach the sink or destination. Further investigation can be engaged to discover an approach that can make this possible as suggested in the Figure 5.1.

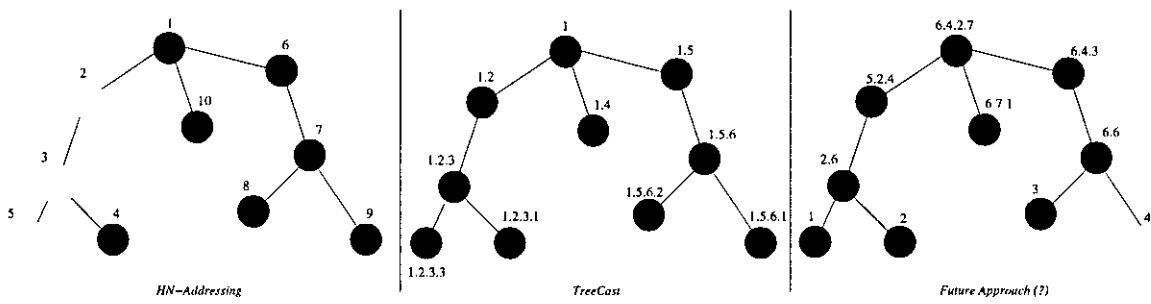


Figure 5.1: Hypothetical addressing approach with greater energy saving. (a) Currently proposed HN-Addressing without leveling, (b) *TreeCast*, (c) Future approach.

3. HN-Addressing suffers a lot in respect of address allocation time. Since each sensor collects address in a well ordered sequence, it takes long time to address the entire network. Measures can be thought to be taken so that multiple subtrees can remain active in address allocation at the same time, thus boosting up the address allocation process. One possible measure can be that instead of assigning a single number to a node, the parent can allocate a child node with a set of numbers to assign number to nodes in the subtree of the child node. In that case multiple subtrees under a parent can be made active. The number of numbers that would be used the parent node
4. Location based routing has an immense application in sensor application. In this routing, a set of sensors in a certain geographic location are to be instructed for data reporting to the sink. In this case, sink does not specify any specific node, rather advertises some attributes to identify the location. Sensors residing in the target region send their observed data to the sink upon receipt of the advertisement. This may lead to a further study to determine whether the proposed technique can be adapted for geographic routing. In current scheme nodes have hierarchical leveling on a tree structure, so certain group of sensors in a subtree can be targeted by their parent node, and it is quite likely that nodes in a subtree may be located in a close geographic vicinity.
5. Data aggregation is another hot research issue for sensor networks. When an event is

occurred in the environment and several nodes close the event spot, after detecting the event, sends the their observation to the sink, multiple copies of the same event from multiple sources propagate in the network. Data aggregation addresses this redundant packet transmission problem. Aggregation restricts duplicate transmission and sometimes fuse data into a single packet obtained from multiple sources. The technique mostly incorporates with routing decision of packet forwarding. It can be an excellent future research endeavor to investigate whether the data aggregation can be incorporated with our current technique. Since nodes have address in a tree like formation, there can be some clue to detect multiple source sending same events.

6. Clustering is a means of grouping of sensor nodes into some disjoint sets, each set being a cluster, such that each sensor belongs to any of the clusters and no sensors be a member of more than one cluster. Nodes in the same cluster maintain some sort of neighborhood among themselves. In a cluster, there would be a cluster-head, that is single hop away from other members of this cluster. Further study can be directed to investigate the possibility of existence of any underlying clustering paradigm based on current addressing approach.

Bibliography

- [1] I. F. Akyildiz, W. S. Sankaararubramaniam and E. Cayirci, "Wireless Sensor Networks: a Survey", *Computer Networks*, vol. 38, pp. 393-422, 2002.
- [2] I. F. Akyildiz and W. Su, "A Power Aware Enhanced Routing (PAER) Protocol for Sensor Networks", *Georgia Tech Technical Report*, January 2002.
- [3] M. Ali and Z. A. Uzmi, "An Energy-Efficient Node Address Naming Scheme for Wireless Sensor Networks", *IEEE International Networking and Communication Conference INCC'04*, 2004.
- [4] J. N. Al-Karaki and A. E. Kamal, "Routing Techniques in Wireless Sensor Networks: A Survey", *IEEE Wireless Communications*, vol. 11 (6), pp. 6-28, December 2004.
- [5] H. Balakrishna, V. Padmanabhan, S. Seshan and R. Katz, "A Comparison of Mechanisms for Improving TCP Performance over Wireless Links", *IEEE/ACM Transactions on Networking*, vol. 5, 1997.
- [6] R. Bagrodia, R. Meyerr *et al.*, "PARSEC: A Parallel Simulation Environment for Complex System", *UCLA Technical Report*, 1997.
- [7] L. Bajaj, M. Takai, R. Ahuja, K. Tang, R. Bagrodia and M. Gerla, "Glomosim: A Scalable Network Simulation Environment", Technical Report 990027, UCLA Computer Science Department, May 1999.
- [8] S. Bandyopadhyay and E. J. Coyle, "An Energy Efficient Hierarchical Clustering Algorithm for Wireless Sensor Networks", *Proc. of INFOCOM 2003: Twenty-Second Annual Joint*

- Conference of the IEEE Computer and Communications Societies*, vol. 3, pp. 1713-1723, March 2003.
- [9] A. Bakre and B. R. Badrinath, "I-TCP: Indirect TCP for Mobile Hosts", *Proc. of the 15th International Conference on Distributed Computing Systems*, pp. 136-143, Vancouver, BC, May 1995.
- [10] V. Bharghavan, "A Dynamic Addressing Scheme for Wireless Media Access", *IEEE International Conference on Communication*, pp. 756-760, Seattle, WA, 1995.
- [11] <http://www.bluetooth.com>.
- [12] P. Bonnet, J. Gehrke and P. Seshadri, "Querying the Physical World", *IEEE Personal Communications*, pp. 10-15, October 2000.
- [13] D. Bragnisky and D. Estrin, "Rumor Routing Algorithm for Sensor Networks", *Proc. of the First Workshop on Sensor Networks and Applications (WSNA)*, Atlanta, GA, October 2002.
- [14] N. Bulusu, D. Estrin, L. Girod and J. Heidemann, "Scalable Coordination for Wireless Sensor Networks: Self-configuring Localization Systems", *International Symposium on Communication Theory and Applications (ISCTA2001)*, Ambleside, UK, July 2001.
- [15] C. Casetti, M. Gerla, S. Mascolo, M. Y. Sanadidi and R. Wang, "TCP Westwood: End-to-End Bandwidth Estimation for Enhanced Transport over Wireless Links", *Journal of Wireless Networks*, vol. 8, pp. 467-479, 2002.
- [16] A. Cerpa, J. Elson, D. Estrin, L. Girod, M. Hamilton and J. Zhao, "Habitat Monitoring: Application Driver for Wireless Communications Technology", *Proc. of 2001 ACM SIGCOMM Workshop on Data Communications in Latin America and the Caribbean*, San Jose, Costa Rica, April 2001.
- [17] U. Cetintemel, A. Flinders and Y. Sun, "Power-Efficient Data Dissemination in Wireless Sensor Networks", *Proc. of the 3rd ACM international workshop on Data Engineering for Wireless and Mobile Access*, pp. 1-8, San Diego, CA, 2003.

- [18] R. S. Chang and C. J. Kuo, "An Energy Efficient Routing Mechanism for Wireless Sensor Networks", *Proc. of the 20th International Conference on Advanced Information Networking and Applications (AINA06)*, vol. 2, pp. 308-312, April 2006.
- [19] K. Charkrabarty, S. Iyengar, H. Qi and E. Cho, "Grid Coverage for Surveillance and Target Location in Distributed Sensor Networks", *IEEE Transaction on Computers*, 2002.
- [20] B. Chen, K. Jamieson, H. Balakrishnan and R. Morris, "SPAN: an Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks", *Wireless Networks*, vol. 8(5), pp. 481- 494, September 2002.
- [21] C. Chien, I. Elgorriaga and C. McConaghy, "Low-power Direct Sequence Spread-Spectrum Modem Architecture for Distributed Wireless Sensor Networks", *Proceedings of the 2001 international symposium on Low power electronics and design*, pp. 251-154, Huntington Beach, CA, August 2001.
- [22] R. W. Clay, N. R. Wild, D. J. Bird, B. R. Dawson, M. Johnston, R. Patrick and A. Sewell, "A Cloud Monitoring System for Remote Sites", *Publications of the Astronomical Society of Australia*, vol. 15(3), pp. 332-335, August 1998.
- [23] R. J. Cramer, M. Z. Win and R. A. Scholtz, "Impulse Radio Multipath Characteristics and Diversity Reception", *IEEE International Conference on Communications ICC98*, vol. 3, pp. 1650-1654, 1998.
- [24] DSN Team, Multilateration Poster, *SensIT Workshop*, St.Petersburg, FL, April 2001.
- [25] A. Dunkels, J. Alonso and T. Voigt, "Making TCP/IP Viable for Wireless Sensor Networks", *Proc. of the First European Workshop on Wireless Sensor Networks (EWSN)*, 2004.
- [26] J. Elson and D. Estrin, "Random, Ephemeral Transaction Identifiers in Dynamic Sensor Networks", *Proc. of 21st International Conference on Distributed Computing Systems*, pp. 459-468, Mesa, AZ, April 2001.
- [27] D. Estrin, R. Govindan, J. Heidemann and S. Kumar, "Next Century Challenges: Scalable Coordination in Sensor Networks", *Proc. of 5th ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom99)*, pp. 263-270, Seattle, WA, 1999.

- [28] D. Estrin, L. Girod, G. Pottie and M. Srivastava, "Instrumenting the World with Wireless Sensor Networks", *International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2001)*, Salt Lake City, Utah, May 2001.
- [29] Q. Fang, J. Leonidas, J. Guibas, V. Silva and L. Zhang, "GLIDER: Gradient Landmark-Based Distributed Routing for Sensor Networks", Source.
- [30] C. Guo, L. C. Zhong and J. M. Rabaey, "Low Power Distributed MAC for Ad Hoc Sensor Radio Networks", *IEEE Proc. GlobeCom*, pp. 2944-2948, 2001.
- [31] H. Gupta, S. Das and Q. Gu, "Connected Sensor Cover: Self-organization of Sensor Networks for Efficient Query Execution", *Proc. of the International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'03)*, 2003.
- [32] K. Han, Y. Ko and J. Kim, "A Novel Gradient Approach for Efficient Data Dissemination in Wireless Sensor Networks", *Proc. of the IEEE 60th Vehicular Technology Conference*, vol. 4, pp. 2979-2983, September 2004.
- [33] T. He, J. A. Stankovic, C. Lu and T. Abdelzaher, "SPEED: A Real-Time Routing Protocol for Sensor Networks", *Proc. of 23rd Conference on Distributed Computing Systems*, 2003.
- [34] S. Hedetniemi and A. Liestman, "A Survey of Gossiping and Broadcasting in Communication Networks", *Networks*, vol. 18(4), pp. 319-349, 1988.
- [35] J. Heidemann, F. Silva, C. Intanagonwiwat, R. Govindan, D. Estrin and D. Ganesan, "Building Efficient Wireless Sensor Networks with Low-Level Naming", *Proc. of Symposium on Operating Systems Principles*, ACM Press, 2001.
- [36] W. Heinzelman, J. Kulik and H. Balakrishnan, "Adaptive Protocols for Information Dissemination in Wireless Sensor Networks", *Proc. of 5th ACM/IEEE MobiCom Conference (MobiCom '99)*, Seattle, WA, pp. 174-185, August 1999.
- [37] W. Heinzelman, A. Chandrakasan and H. Balakrishnan, "Energy-efficient Communication Protocols for Wireless Microsensor Networks", *Proc. of the 33rd Hawaii International Conference on System Sciences (HICSS '00)*, pp. 2-10, January 2000.

- [38] W. Heinzelman, "Application-Specific Protocols Architectures for Wireless Networks", *PhD Thesis*, MIT, June 2000.
- [39] W. Heinzelman, A. Chandrakasan and H. Balakrishnan, "An Application-Specific Protocol Architecture for Wireless Microsensor Networks", *IEEE Transactions on Wireless Communications*, vol. 1(4), October 2002.
- [40] C. Herring and S. Kaplan, "Component-based Software Systems for Smart Environments", *IEEE Personal Communications*, pp. 60-61, October 2000.
- [41] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler and K. Pister, "System Architecture Directions for Networked Sensors", *Proc. of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems*, 2002.
- [42] G. Hoblos, M. Staroswiecki and A. Aitouche, "Optimal Design of Fault Tolerant Sensor Networks", *IEEE International Conference on Control Applications*, pp. 467-472, Anchorage, AK, September 2000.
- [43] <http://www.homerf.com/hrfwgtec.pdf>.
- [44] V. S. Hsu, J. M. Kahn and K. S. J. Pister, "Wireless Communications for Smart Dust", *Electronics Research Laboratory Memorandum Number M98/2*, UC Berkley, January 1998.
- [45] C. Intanagonwiwat, R. Govindan and D. Estrin, "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks", *Proc. of ACM MobiCom '00*, Boston, MA, pp. 56-57, 2000.
- [46] J. M. Kahn, R. H. Randy and K. S. J. Pister, "Next Century Challenges: Mobile Networking for Smart Dust", *Proc. of the ACM MobiCom '99*, pp. 271-278, WA, 1999.
- [47] B. Karp and H. T. Kung, "GPSR: Greedy perimeter stateless routing for wireless sensor networks", *Proc. of the 6th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 00)*, Boston, MA, August 2000.

- [48] L. Subramanian and R. H. Katz, "An Architecture for Building Self Configurable Systems", in the *Proc. of IEEE/ACM Workshop on Mobile Ad Hoc Networking and Computing*, Boston, MA, August 2000.
- [49] T. H. Keitt, D. L. Urban and B. T. Milne, "Detecting Critical Scales in Fragmented Landscapes", *Conservation Ecology* 1(1)(1997), Available from <http://www.consecolo.org/vol1/iss1/art4>.
- [50] R. Kravets, K. Schwan and K. Calvert, "Power-aware Communication for Mobile Computers", *Proc. of MoMUC99*, pp. 64-73, San Diego, CA, November 1999.
- [51] F. Kuhn, R. Wattenhofer and A. Zollinger, "Worst-Case Optimal and Average-case Efficient Geometric Ad-hoc Routing", *Proc. of the 4th ACM International Conference on Mobile Computing and Networking*, pp. 267-278, 2003.
- [52] J. Kulik, W. R. Heinzelman and H. Balakrishnan, "Negotiation-based Protocols for Disseminating Information in Wireless Sensor Networks", *Wireless Networks*, vol. 8, pp. 169-185, 2002.
- [53] C. J. LeMartret and G. B. Giannakis, "All-digital Impulse Radio for MUI/ISI-Resilient Multiuser Communications over Frequency-Selective Multipath Channels", *Proc. of IEEE Military Communications Conference (MILCOM00)*, vol. 2, pp. 655-659, 2000.
- [54] H. Lee, B. Han, Y. Shin and S. Im, "Multipath Characteristics of Impulse Radio Channels", *Proc. of IEEE Vehicular Technology Conference*, vol. 3, pp. 2487-2491, Tokyo, 2000.
- [55] P. Levis and N. Lee, "TOSSIM: A Simulator for TinyOS Networks", *User's manual*, in *TinyOS documentation*, CS UC Berkley, September 2003.
- [56] C. R. Lin and M. Gerla, "Adaptive Clustering for Mobile Wireless Networks", *IEEE Journal on Computer Aided Design of High Performance Wireless Networked Systems*, vol. 15 (7), pp. 1265-1275, September 1997.
- [57] S. Lindsey and C. Raghavendra, "PEGASIS: Power-Efficient Gathering in Sensor Information Systems", *Proc. of IEEE Aerospace Conference*, vol. 3, pp. 1125 - 1130, 2002.

- [58] G. Lu, B. Krishnamachari and C. S. Raghavendra, "An Adaptive Energy-Efficient and Low-Latency MAC for Data Gathering in Wireless Sensor Networks", *Proc. of the 18th International Parallel and Distributed Processing Symposium (IPDPS04)*, pp. 1-8, 2004.
- [59] A. Manjeshwar and D. P. Agrawal, "TEEN: a Routing Protocol for Enhanced Efficiency in Wireless Sensor Networks", *Proc. of 1st International Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing*, April 2001.
- [60] A. Manjeshwar and D. P. Agrawal, "APTEEN: a Hybrid Protocol for Efficient Routing and Comprehensive Information Retrieval in Wireless Sensor Networks", *Proc. of IEEE International Conference on Parallel and Distributed Processing Symposium (IPDPS)*, pp. 195 - 202, 2002.
- [61] S. Meguerdichian and M. Potkonjak. "Low Power 0/1 Coverage and Scheduling Techniques in Sensor Networks", *UCLA Technical Reports 030001*, January 2003.
- [62] J. Mirkovic, G. P. Venkataramani, S. Lu, L. Zhang, "A Self-organizing Approach to Data Forwarding in Large Scale Sensor Networks", *IEEE International Conference on Communications ICC01*, Helsinki, Finland, June 2001.
- [63] T. Nandagopal, T. Kim, X. Gao and V. Bhargavan, "Achieving MAC Layer Fairness in Wireless Packet Networks", *Proc. of the ACM MobiCom'00*, Boston, MA, 2000.
- [64] Network simulator - NS-2, <http://www.isi.edu/nsnam/ns>.
- [65] N. Noury, T. Herve, V. Rialle, G. Virone, E. Mercier, G. Morey, A. Moro and T. Porcheron, "Monitoring Behavior in Home Using a Smart Fall Sensor", *IEEE-EMBS Special Topic Conference on Micro Technologies in Medicine and Biology*, pp.607610, October 2000.
- [66] E. Ould-Ahmed-Vall, D. M. Blough, B. S. Heck, and G. F. Riley, "Distributed Global Identification for Sensor Networks", *Technical Report 2005*, Georgia Tech, 2005.
- [67] S. PalChaudhuri, S. Du, A. K. Saha and D. B. Johnson, "TreeCast: A Stateless Addressing and Routing Architecture for Sensor Networks", *Proc. of the 18th IPDPS International Workshop on Algorithms for Wireless, Mobile, Ad Hoc and Sensor Networks (WMAN 2004)*, p. 221a, Santa Fe, New Mexico, April 2004.

- [68] S. PalChaudhuri, R. Kumar, R. G. Baraniuk and D. B. Johnson, "Design of Adaptive Overlays for Multi-scale Communication in Sensor Networks", *First IEEE International Conference Distributed Computing in Sensor Systems DCOSS 2005*, p. 173, Marina del Rey, CA, June 2005.
- [69] S. Park, A. Savvides and M. B. Srivastava, "SensorSim: A Simulation Framework for Sensor Networks", *Proc. of the ACM international Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pp. 104-111, Boston, MA, 2000.
- [70] A. Perrig, R. Szewczyk, V. Wen, D. Culler and J. D. Tygar, "SPINS: Security Protocols for Sensor Networks", *Proc. of ACM MobiCom'01*, pp. 189-199, Rome, Italy, 2001.
- [71] A. Porret, T. Melly, C. C. Enz and E. A. Vittoz, "A Low-power Low-voltage Transceiver Architecture Suitable for Wireless Distributed sensor networks", *IEEE International Symposium on Circuits and Systems'00*, vol. 1, pp. 56-59, Geneva, 2000.
- [72] G. J. Pottie and W. J. Kaiser, "Wireless Integrated Network Sensors", *Communications of the ACM*, vol. 43 (5), pp. 51-58, 2000.
- [73] H. Qi, X. Wang, S. S. Iyengar, and K. Chakrabarty, "Multisensor Data Fusion in Distributed Sensor Networks Using Mobile Agents", *Proc. of the 4th International Conference on Information Fusion*, Montreal, CA, 2001.
- [74] J. M. Rabaey, M. J. Ammer, J. L. daSilva Jr., D. Patel and S. Roundy, "Pico Radio Supports Ad Hoc Ultra-low Power Wireless Networking", *IEEE Computer Magazine*, pp. 42-48, 2000.
- [75] J. Rabaey, J. Ammer, J. L. daSilva Jr. and D. Patel, "PicoRadio: Ad-hoc Wireless Networking of Ubiquitous low-energy Sensor/Monitor nodes", *Proc. of the IEEE Computer Society Annual Workshop on VLSI (WVLSI00)*, pp. 9-12, Orlando, Florida, April 2000.
- [76] V. Rodoplu, T. H. Meng, "Minimum Energy Mobile Wireless Networks", *IEEE Journal of Selected Areas in Communications*, vol. 17(8), pp. 1333-1344, 1999.
- [77] C. Schurgers, G. Kulkarni and M. B. Srivastava, "Distributed Assignment of Encoded MAC Addresses in Sensor Networks", *Proc. of ACM MobiHoc'01*, Long Beach, CA, October 2001.

- [78] C. Schurgers and M.B. Srivastava, "Energy Efficient Routing in Wireless Sensor Networks", *Proc. of MILCON Conference on Communications for Network-Centric Operations: Creating the Information Force*, McLean, VA, 2001.
- [79] S. Shakkottai, T. S. Rappaport and P. C. Karlsson, "Cross-layer Design for Wireless Networks", *IEEE Communications Magazine*, October 2003.
- [80] C. Shen, C. Srisathapornphat, C. Jaikaeo, "Sensor Information Networking Architecture and Applications", *IEEE Personal Communications*, pp. 52-59, August 2001.
- [81] E. Shih, S. Cho, N. Ickes, R. Min, A. Sinha, A. Wang and A. Chandrakasan, "Physical Layer Driven Protocol and Algorithm Design for Energy-efficient Wireless Sensor Networks", *Proc. of ACM MobiCom01*, pp. 272286, Rome, Italy, July 2001.
- [82] S. Singh, M. Woo and C. S. Raghavendra, "Power-aware Routing in Mobile Ad hoc Networks", *Proc. of ACM MobiCom98*, pp. 181-190, Dallas, Texas, 1998.
- [83] A. Sinha and A. Chandrakasan, "Dynamic Power Management in Wireless Sensor Networks", *IEEE Design and Test of Computers*, March 2001.
- [84] K. Sohrabi, J. Gao, V. Ailawadhi and G. J. Pottie, "Protocols for Self-Organization of a Wireless Sensor Network", *IEEE Personal Communications*, vol.7 (5), pp. 16-27, October 2000.
- [85] W. Stallings, "Wireless Communications and Networks", *Pearson Education*, 2002.
- [86] T. H. Stankovic, J.A. Chenyang Lu and T. Abdelzaher, "SPEED: a Stateless Protocol for Real-time Communication in Sensor Networks", *Proceedings. 23rd International Conference on Distributed Computing Systems*, pp. 46-55, May 2003.
- [87] I. Stojmenovic and X. Lin, "GEDIR: Loop-Free Location Based Routing in Wireless Networks", *Proc. of the International Conference on Parallel and Distributed Computing and Systems*, Boston, MA, November 1999.
- [88] C. Srisathapornphat, C. Jaikaeo, C. Shen, "Sensor information Networking Architecture", *International Workshop on Parallel Processing*, pp. 23-30, September 2000.

- [89] H. Tan, Köpeoglu, "Power Efficient Data Gathering and Aggregation in Wireless Sensor Networks", *ACM SIGMOD Record archive, SPECIAL ISSUE: Special section on sensor network technology and sensor data management*, pp. 66-71 , vol. 32 (4), December 2003.
- [90] S. Vural and E. Ekici, "Wave Addressing for Dense Sensor Networks", *Proc. of Second International Workshop on Sensor and Actor Network Protocols and Applications (SANPA 2004)*, pp. 56-66, Boston, MA, August 2004.
- [91] B. Walker and W. Steffen, "An Overview of the Implications of Global Change of Natural and Managed Terrestrial Ecosystems", *Conservation Ecology 1 (2)(1997)*, Available from <http://www.consecol.org/vol1/iss2/art2>.
- [92] X. Wang, G. Xing, Y. Zhang, C. Lu, R. Pless and C. Gill, "Integrated Coverage and Connectivity Configuration in Wireless Sensor Networks", *Proc. of the ACM SenSys 2003*, 2003.
- [93] B. Warneke, B. Liebowitz and K. S. J. Pister, "Smart dust: Communicating with a Cubic-millimeter Computer", *IEEE Computer*, pp. 2-9, January 2001.
- [94] G. Werner-Allen, J. Johnson, M. Ruiz, J. Lees and M. Welsh, "Monitoring Volcanic Eruptions with a Wireless Sensor Network", *Proc. of the Second European Workshop on Wireless Sensor Networks*, pp. 108-120, January 2005.
- [95] L. Xia, X. Chen and X. Guan, "A New Gradient-Based Routing Protocol in Wireless Sensor Networks", *ICISS 2004, Lecture Notes on Computer Science 3605*, pp. 318-325, 2005.
- [96] Y. Xu, J. Heidemann and D. Estrin, "Geography-informed Energy Conservation for Ad-hoc Routing", *Proc. of the Seventh Annual ACM/IEEE International Conference on Mobile Computing and Networking*, pp. 70-84, 2001.
- [97] Y. Xu, W. Lee, J. Xu and G. Mitchell, "PSGR: A Stateless Routing Protocol for Location-Aware Wireless Sensor Network", *Poster, Industry Day PSU*, University of Pennsylvania, 2005.

- [98] W. Ye, J. Heidemann and D. Estrin, "An Energy-efficient MAC Protocol for Wireless Sensor Networks", *Proc. of IEEE INFOCOM 2002, Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 3, pp. 1567-1576, 2002.
- [99] F. Zhao and L. J. Guibas, "Wireless Sensor Networks: An Information Processing Approach", *The Morgan Kaufmann Series in Networking, Elsevier*, 2004.
- [100] M. Zorzi and R. Rao, "Error Control and Energy Consumption in Communications for Nomadic Computing", *IEEE Transactions on Computers*, vol. 46(3), pp. 279-289, 1997.

Index

Addressing protocols

- distributed assignment of MAC, 26
- distributed global identification, 27
- dynamic addressing scheme, 23
- node address naming, 26
- RETRI, 25
- TreeCast, 26

Graph, 32

- connected graph, 33
- definition, 32
- disconnected graph, 33
- spanning tree, 35
- unit disk graph, 34

graph

- tree, 35

HN-Addressing

- address allocation
 - preoder* numbering of nodes, 50
 - bits per level, 54
 - communication graph, 50
 - finding spanning tree, 50
 - with hierarchical levels, 53
 - without hierarchical levels, 49
- address allocation algorithm , 65

- address allocation technique, 55
 - address allocation messages, 56
 - numbering of nodes, 58
 - ready queue, 56
 - states of node, 57
 - with hierarchical levels, 59
 - without hierarchical levels, 56
- address parameters, 72
- address reallocation, 61, 63, 64, 66
- data routing, 52, 55
- design goals, 48
- hierarchical levels, 54
- node failure, 61, 62, 64
- state diagram, 58, 67

MAC address, 5

Multihop communication, 3

network address, 5

Research issues in sensor networks, 20

- communication architecture, 21
- information processing, 21

Research projects on sensor networks, 22

Routing protocols

- directed diffusion, 28
- flooding and gossiping, 28

- gradient-based routing, 30
 - LEACH, 31
 - location based routing, 31
 - rumor routing, 29
 - SPIN, 28
 - stateless routing, 31
- Sensor network, 12
- Sensor network simulation
 - SensorSim, 43
 - TOSSIM, 42
- Sensor network simulators
 - NS-2, 43
- Sensor networks
 - applications, 16, 17
 - design factors, 16
 - fault tolerance, 18
 - hardware constraints, 18
 - power consumption, 19
 - scalability, 17
 - transmission media, 19
 - future application trend, 16
- Sensors, 16
 - sensing task, 2
- Simulators
 - GloMoSim, 45
 - Parsec, 44
- TreeCast, 35
 - address allocation, 35
 - messages
 - control, 40
 - query, 39
 - response, 39
 - routing, 39
- Wireless networks, 14
 - Bluetooth, 15
 - cellular network, 14
 - HomeRF, 15
 - Mobile ad hoc network (MANET), 14

