ANALYSIS AND DEVELOPMENT OF A MATHEMATICAL STRUCTURE TO DESCRIBE ENERGY CONSUMPTION OF SENSOR NETWORKS

by

Peter Joseph Hawrylak

BS, University of Pittsburgh, 2002

MS, University of Pittsburgh, 2004

Submitted to the Graduate Faculty of

School of Engineering in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2006

UNIVERSITY OF PITTSBURGH

SCHOOL OF ENGINEERING

This dissertation was presented

by

Peter Joseph Hawrylak

It was defended on

November 10, 2006

and approved by

Dissertation Director: Marlin H. Mickle, Nickolas A. DeCecco Professor, Electrical Engineering Department

Dissertation Director: J. Tom Cain, Professor, Electrical Engineering Department

Steven P. Levitan, John A. Jurenko Professor, Computer Engineering Department

Heung-no Lee, Assistant Professor, Department of Electrical and Computer Engineering

J. Robert Boston, Professor, Department of Electrical and Computer Engineering, Department of Bioengineering, Department of Communications Science and Disorders

Mike Lovell, Associate Professor Industrial Engineering Department

ANALYSIS AND DEVELOPMENT OF A MATHEMATICAL STRUCTURE TO DESCRIBE ENERGY CONSUMPTION OF SENSOR NETWORKS

Peter Joseph Hawrylak, PhD

University of Pittsburgh, 2006

Collections of several hundred, thousands, or even millions of small devices scattered or placed throughout an area monitoring the environment called sensor networks have several useful applications. Until recently, the economic cost of development, manufacture, and deployment limited the use of sensor networks to military and government applications. Recent advances in technology provide a means for economical development, deployment, and manufacture of sensor networks.

Current methodology designs, then implements and simulates the sensor network, then goes back and redesigns to better meet the specifications. The model developed in this dissertation provides an early indication of what types of solutions will meet the requirements and what types of solutions will not. With this ability, the time required for simulation and proof of concept is reduced, allowing more time and money for design and testing of the real world system.

The model developed characterizes the energy consumption of a sensor or RFID network as a whole is extremely beneficial and is needed. The model provides a means to benchmark different types of sensor networks (i.e. different protocols, hardware, software) and to determine which type is the better solution. A model such as this removes the requirement to develop a simulation to compare different types. Using the model reduces the time (and save money) needed to verify the solution and helps with development as multiple designs can be quickly tested and compared possibly at a much earlier stage in the development cycle allowing a thorough investigation of different design alternatives.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1.0 INTRODUCTION

A collection of hundreds, thousands or even millions of small inexpensive devices with sensing capabilities scattered or placed throughout an area is commonly referred to as a sensor network. RFID networks consist of hundreds or thousands or millions of tags and dozens of readers. The devices that compose the sensor or RFID networks communicate via wireless communication equipment. The ability of the sensor network to monitor an area for extended periods of time is a critical application requirement. RFID systems must track items for up to 25 years. Current advances in technology have made the creation and deployment of sensor and RFID networks not only technologically, but also economically feasible. Specifically, the availability of powerful, low power, and inexpensive processors and accompanying hardware facilitate the development of sensor and RFID networks. The key to wide scale deployment of these networks is in the reduction of energy consumption both of individual nodes (or tags) and of the network as a whole.

The concept of a sensor network is not new as wired sensor networks have been in place for many years. The availability of low cost electronics for sensors has made the cost of the wired connections between the sensors and the central controller a significant part of the overall system cost [1]. The need for a wired connection to exist between the sensor and the central controller places significant limitations on where sensors can be placed and deployment in general [1]. For example, sensors cannot be placed in numerous applications having moving parts because the wire connection would either break as a result of the movement or the wire would get in the moving parts and cause the machine that the sensor is to monitor to fail [1]. The limited ability to resist interruptions in communication links is another drawback of wired sensor networks [1]. In the event of a failure or death of some sensors in a wireless sensor network, it is still possible for the remaining sensors to reroute their messages through those sensors still in operation [1]. In a wired sensor network once a communication link is broken, all

communication that traveled through the broken link is lost, thus multiple wired links are needed to provide a comparable level of resiliency [1].

The military has performed extensive research into and deployed a number of sensor networks. One of the most notable of these networks is the SOund SUrveillance System (SOSUS), deployed in the Cold War to monitor movement of submarines [2]. Today, the National Oceanographic and Atmospheric Administration (NOAA) uses the SOSUS sensor network to monitor conditions of the oceans [2]. A system such as this could be used to monitor for and provide advanced warning of tsunamis.

Sensor networks have a wide range of possible military, commercial and public safety uses. Widespread use of sensor networks can provide solutions to problems in the areas of environmental monitoring, security, traffic control, and monitoring conditions within an area or structure. Monitoring the environment on a large scale is an ideal use of sensor networks. Ecologists require a great deal of information about the environments of organisms they study [3]. If the environment under study is remote, the frequent trips required to collect the data can become prohibitive to further study. Disturbances caused to the creatures and the environment under study in collecting the data can result in unintended harm to the creatures or environment being studied [4]. Sensor networks provide a means to monitor the environment with the required granularity while causing little disturbance of the organisms under study [4]. Providing the ability to monitor large areas, sensor networks, allow possible intruders to be quickly discovered and apprehended [2]. The use of sensor networks to monitor traffic conditions in order to reduce congestion is another promising area of development [5]. Indoor climate control systems continue to advance, incorporating an increasingly complex amount of technology in the system. With respect to climate control the more temperature readings available to the system, the better the system can modulate the amount of heat or cooling generated and the distribution of that heat or cooling, providing a more efficient and conformable environment.

The field of sensor networks is concerned with designing, deploying, and investigating sensor networks. Of critical importance to sensor and RFID networks is the power consumption of individual nodes and of the network as a whole. The nodes can derive the power needed for operation from an onboard power supply, harvest energy from the environment, or from an external power source. Nodes deriving power from an external source place severe limitations on the applications available to networks consisting of such nodes. Energy harvesting, while

promising, still requires significant development to enable enough energy to be harvested to power a node. Therefore, the node must be powered from an onboard power supply, specifically a battery. The node remains operational as long as the battery can provide the needed operational power.

Minimizing the power consumption of an individual node, will increase the lifetime of that node. Depending on what other factors are altered in minimizing the power consumed of a single node, the network lifetime could increase, decrease, or remain unchanged. Therefore, it is important to investigate power consumption on an individual node and over the entire network. Minimizing power consumption over the network as a whole will result in increased lifetime of the network.

Work in the field of sensor networks has generated a significant amount of attention in recent years. Much work has been done in the development of communication and routing protocols, algorithms for the initial configuration and maintenance of the network, with some work in simulation of these sensor networks to evaluate performance.

1.1 CURRENT/RELATED WORK IN SENSOR NETWORKS

Current work in sensor networks focuses on several different areas. Research investigating initialization, construction, and routing protocols has received the most attention. Optimizing the operation of the individual nodes making up the sensor network comprises another area of research. Network architectures for selective activation of nodes or for operations performed on data to reduce the amount of data transmitted through the network are a growing area of interest. Security of sensor networks and the data transmitted through the network is also a growing area of research as sensor and RFID networks are quickly becoming economically feasible to deploy. Energy harvesting is another growing area of interest, especially since sensor networks rely on limited power supply, usually a battery. Mobility of nodes within the sensor networks presents significant problems to maintaining communication paths within the network. However, networks with mobile nodes are being investigated as mobile nodes can provide additional benefits and open new areas for use of sensor networks.

Routing and initialization are critical to the operation of a sensor network and most research in sensor networks is focused in this area. While some sensor networks are deployed by hand and use predefined routing tables. In state of the art networks nodes will be randomly deployed over an area and must build and connect the sensor network themselves. Initialization is the building and connection of the individual sensors to form the sensor network. Initialization consists of achieving two goals. The first is to construct the network by finding and connecting nodes together to form a network. The second objective is setup the initial communication paths, routes, by which information will be disseminated. Routing focuses on the paths that messages take between entities in the network. Routing schemes for sensor networks must be flexible to handle changes (additions and deletions) of nodes in the network.

The first step in initializing a sensor network is for the individual sensor nodes to identify their neighbors. Several methods are possible for discovery, but a node must either listen for messages from other nodes announcing their presence, or broadcast a message announcing its presence. In one algorithm for network construction, new nodes first listen for an invitation to join the network [6]. Receiving an invitation within a certain time causes the new node to broadcast a request to join the network [6]. Nodes form local networks with the other nodes they discover. This process continues the local networks continue to grow and the network gradually becomes connected [6].

The location of each sensor node is an extremely useful piece of information to determine the routing paths within the network. Knowledge of location allows individual nodes to identify nodes that are closer to the destination node of a particular message. With that information, the message can be addressed only to those nodes that are closer to the destination. Another use of location is to base the routing paths on minimizing the distance between all the nodes since received power is inversely proportional to the distance squared.

However, obtaining the location of each sensor node presents significant problems. Clearly if each node is placed intentionally, the location of each node would be known and could easily be programmed into the node. Placing each node when a network may contain thousands of nodes is not feasible, and this solution is limited only to very small sensor networks. Location finding devices such as GPS are costly in terms of energy use, and take up valuable real estate on the node. Further, the error range of a few meters, of the GPS system, significantly reduces the usefulness of the location provided, because the distance between nodes will usually be only a

meter or two at the most. Relative distances between two nodes can be estimated through measurements of the signal strength of messages from the other node. While this relative distance does not give an exact position (i.e. latitude, longitude, and altitude) knowing just the relative distance allows for significant improvement and optimization for during topology formation and routing. Using three special nodes the location of a fourth node can be obtained through triangulation based on the received signal strength of a message from a fourth node [7]. Mondinelli and Kovacs-Vajna present another location finding method using just one special node, but that method requires complicated computations and other regular nodes to participate in the process as well where the method using triangulation does not require the participation of other regular nodes.

The topologies used in sensor networks can be generalized into two categories; cluster topologies, or flat topologies. Cluster topologies partition the network into a set of smaller local networks, or clusters. Each cluster has a cluster head that is responsible for communication among clusters. Further, establishment of a local network within each cluster links each node to the cluster head. Routing within a cluster is simplified as the number of nodes and possible paths are reduced to only those in the cluster. Clustering also allows for the application of higher-level optimizations based on clusters rather than individual nodes. One such optimization is to partition all the sensor nodes into mutually exclusive sets of nodes that cover the entire area (coverage area is defined by the application and node) being monitored [8]. Only one of the sets must be active at any given time, thus for a network of homogeneous nodes, the lifetime of the network is directly related to the number of sets, increasing as the number of sets increases [8]. In a flat topology, the nodes are not grouped together.

Once the topology is defined communication links and routing paths must be setup to allow information to flow through the network. The primary objective of any routing algorithm is to discover communication paths that enable all nodes in the network to communicate their data to the network. The secondary objective of the routing algorithm is to minimize the energy consumption of the entire network keeping the data delay within acceptable limits, and ensuring that the network remains connected.

Networks that employ a cluster topology can perform higher-level optimizations to setup the data routes. For example, the LEACH protocol randomly changes the designated cluster head distributing the additional energy consumed being a cluster head throughout the cluster and

the PEGASIS protocol, which extends LEACH, can be used if enough global information is known about the network [9].

Other protocols support data aggregation in a network, or data centric networks. Data aggregation attempts to reduce the amount of data sent throughout the network by combining the data received with the sender's data, for example through averaging or compression. The majority of research focusing on data centric sensor networks sees the sensor network as a large database. Data centric networks view the request similar to an SQL query in a database. The request will contain information specifying what data are requested and only those nodes with data matching the requested data type will respond.

Sensor networks must continue to function as nodes die. From a routing viewpoint, the simplest method to increase the fault-tolerance of a network is to send data along multiple paths. As long as one path is intact, the data will arrive at the destination. One routing algorithm forms multiple routes between a source and a destination by providing intermediate nodes several choices of where to forward the packet [10]. With multiple choices, most, or ideally all intermediary nodes, the network can tolerate significant numbers of node failures before the network becomes completely disconnected, and the network can better balance the communication load through selective routing [10]. These benefits can increase the lifetime of the network, but sending a single message along multiple paths obviously requires more energy than sending the message along a single path. The drawbacks to this solution are that as the number of nodes in the network increases so does the potential size of the routing tables stored in the nodes and the overhead associated with finding and using multiple routes.

Optimizations applied to individual nodes are another area of research interest. Most optimization strategies focus on reducing the power consumption of the processor. Optimization of the other hardware contained on the sensor node, such as the sensor, analog to digital converter (ADC), etc., also reduces power, but short of custom designed components selecting more efficient off the shelf parts is the only solution.

Reducing power consumption at the node level can be achieved through more efficient processing and routing of messages. Sensor networks generate huge numbers of messages and a message arriving at a node falls into one of three categories; the message is for that node, the node must relay the message, or the message can be ignored. The wireless interface normally handles the physical reception and collection of the message, and the processor decodes and

processes the message. In this scheme, the processor determines which of the three categories the message falls into and takes appropriate action. In the case that the message is to be relayed to another node, custom logic has been developed and integrated onto the wireless interface board to relay the message to the next node if the message is to be relayed [11]. This reduces the node power usage as the message is not transferred to the processor board, decoded, and then back to the wireless interface board for retransmission [11].

Another node level optimization is to ignore all packets not destined for the node. Particularly useful in networks made up of active RFID tags, the ignoring of messages addressed to other tags saves power. Waking the processor up only when a message arrives addressed to the tag will allow the processor to remain in a dormant state for the maximum amount of time. This results in significantly less energy consumption per unit time over a tag, that requires the processor to decode every message to check the address. This optimization works best when the network is frequently interrogated.

Management of the processor state is critical to the energy consumption of a node. The ideal node would keep the processor in the lowest level sleep mode unless work must be done. However, entering and leaving each level of sleep mode requires a time delay to power down or up, as required information must be stored or retrieved by the processor to transition between modes [12]. During this time, no useful work can be performed, so the energy consumed during this time is wasted, thus the processor must remain in that state long enough to save at least as much power as was wasted. Sinha and Chandrakasan have developed an algorithm that attempts to predict the future workload, and by extension predict how long the processor can remain in a given state and use this to determine when and what level of sleep mode to enter. If the workload is predicted to be high in the immediate future the algorithm will either stay in the active processing mode or enter a sleep mode saving minimum power, but requiring minimum delay and energy to enter and leave [12].

Sensor networks have tasks that must be performed throughout the network. Some or all of the tasks may be able to be processed on several different nodes, while still providing the required level of detail and accuracy. In such a case, tasks can be assigned to different nodes to prevent some nodes from quickly draining their energy reserve and failing. One algorithm has been developed to spread the workload over a group of nodes. Assigning a reward and an energy cost to each task as a part of a network policy, each node is then able to determine if it should

perform a given task based on the reward for performing it, the network policy, and the remaining energy in the node [13]. Nodes that are low on energy will perform less tasks or tasks with higher rewards than nodes with significant energy remaining. One drawback of this solution is that if all nodes that are issued a request to perform a certain task are low on power, there is a possibility that none of the nodes perform the task. If the task were critical, the network would fail to meet the operational requirements, which would be a failure of the same magnitude as a significant number of nodes failing due to lack of energy.

Sensor networks containing a large number of nodes also contain and ultimately report huge amounts of information. Rarely are all the available data required, and usually only more specific data are required, such as the temperature, or the temperature in a certain area of the network. Sending back readings from all nodes wastes the energy of those nodes whose readings are not needed, and communication is greatly increased, further wasting energy. One solution is to reduce the amount of information reported. This can be accomplished by activation of only a select number of nodes, or the combination or compression of results as they are making their way through the network.

Selectively requesting and reporting data reduces energy consumption of the sensor network and reduces the amount of processing required on the entity that must process the readings and make decisions based on those readings. One way to reduce the amount of data is to query the sensor network for only the desired information. Several papers have compared a sensor network to a large database, distributed over a large number of nodes [14, 15]. These papers envision the querying of the sensor network with SQL like commands where only those nodes with the requested data would reply. This is similar to the use of the SELECT SQL command used in a database to return those records matching the query. This solution would require additional processing to be performed by the nodes to determine if they have the requested information, but it reduces the amount of data sent back. The energy consumption of the additional processing must be weighed against the energy savings of reducing the amount of data sent through the network.

Sensor networks that track targets require only those nodes in the area of the target to be actively monitoring. Research has been conducted by Zhao, et. al., for development of a network that determines which sensors should be activated and how to handoff the monitoring requirements as conditions change [16]. Target tracking is an ideal application for this type of

network, because only those sensors within range of the target need to be actively monitoring it, the other sensors can remain dormant until the target enters their area. They propose methods to estimate the useful information added with the addition of information from another sensor. Those sensors that maximize the estimates will be activated. Keeping sensors without much useful information dormant increases the lifetime of those sensors and possibly that of the network.

Selective querying and activation reduces the number of replies, but in larger networks, the number of replies can still be large. Data fusion and compression can reduce the amount of data transmitted while keeping a high number of replies. One solution is to average the reported data over a small geographic area and then send only that average on to the destination. This reduces the amount of data that must be processed and the number of messages sent to the destination. This solution is easily implemented in a network that can be divided into clusters by having each cluster head average the data and sending only that average. Another solution is compression of the data in order, to send the same information but with fewer bits. One such solution is to divide up the data space of the response into indexed bins, each containing a portion of the response space, and then only send the index of the bin back [17]. As long as the length of the index is less than the length of the data, sending the index reduces the amount of total number of bits transmitted. Energy is saved by transmitting fewer bits. The sizing of the bins has impacts on accuracy and precision as well as the amount of energy saved.

Security in sensor networks has received less attention than research focusing on implementation and deployment of the network. However, security will be an issue as sensor networks are deployed. Chan and Perrig list eavesdropping, data privacy, and attacks on networks themselves as some of the security concerns with sensor networks [18]. Monitoring the messages transmitted through the network can be as simple as listening on the same frequency that the nodes use, or by inserting nodes into the network to collect data. Private data can now be accessed and read by third parties. Encryption is one possible solution, but it requires significant strength, robust key distribution, and more energy [18]. Another solution suggested is to fragment the data and send each fragment through a different route [18]. This would require the attacker to successfully monitor all or a majority of the possible routes. This could reduce the required strength of the encryption scheme. Attacks on sensor networks can take the form of jamming communications or depleting the remaining energy of the nodes [18].

Jamming prevents communication as the channel becomes unusable, but using communication schemes specifically targeted at overcoming noisy channels can counter jamming, and jamming can be quickly discovered [18]. Draining the remaining energy of enough of the nodes will result in the network becoming too disconnected to function. Injecting large numbers of commands into the network can quickly drain the batteries of the nodes and may not be immediately detected [18]. Requiring authentication of commands will counter this attack [18]. However, these counter measures must be carefully designed as they will consume more energy than they can save.

A few researchers have looked at using energy harvesting to power wireless networks, or recharge batteries in the nodes [19-22]. This work can be applied to sensor networks as well. Energy harvesting methods viable for use in sensor networks include RF energy harvesting, thermoelectric generation, solar power, and harvesting energy from vibrations in the environment.

The addition of RF energy harvesting circuitry to the node requires minimal additional hardware as the node is already equipped with an antenna for communication. The large amount of available RF in the environment both from the sensor network and outside sources (i.e. radio stations) makes RF energy harvesting one of the leading choices for energy harvesting in a sensor network. However, the drawback of RF energy harvesting is that it is limited to the amount of power that can be received by the antenna. The received power in free space is determined by the following equation.

$$P_r = \frac{P_t \lambda^2}{(4\pi)^2 d^\alpha} \tag{1-1}$$

Where λ is the wavelength of the carrier frequency, d is the distance between the transmitter and the receiver, α is the power that the distance, d, is raised to, P_t is the transmit power, and P_r is the received power. The free space model uses 2 for α, while in a real world environment α will be larger due to interference. Thus, the received power decreases quickly as distance increases making it nearly impossible to power sensor networks with current technology using RF energy harvesting.

Thermoelectric generation is another method of energy harvesting applicable to sensor networks. Thermoelectric devices generate energy in response to a difference of temperature across the generator, the larger the difference the more energy generated. However, for small temperature differences, little energy can be generated. An individual node in a sensor network will normally not experience large temperature differences as the internal temperature of the node will be close to the temperature of the surrounding environment. The resulting small temperature difference is not enough to provide the energy required by the node for operation.

Solar power is another alternative for powering sensor networks. Solar cells harvest energy from light in the environment. Currently solar cells can harvest sufficient energy for small devices such as a handheld calculator to operate. The drawback of solar cells is their need to be in a brightly illuminated environment. In a low light environment, the solar cell generates little energy. Another drawback is that the light level is rarely constant, for example, a node outdoors should have enough light in the day but not during the night. Uncertainty with respect to the maximum light level and the consistency of the light level available to each node after deployment of the network makes solar power ineffective.

Using vibration to generate energy has long been used in the manufacture of self-winding watches [20]. Piezoelectric materials are commonly used to harvest energy from vibration. This type of energy harvesting suffers from similar drawbacks as solar cells as they are dependent on the environmental conditions. Most sensor networks are targeted at deployment over a stationary terrain such as a forest or desert that experiences very little vibration. Further, the vibration would have to exist for a long period to recharge the batteries that power the node. For these reasons, harvesting energy from vibrations provides limited support for sensor networks.

Several networks have been proposed and studied in literature utilizing energy harvesting [21, 23]. The work done by Kansal and Srivastava proposes a network that attempts to divide tasks among the nodes based on the amount of energy each node can harvest from the environment. The goal is to distribute the workload such that those nodes that can harvest more energy will perform more work. This extends previous work investigating distributing work based only on remaining battery energy. Dividing the workload in such a manner requires that nodes be able to determine or estimate the amount of energy they can harvest from the environment and communicate this value to other nodes. Determination of the available energy and communication of that value with other nodes must be done in an energy efficient manner.

Scheduling tasks based on the energy available to each node and remaining battery power theoretically increases the lifetime of the network simulated. Rahimi, et. al. describe a network architecture consisting of two types of node, the first having the ability to harvest energy, move around and replace depleted batteries with batteries with a full charge. The second are regular nodes that cannot recharge their batteries or move. The number of energy harvesting nodes required to cover an area is determined by the total area, energy for the energy harvesting node to move, available energy, energy consumption of regular nodes, and the number of regular nodes [21]. Implementation of such a network requires that enough available energy can be harvested, and that the terrain over which the network is deployed allows movement of the energy harvesting nodes to replace batteries. Even with high efficiency energy harvesting technology and an abundant source of ambient energy, the inability to move that energy over rough terrain prevents replacement of depleted batteries.

1.2 DISCRETE EVENT SIMULATION

Discrete event simulations provide a model platform for a system by determining and updating a set of state variables describing the system at each time increment. Discrete simulations can be expanded into a number of different types. Two of the most common are event driven and time stepped simulations [24]. In time stepped simulations, time is advanced an equal amount at the time step at which point the simulation states are updated [24]. In a time stepped simulation all events occurring within one time step are assumed to happen at the same time, thus the choice of the length of the time step in a time stepped simulation is critical to the accuracy and precision of the results [24].

Event driven simulations update the state only when something of interest, called an event, happens [24]. Each event contains information indicating the time that the event is to occur in the simulation [24]. The simulation state is only updated in response to the occurrence of an event requiring that the order in which events are processed be maintained in a chronological ordering (i.e. earliest event is processed first) [24]. Ensuring this chronological ordering of events results in the primary overhead in the simulation and thus causes the simulation to slow down considerably [4]. Even with optimizations, the number of events

generated per second for a small or medium sized sensor network is considerable. Time stepped simulations are usually faster but provide less accurate results than discrete event simulations [4].

Discrete event simulations use events to pass messages between two simulation entities. Events represent such things as one entity sending a data message to another entity. Events in discrete event simulations contain a time stamp indicating the time that an event is to occur. Events must be processed in order based on their timestamps [24]. This is termed, "the synchronization problem" [24]. In a parallel simulation, the events must be processed in such a manner as to generate the same result as given by a sequential system processing events one-by-one in time stamp order [24]. Processing events out of order can lead to situations in which some entities have advanced to a time ahead of still unprocessed events leading to "causality errors" [24].

Simulations of wireless networks typically use the event driven discrete model for their improved accuracy. Simulation entities typically represent nodes or sinks in the sensor network, and events typically represent the messages sent between nodes or sinks. For each message sent, two events may be required, one event for the transmission of the message, and a second event for the receipt of the message [25]. While not a significant problem for simulations of networks containing a small number of nodes, the event ordering overhead becomes an issue as the size of the network increases. One study reports that for a network of 3,200 nodes more than 5.3 million events per second were generated, using their proposed optimizations the number of events per second was reduced to slightly more than 210,000 [26]. Adding more processors to the simulation environment only elevates the problem to a point, where the extra messages passed between the processors will begin to reduce or negate gains achieved.

The synchronization algorithms used by discrete event simulators to determine when and which events can be processed generally fall into one of two categories; conservative synchronization, or optimistic synchronization. In conservative synchronization, an event is processed if and only if the simulation determines that there is no event with a timestamp before the event in question, and that no event with a timestamp for a time before the timestamp of the event in question will be received in the future [24]. The possibility exists where all simulation entities are waiting on possible events from other entities and all entities block, resulting in deadlock [24]. There are a number of methods to recover from a deadlocked situation. All deadlock recovery methods require additional overhead in the form of first detecting a deadlock

situation, then determining what messages to send to break the deadlock, and finally the actual sending of messages to break the deadlock. These additions increase the amount of communication, the amount of processing, and ultimately the running time of a parallel discrete event simulation.

Parallel simulations must prevent messages from being delivered to a simulation entity with a time stamp in that entity's past. Messages that are still propagating through the network and have not been delivered to the receiving entity yet, called transient messages, must be taken into account when a simulation entity wishes to advance the simulation time [24]. The problem surfaces when a transient message exists for an entity having time stamp, T1 and the entity, not aware of the transient message, finds that it is safe to advance to time T2, and T2 > T1. Causality of the simulation is violated when the transient messages arrives at the entity with time stamp T1, since the entity is now at time T2, and T2 > T1. Detection of the presence of transient messages requires entities to keep track of the number of messages they have sent and the number of messages they have received [24]. This requirement adds to the simulation overhead as two counters must be maintained in the system, and additional processing must be performed by the simulation controller when an entity wishes to advance its simulation time. Further, some entities will experience additional delays because they must wait until no transient messages are present to advance their simulation time.

The amount of time that the simulation time can be advanced must be determined for each time advance. Determination of the amount of time the simulation can be advanced requires information from other simulation entities. In the worst case, all entities require information from all other entities. In a simulation consisting of N entities, in the worst case N^2 messages containing the required information are sent through the simulation environment [24]. While methods to improve the required number of messages are possible, determination of the amount of time the simulation can advance still requires significant overhead [24]. Thus, maximizing the amount simulation time advances at each advance is critical to the performance of the simulation [24]. Simulations in which the maximum amount of time that the simulation can advance at each step are limited to a small value will not scale well as entities are added to the simulation [24].

Optimistic synchronization algorithms form the second major class of synchronization algorithms for discrete event parallel simulations. Optimistic synchronization algorithms allow

for the processing of messages without first determining if they are safe allowing for a possible violation of causality in the simulation [24]. The simulation is rolled back to a previous simulation time in response to a causality violation [24]. Pipelining microprocessor architectures are one example of an optimistic synchronization algorithm, and provisions are made to remove those instructions that were issued when a hazard is detected [24]. Optimistic algorithms do not require that messages be sent or received in order based on their time stamps, nor do communication links between simulation entities need to be explicitly determined and defined [24].

Supporting simulation roll back requires the storage of information describing the state of the system in the past [24]. Two popular methods to store state information are copy state saving, where all state variables are stored before processing each event, and incremental state saving, where for each event the prior value of every state variable modified by that event is recorded in a log [24]. Better performance is achieved using copy state saving in simulations where a high percentage of the state variables are modified by each event, and incremental state saving results in better performance when only a small percentage of the state variables are modified with each event [24]. Both methods require additional overhead for storing the redundant state information needed to support roll back. The additional memory needed grows as the number of simulation entities increases and as the number of events increases because events cause changes to the state variables. This memory requirement can severely limit the ability to simulate large networks.

In addition to state saving, support of simulation, roll back requires a method to cancel messages that were sent incorrectly [24]. The messages requiring deletion were sent by simulation entities that did not process a message with a time stamp earlier than their current time [24]. The data contained within these messages may be altered by the earlier message (message causing the roll back) or the message may not have been sent at all. Processing of these messages can initiate a causality violation in the simulation, and possibly lead to incorrect results. In the Time Warp system, anti-messages are used to destroy messages that must be unsent [24]. When an anti-message and a message appear together in the input queue they cancel each other out and thus the message is deleted [24]. An anti-message must be sent for each message that must be deleted due to a roll back [24]. In cases were large numbers of messages must be deleted per roll back, this can add significant overhead to the simulation.

To be valid, a simulation must produce identical results over multiple runs of the same simulation [24]. For a parallel processor, this requires that all events must be processed in the same order for each execution of the simulation [24]. Further, differences in the hardware used to execute the simulation can affect the results. For example, a difference in floating point calculations computed on two different processors can result in different results if the messages are not processed on the same processor for each execution of the simulation [24]. The difficulty in producing repeatable simulations significantly increases the difficulty in obtaining accurate comparisons of different systems using results of a simulation.

Messages transmitted in the network are not point to point as with a wired network, but are transmitted in all directions from the transmitting node. The only requirement is that the designated receiving node(s) be within range of the signal to 'hear' it. One implication of this communication strategy is that many nodes will overhear messages not designated for them. The nodes overhearing the message may still be required to process the message adding overhead to those nodes, and the simulator must decide the nodes within range and will hear the message sent by the transmitting node. This causes another performance bottleneck in discrete event simulations. In the worst-case in a simulation containing N nodes, for each node in the simulation the simulator must determine if a signal sent by any of the other N-1 nodes is received at the Nth node [27]. This leads to $O(N^2)$ checks per node [27]. This problem magnifies itself both as the number of nodes and the density of the nodes increase. Even though the number of checks per node may be able to be reduced, the increasing of this overhead with the increasing size of the network under simulation does not provide a platform that scales well. Although optimizations have been proposed [27] these come at a cost of reduced accuracy in the simulation results [26].

1.3 CURRENT/RELATED WORK IN SENSOR NETWORK SIMULATION

There currently exists a vast number of simulators for networks. A survey conducted by Akhtar lists a total of 42 different network simulators [28]. A brief survey of papers investigating sensor networks that identifies the method used to obtain results (i.e. simulation or modeling) is shown in Table 1.1 below.

Of the 19 papers surveyed, most papers used one of three simulators, GloMoSim, ns-2, or PARSEC as shown in Figure 1.1. All three simulators are discrete event simulators. In two papers, a custom simulator was used [29, 30]. Discrete event simulators have a number of limiting factors in the field of the simulation of large sensor networks.

Table 1.1: Number of times each method is used in the 19 papers surveyed.

Simulator	Number of Papers Used In
GloMoSim	5
ns-2	4
PARSEC	3
Implement and Test Proposed Solution	1
EmStar	1
Custom Simulator	2
OPNET	1
Matlab	1
Numeric or Analytic Evaluation	1

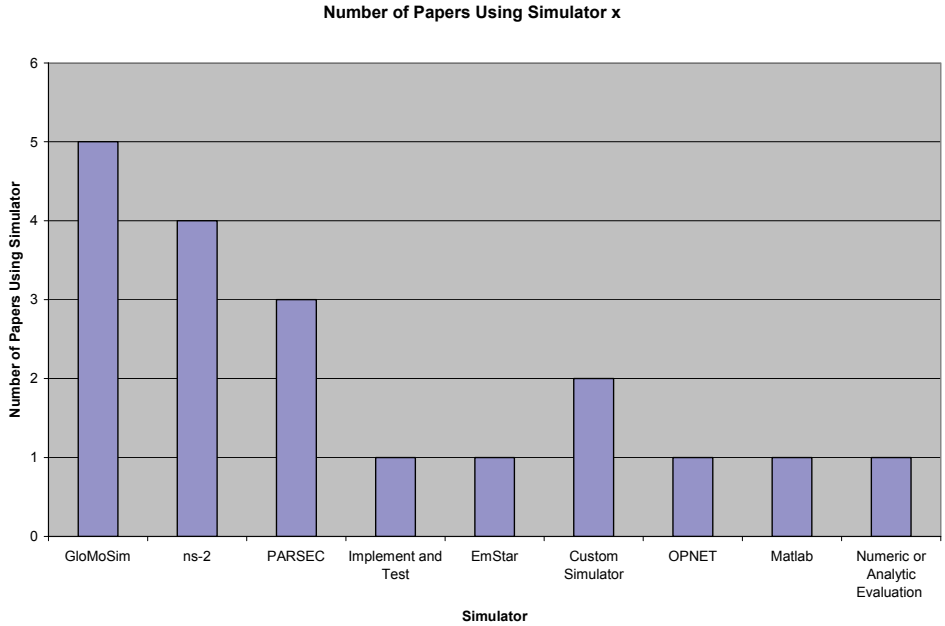

Figure 1.1: Graph showing usage of simulators.

GloMoSim is a library providing the capability for discrete-event simulation of wireless networks using PARSEC [31, 32]. The GloMoSim simulator was developed to provide an environment capable of simulating wireless networks containing thousands of nodes [31]. The GloMoSim simulator supports both sequential and parallel simulations [32]. The GloMoSim environment consists of a number of layers with each layer having interfaces to the layers immediately above and below the layer in question making it compatible with the seven-layer OSI model [31]. The layered structure of GloMoSim allows several different models to be evaluated at one level without requiring reimplementation of the other layers. The primary focus of GloMoSim is the modeling and evaluating the performance of protocols at the different layers of the OSI stack.

GloMoSim supports parallel simulation. As mentioned above, a critical factor influencing the efficiency of a parallel simulation is the number of messages sent between simulation entities. The simple solution of representing each node in a wireless network with an individual entity would quickly lead to simulations containing thousands of entities. This will result in several thousand or possibly more messages being sent between entities. Such a large number of messages will quickly saturate the communication infrastructure of the parallel system and performance will degrade significantly as the message delay increases. In order to reduce the number of messages passed between entities and to allow for scalability, GloMoSim divides the area where the wireless network exists into smaller areas [31]. Each smaller area is represented by a single simulation entity and contains all nodes placed within the smaller area [31]. Messages sent between nodes within a smaller area are routed locally by the simulation entity representing that area [31]. Only those messages from a node within one smaller area to a node within another smaller area are transmitted across the system communication structure [31].

Configuration of the GloMoSim simulation is achieved by altering an input file read by the simulator [31]. The area covered by the network is approximated by a rectangle with the length in both the x and y direction being customizable by the user [31]. The number of smaller areas is specified by specifying the number of divisions in the x direction (columns) and the number of divisions in the y direction (rows) [31]. An example of dividing the total area into nine smaller areas with three divisions in the x direction (rows) and two divisions in the y direction is shown in Figure 1.2 below. For example, for nodes N1 and N3 (or N2) to communicate, the message is routed internally by the simulation entity. However, if nodes N7

and N9 wished to communicate a message must be sent from the simulation entity containing node N7 is in to the simulation entity that contains node N9 over the communication structure of the system.

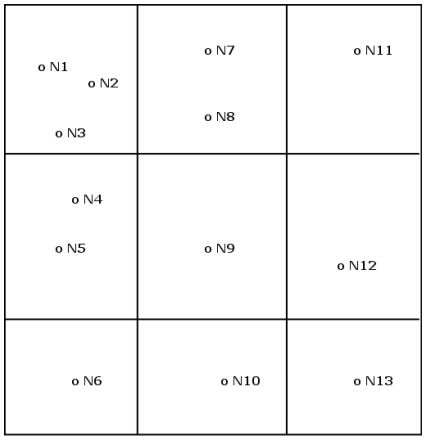

Figure 1.2: A network of 13 nodes, divided into nine smaller areas.

In addition to the number of smaller areas, the user can alter the maximum transmitter range of the wireless interface. Because the goal of dividing the entire area into smaller areas is to allow messages to be handled locally within the entity having a large enough smaller area that the majority of nodes cannot transmit outside of the smaller area best utilizes this optimization. Thus, the maximum transmitter range is critical to deciding the size of each smaller area, and by extension, the number of smaller areas. During the simulation there are many times when one of a number of events may occur, and one must be chosen (i.e. is a message corrupted by environmental noise). In most cases, a random number is obtained and used to determine which of several events occurs. GloMoSim utilizes a random number generator that requires a seed to initialize the random number generator. The seed affects the numbers generated and is configurable by the user. The user must also specify the maximum time that the simulation must execute.

Further, details about the nodes that make up the wireless network must be specified. First, the user must specify the total number of nodes in the wireless network. The nodes can be

placed within the area randomly, uniformly spaced within the area, in a 2-D grid (must be a full 2-D grid), and manually placed where the location (x and y coordinate) of each node is specified by the user [31]. Second, the model for signal propagation selected can be the free space model where received power is inversely proportional to the square of the distance between sender and receiver, the Rayleigh fading distribution, or the Ricean fading distribution [31]. Third, the data rate of the network must be specified.

GloMoSim provides support for mobile nodes. Two different types of mobility can be simulated within GloMoSim. The first type is random mobility in which a node moves one unit in either the x or the y direction. The second type of mobility simulates the node moving to a randomly selected waypoint. The speed at which the node moves, and the time that the node stays at the waypoint, once reached, are configurable by the user [31]. The simulation randomly selects the waypoint [31]. Finally, the simulation allows the user to set the position precision which determines how often the location of the mobile nodes must be updated [31].

As GloMoSim focuses mainly on the development and evaluation of protocols a number of built in protocols and algorithms are provided with the library [31]. These components can be added to simulation, allowing the developer to investigate other areas without having to implement those layers for which code is already available. Finally, the statistics that are collected during the simulation must be specified. Again the statistics available are based on protocol development and evaluation. Statistics such as number of packets sent and received, number of each type of packet (UDP, TCP, broadcast packets), and throughput are just some of the many statistics that are able to be logged by GloMoSim [31].

PARSEC is both a language for parallel programming and an environment for parallel simulation. PARSEC is general in nature and not specifically designed for wireless sensor networks or RFID networks [33]. PARSEC was designed around the message-passing C (MPC) kernel for parallel programming, providing the ability for PARSEC to be used as a programming language and a parallel simulation environment [33]. Development of PARSEC was in response to the lack of tools focusing on parallel simulation [33]. PARSEC provides the basic environment and framework for parallel simulation in which the user can implement models to simulate the system in question. The time-consuming nature of implementing models lead to the development of the GloMoSim library for wireless networks that is built on top of the PARSEC framework [33].

1.4 BACKGROUND SUMMARY

Currently there exists a wide range of choices of hardware, of network communication protocols, and of network setup and maintenance algorithms for sensor networks. While the requirements of the sensor network being designed can reduce the number of alternatives in each of the above three categories, there still exists a wide range of alternatives and potential choices in each. The developer must evaluate and weigh a number of different alternatives comparing benefits and drawbacks of each before reaching a decision.

Simulators, in various forms, currently exist that can simulate a sensor network. These simulators are sometimes difficult to learn, require substantial time to develop the simulation environment, and can require a significant amount of time to produce results. Because simulators require so much time to produce results, the number of different possible solutions that can be investigated is limited. Both OPNET and ns-2, two simulators commonly used to simulate sensor networks, originated as simulators for wired networks that have been extended to simulate wireless networks [34].

The large number of simulators available and a difficulty in guaranteeing repeatability increases the difficulty for comparison of results obtained from different simulators. Ideally, different simulators should present similar results, or trends for a given system. A study by Cavin, et.al. investigating the variation of results simulating a network on ns-2, GloMoSim, and OPNET showed that the results were significantly different [34]. The results obtained showed significant differences not only in values, but also in behavior of the network in general [34]. These differences lead Calvin, et.al. to conclude that simulations provide little improvement to the design process [34]. Given the variation in results of the simulators investigated, making accurate comparisons requires the developer to implement not only their work, but also the network(s) they wish to compare their solution to, in the simulator used to evaluate their solution. While possible, the amount of extra work and time needed to implement other solutions becomes prohibitive to thorough testing and evaluation of a particular solution.

While some analytic models exist, they are usually targeted at a specific protocol, algorithm, or some other area of interest in the field of sensor networks. Analytic models to model entire sensor networks are extremely limited in scope, often to the single network protocol

in question, and are not well suited to allow customization by users or to evaluate alternatives wishing to investigate variations of the original idea.

Analytical models should be faster to evaluate and scale better than simulated models, especially when looking at large networks (10,000+ nodes) or longer time frames (i.e. months or years). Most simulators currently in use have a discrete event simulation at the backend. For example, a leading discrete event simulator ns-2, scales poorly, thus limiting simulations to a networks with at most a few thousand nodes [34]. The use of a discrete event simulator at the core is problematic if a large number of messages are sent because the discrete event simulator must schedule each event and ensure delivery of events in the proper order. In sensor or RFID networks, the number of messages sent (and thus needing scheduling) often becomes very large. Such a model should execute significantly faster and scale well as the network size and the number of messages increases.

Chiasserini and Garetto describe an analytical model describing the energy consumption in sensor networks in which nodes can be put to sleep to conserve power [35]. This work limits the model of the node to that of the processor and the communication hardware, while the network is modeled as a queuing network in which all messages are received without error [35]. Such simplifications neglect critical facts such as how nodes receive messages when a node is asleep. An analytical model allowing modeling of individual components within nodes and the network connecting them using assumptions based on real world behavior and performance is needed.

The ability to compare two or more different networks accurately is needed to decide between different design alternatives. With a modeling approach, it is easier to verify if the implementation of the network follows the specifications of the network being investigated. Further, an analytical model is more transparent than elaborate code written for a simulator, allowing other researchers to quickly see and understand how the model was implemented and to determine if the implementation closely follows the specifications for the particular network evaluated. This increased transparency will increase the ability to compare results obtained for two or more different networks.

1.5 STATEMENT OF THE PROBLEM

The proposed research ***will provide a generic analytic modeling framework for sensor networks for thorough analysis at a variety of levels. This analytic model will be designed to provide solutions to the current limitations sensor network developers*** currently face. The initial goal of this research is to develop the structured modeling environment of sensor networks in their entirety earlier in the design cycle than is currently feasible. This will provide designers with additional knowledge enabling more informed design decisions to be made in the design phase of the sensor network. The ultimate goal is for the modeling framework to provide the mathematical model and topological basis for a tool during the specification phase (or very early phases) of the design phase to determine the best way to proceed to a particular analysis by comparing several alternatives and verifying initial concepts quickly and easily.

This dissertation will present the development of ***the framework for a mathematical structure (model) of the current state of recognized sensor networks***. First, a thorough analysis of the relevant literature will provide the basis for the current state of sensor networks. This analysis will provide the means to divide existing sensor networks into a number of generic classes that cover the entire field. Further, this information will allow identification of the basic components that make up a sensor or RFID network.

Second, a similar analysis of existing tools for analysis and simulation of these networks identifying the relevant characteristics that researchers and developers focus on will be presented in this dissertation. The tool summary will then be analyzed to extract the implicit and explicit structural components that appear to be the primary focus of the previous analyses. These two surveys and the classification of sensor networks will provide the basis for the ***framework for the mathematical structure (model) to be developed describing sensor networks***.

The initial structural formulation will then be tested to insure the inclusion of both the implicit and explicit features of sensor network analysis. Based on this research, the final dichotomy of sensor networks will be formulated, comprising the third contribution of this research.

The proposed modeling environment will allow expansion and customization providing structural hooks throughout the model elements. These hooks will allow a user to insert customized sub-models or to modify existing modeling structures. Customized or expanded sub-

models can be hung from these hooks. Using hooks, models can easily be created to study new network types, or modified to examine different possibilities of a given network type.

The model will allow the user to focus on a particular area of interest and allow the simplification of other components (i.e. only simulate the energy used by the sensors, and look at nothing else). Again, by using hooks at each of the modeling levels, the user will be free to select a detailed or a simplistic model at various levels. Further, the user will be able to add their own model components to the existing model. This framework provides the user with the freedom to decide and perform modeling with varying levels of detail for each of the top-level parameters.

This modeling framework will be used on some example networks and the results compared with results of previous work. Differences between the results will be discussed. The example cases will consist of networks based on the ISO 18000-7 RFID standard and on the IEEE 802.15.4/ZigBee standard.

The remainder of this dissertation is structured as follows. The base level entities from which sensor and RFID networks are constructed are presented in Section 2. In Section 3, sensor and RFID networks are broken down into specific classes to assist designers in determining the tasks of the topological entities. Section 4 introduces Markov processes and provides the basis for the operations with Markov processes used in this dissertation. Generic Markov processes for the base entities and the generic topological entity are described in Section 5. The basic communication graph concept that is used during the analysis of a network is described in Section 6. Section 7 presents methods to identify topologies in already existing networks and a method to use topologies to design a new network. The base topologies of a sensor or RFID network are illustrated in Section 8. The algorithm to identify topological entities is presented in Section 9. The ISO 18000-7 example case is presented in Section 10, and the ZigBee example case is presented in Section 11. The operations required to recalculate the energy consumed after altering the parameters in the method developed in this work are presented in Section 12, and Section 13 presents the conclusions.

2.0 COMPONENTS OF A SENSOR AND RFID NETWORK

Sensor and RFID networks contain hundreds or thousands of entities. The entities of sensor or RFID networks are composed of two principle components, nodes and sinks. The node is primarily concerned with the collection or storage of data or monitoring of the environment, and transmission of that data back to the outside world. The primarily concern of the sink is to provide a bidirectional link between the sensor network (network of nodes) and the outside world. The outside world consists of the user or controller, or a device (computer) that collects and processes the information generated by the sensor network. In most networks, the sink also takes on the role of network controller on either a local or a global level.

2.1 NODES

At the highest level, the node consists of five components, the energy supply (usually a battery), energy harvesting circuit(s), communications circuitry, a processor and a transducer (a sensor or an actuator). In a RFID network, a RFID tag acts like a node and may not have a transducer. The node interacts with the outside world in three different ways. The first interaction, communication with sinks or other nodes, is achieved by sending and receiving messages. Secondly, the node interacts with the outside world through use of the transducer, either by reading the current measurement from a sensor, or by altering the state of an actuator. Third, the node may harvest energy from the environment. It is possible to have multiple energy harvesting circuits in a single node (i.e. motion and solar energy harvesting circuits). The node contains communications circuitry enabling it to send and receive messages from the network. Finally, the node contains an on-board energy source supplying the power for operation of the node.

The top-level diagram and five top-level blocks will form the basis for the energy models for the nodes in all classes of networks investigated. The top-level block diagram of the node is shown in Figure 2.1.

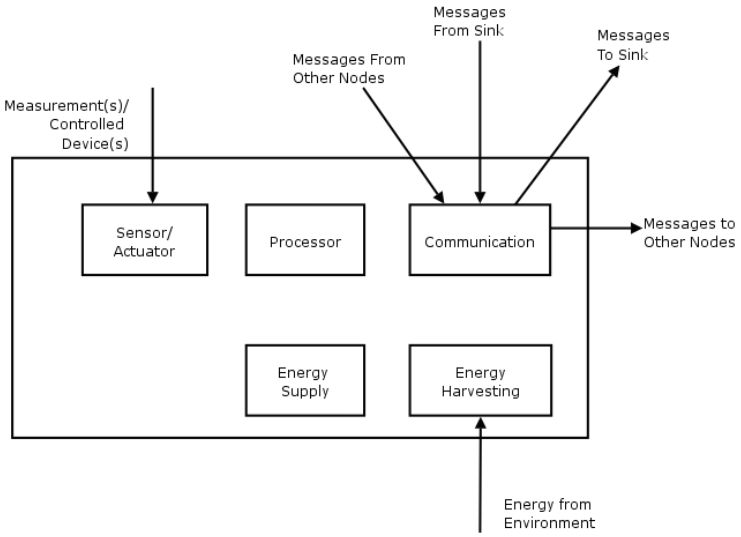

Figure 2.1: Block diagram of a node.

2.2 SINKS

The sink provides a link between the outside world and the network. There are two possibilities for the sink. The first, a type 1 sink, is just a node modified with the capability to communicate with the outside world. The type 1 sink can act as a node or as a sink and can change roles during operation depending on existing conditions. The second, a type 2 sink, is a sink designed specifically to be a sink. The only purpose of the type 2 sink is to provide a communications link between the network and the outside world. Therefore, the type 2 sink cannot change roles during operation to act as a node like the type 1 sink can.

The type 1 sink is a node, modified to provide the capability of communication with the outside world. The type 1 sink is used to represent entities in a sensor network that can switch roles to be a node or a sink as needed by the network. The ability to switch roles provides for

more flexibility for a network to respond to changing conditions. Entities of this type can be used to model networks employing a clustering strategy that rotates the role of cluster head within each cluster. The block diagram of the type 1 sink is shown in Figure 2.2.

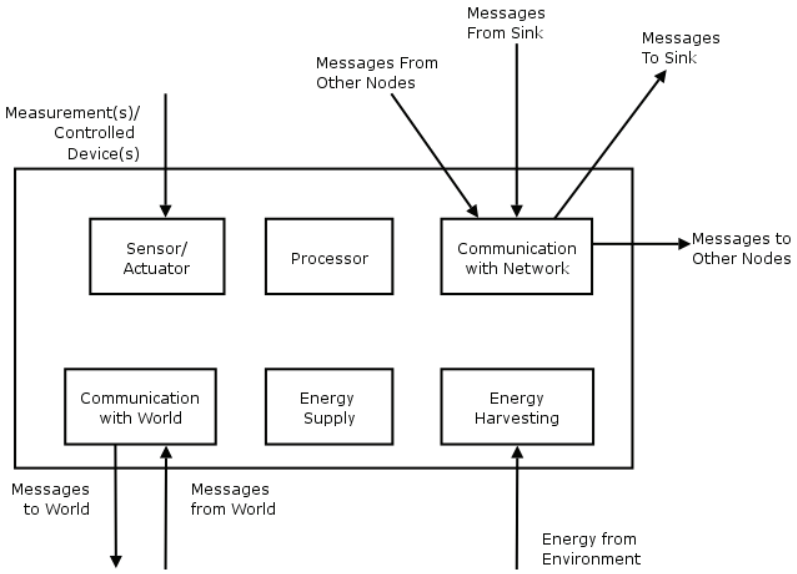

Figure 2.2: Type 1 sink node.

The type 1 sink must be capable of performing all the functions of a node (i.e. taking measurements) and of a sink (communicating with outside world). Therefore, the five top-level blocks of the node are also contained within the type 1 sink. The type 1 sink has an additional block, Communication with World, which provides the communications link between the sink and the outside world. This communication block is separate from the other existing communication block (Communication with Network) to account for possible differences between the communication method used for inter-network communication and for the communication link between the network and the outside world. When acting as a node, the Communication with World block is not used and the Sensor/Actuator block is used. Conversely, when acting as a sink the Sensor/Actuator block is not used and the Communication with World block is used.

The type 2 sink differs from the type 1 sink in that it is specifically designed to function as a dedicated sink. The block diagram of the type 2 sink is shown in Figure 2.3. The type 2 sink cannot switch roles and become a node as the type 1 sink can. Therefore the type 2 sink functions purely as a communications gateway to the outside world. Therefore, the type 2 sink has no capability to have a transducer (Sensor/Actuator) attached. The node and the type 2 sink have four of the top-level blocks in common (the Sensor/Actuator is not part of the type 2 sink). The type 2 sink has an additional block, the Communications with World block, providing the communications link between the sink and the outside world. This communication block is separate from the other existing communication block (Communication with Network) to account for possible differences between the communication method used for inter-network communication and for the communication link between the network and the outside world.

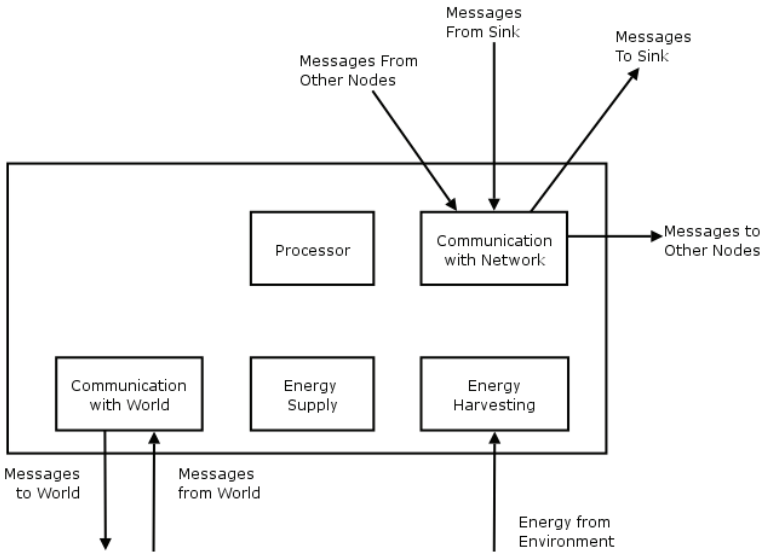

Figure 2.3: Type 2 sink node.

3.0 CLASSIFICATION OF SENSOR AND RFID NETWORKS

Existing sensor networks can be broken up into a number of different categories. Each category describes a particular type of wireless network. This determination of this classification will be the starting point for the development of a mathematical structure describing the energy consumption for each category of networks.

3.1 CLASSES OF SENSOR AND RFID NETWORKS

Mobility of the node and/or sink is a critical factor in classifying a particular network into a category. Networks with only stationary entities are significantly easier to model, as the network topology changes only with the addition or deletion of entities in the network. Conversely, networks with entities with mobile capabilities present an added problem that the network topology can change without the addition or deletion of entities to the network. The initial division of sensor networks is based on whether or not the network contains entities with mobility or contains only stationary entities and is shown in Figure 3.1.

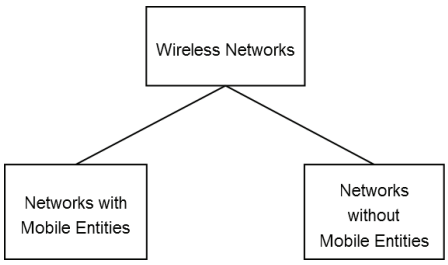

Figure 3.1: Top-level breakdown of wireless networks.

For networks containing mobile entities, the energy consumed for the movement of each mobile entity in the network must be taken into account. A mathematical structure will be developed to account for this energy. A mobile entity, when moving can be viewed as being deleted at the starting position and added at the destination if the mobile entity does not communicate during transit. If the mobile entity must communicate during transit, the model must account for that communication. Therefore, the class of networks with mobility requires different structures depending on the requirements of the mobile entities and is further divided as shown in Figure 3.2 below.

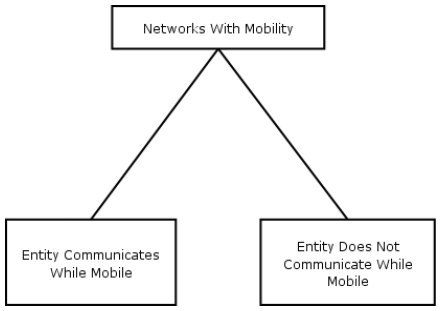

Figure 3.2: Sub-division of networks containing mobile entities.

With the initial division of sensor and RFID networks into two classes, the first containing all networks without mobile entities, and the second containing all networks with mobile entities, further subdivision of each class is possible based on the requirements and tasks of each network. The class of networks without mobile entities can be subdivided into three categories, shown below in Figure 3.3. These three sub-divisions cover all networks without mobile entities.

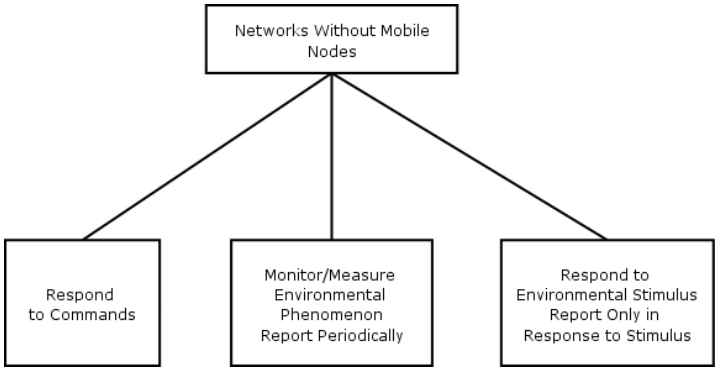

Figure 3.3: Sub-division of the networks containing only stationary entities.

The first sub-class, Respond to Commands, contains all networks where the nodes take action based on the commands sent by the sink or controller. In this type of network, the nodes do not communicate without first receiving a command from the sink/controller. Further, the nodes communicate only with the sink, there is no node-to-node communication other than the possible relaying of messages between the sink and a particular node (in either direction).

The second sub-class, Monitor/Measure Environmental Phenomenon Report Periodically, consists of networks in which the nodes must monitor an environmental phenomenon and periodically communicate those readings back to the sink or controller. One example, of such a sensor network is a network that must monitor and report the temperature of a room once a second. In this type of network nodes periodically take a reading and send that reading to the sink. At the top-level, the communication is from node to sink possibly using other nodes as intermediary relays. The nodes will send their information to the sink periodically and do not need to wait for the sink to request the information, as in the first case.

The third sub-class, Respond to Environmental Stimulus Report only in Response to Stimulus, contains all networks that are responsible for monitoring the environment and reporting information to the sink only when specific conditions are met. In this class of networks, readings are taken locally by the nodes and messages are sent only when the phenomenon being monitored exceeds some threshold value. A network used by the military to track targets on a battlefield is one example of this type of network. In such a network, readings would only be taken and messages sent to the sink when a target is detected. If no targets are

detected no readings would be taken, or messages sent to the sink. This network is similar to the second class, as the nodes send information without the sink explicitly requesting it. It extends the second class to the class of networks that respond to environmental stimulus rather than report data at regular intervals.

With wireless communication, two entities can communicate directly with each other only if they are both within range of each other, as shown in Figure 3.4. However, if the two entities in question are not within range, communication is still possible, but an intermediary entity must be used to relay the message as shown in Figure 3.5. The case of relaying using a single entity, as shown in Figure 3.5, can be extended the case where multiple relay entities are used. This case is shown in Figure 3.6 where multiple entities must relay the message. Both of the preceding examples (Figure 3.5 and Figure 3.6) illustrate multiple hop communication. The final sub-division of each of the three network classes presented above is based on the type of communication used, direct or multiple hop communication.

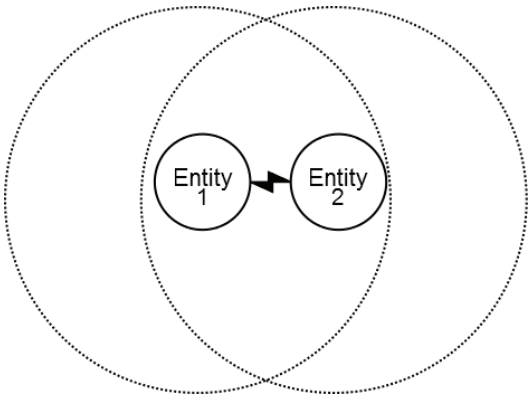

Figure 3.4: Two entities (Entity 1 and Entity 2) can communicate directly because they are within range of each other.

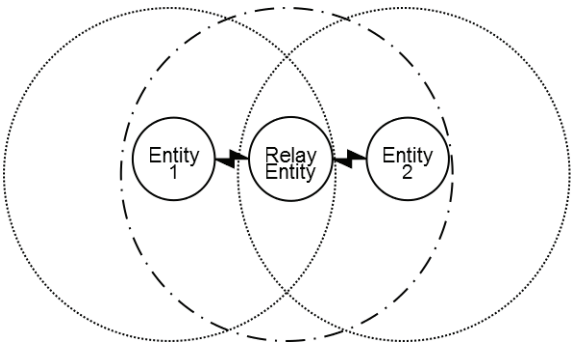

Figure 3.5: Two entities (Entity 1 and Entity 2) are not within range of each other, messages must be relayed by the Relay Entity.

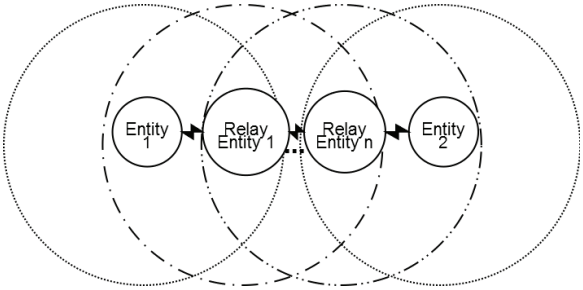

Figure 3.6: Two entities (Entity 1 and Entity 2) are not within range of each other, messages must be relayed by multiple Relay Entities.

The Responds to Commands sub-class of sensor and RFID networks consists of those networks that are a master/slave architecture. In these networks, the sinks are the masters and the nodes are the slaves. Nodes only respond to commands and inquiries sent from a sink. Nodes do not initiate communication by sending commands or inquiries and only communicate with other nodes in order to relay a message to a node from a sink or to the sink from a node.

The Responds to Commands network class can be further divided into smaller classes based on the communication links, multiple or single hop, between the sinks and the nodes. Based on the number of combinations of multiple hop and single hop communication for the node to sink and sink to node links there are four sub-classes of Responds to Commands networks. These four subclasses are shown in Figure 3.7.

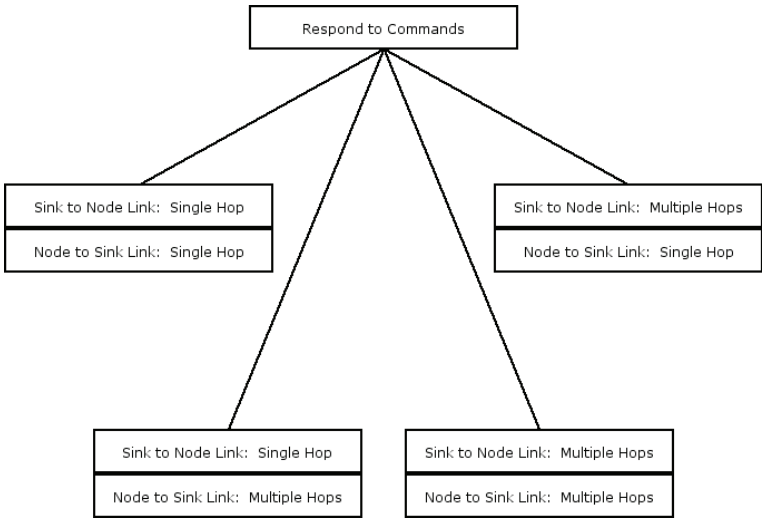

Figure 3.7: Division of the Respond to Commands class of networks based on the communication schemes.

The first sub-class (upper left child) contains all networks in where both the sink and the nodes communicate with each other without going through intermediary nodes in a single hop. The second sub-class (bottom left child) is comprised of all networks in which the sink(s) communicate with all nodes directly, in a single hop, while there are some nodes that cannot communicate with the sink directly, in a single hop. Those nodes unable to transmit a message to the sink in a single hop must use other nodes to relay their messages to the sink, requiring multiple hops for their message to reach the sink. The third sub-class (bottom right child) consists of all networks in which some of the sinks and some of the nodes require multiple hops to communicate with each other. The fourth sub-class (top right child) consists of networks in which the sink requires multiple hops to communicate with some nodes, while all nodes can communicate with the sink in a single hop.

The Monitor/Measure Environmental Phenomenon Report Periodically class represents the type of network where the nodes periodically report information back to the sinks in the network and again can be divided into four classes based on node/sink communication links as shown in Figure 3.8. The measurements are reported to the sink periodically. In this network, the majority of the communication is from the nodes to the sink. The node to sink communication variations account for two of the four sub-classes where the node to sink link is

either limited to a single hop or can be multiple hops. To make this type of network more general, the ability for the sinks to issue commands to the network is added. Some examples of these commands could be a request for the current readings or to control or initiate configuration of the network. The communication links between the sink and the nodes can be either limited to a single hop, or require multiple hops, providing the remaining two subclasses of this type of network.

The Respond to Environmental Stimulus Report only in Response to Stimulus class of networks is very similar to the Monitor/Measure Environmental Phenomenon Report Periodically class. The key difference between the two types is in the reason the nodes report information to the sinks. In the Monitor/Measure Environmental Phenomenon Report Periodically network type, the nodes report readings to the sinks at regular intervals or in response to a command issued by a sink. In Respond to Respond to Environmental Stimulus Report only in Response to Stimulus type networks the nodes only report information to the sinks if the phenomenon monitored surpasses a certain threshold, and, therefore, the sending of information is not necessarily regular. For example, the threshold could be a temperature set point, above which a central alarm must activate, or the threshold could be the detection of a target in a military terrain monitoring system.

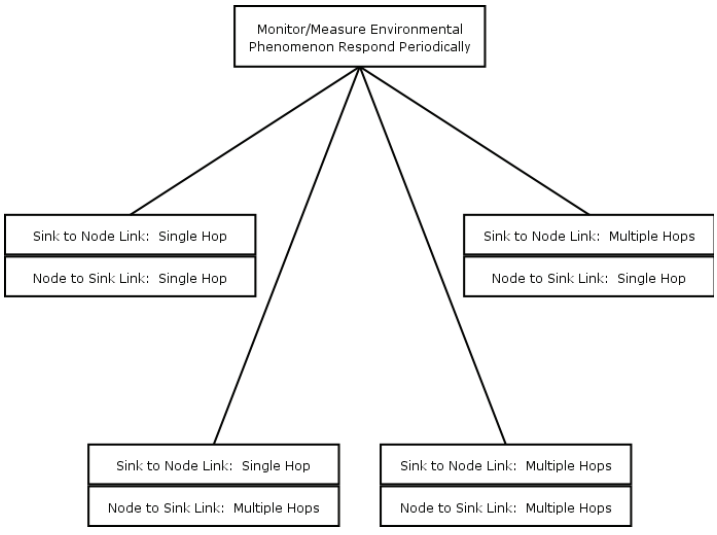

Figure 3.8: Division of the Monitor/Measure Environmental Phenomenon class of networks based on the communication schemes used.

The division of the Respond to Environmental Stimulus Report only in Response to Stimulus class is shown in Figure 3.9. Two of the four subclasses account for single and multiple hop node to sink paths. To provide support for commands to be issued into the network through the sinks, two subclasses are provided to model different sink to node paths. The remaining two subclasses account for single and multiple hop sink to node paths.

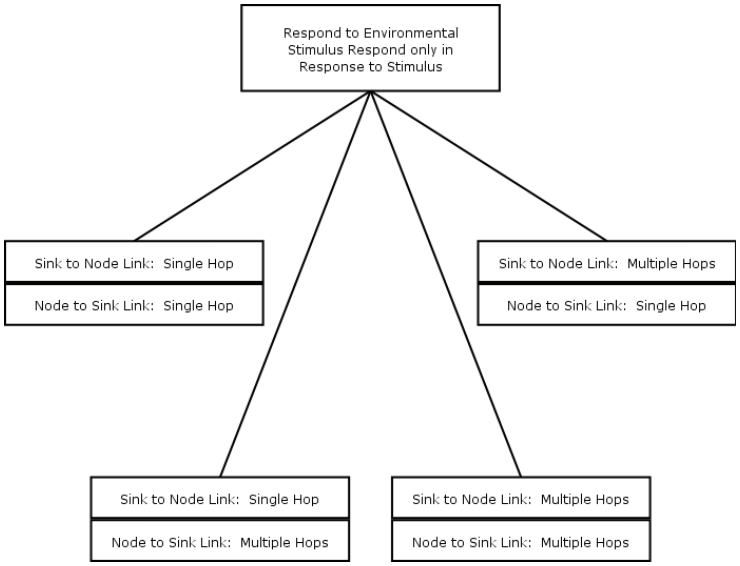

Figure 3.9: Division of the Respond to Environmental Stimulus class of networks based on the communication schemes used.

4.0 MARKOV PROCESS

Markov processes are an efficient way to analyze systems. Markov processes can be used to model a system when the next state of the system depends on the current state of the system, and not on the past states [36]. A Markov process consists of a set of states, a set of transition probabilities, and a set of reward values. The states represent different conditions in a system. The transition probabilities provide the likelihood, or probability, that the system will transition from state i to state k. The reward values denote the gain or profit, or the loss or penalty from moving from state i to state k.

It is often convenient to illustrate a Markov process using a specialized graph, or state diagram. In a state diagram, the vertices correspond to system states, and the edges correspond to transitions between two states. Each edge is assigned a value representing the probability of moving from state i to state k (i and k may be the same state). The transition probability is contained in the interval [0, 1]. For each state, i, the sum of the transition probabilities of all edges originating from state i must be 1. Similarly, the sum of all transitions leaving (beginning at) state i must equal 1.

A simple temperature sensor will be analyzed using a Markov process. The simple temperature sensor must take a temperature reading every minute and send the current reading to a system controller. A three state Markov process can describe the operation of the temperature sensor. Each state represents one of the three tasks that the temperature sensor must perform. First, the temperature sensor must read the on-board temperature sensor. Next, the current temperature reading must be transmitted back to the system controller. Finally, the temperature sensor must wait until it is time to take the next reading. The three states will be referred to as READ (RD), XMIT (TX), and WAIT (WT). The state diagram of the Markov process describing the temperature sensor with transition probabilities is shown in Figure 4.1.

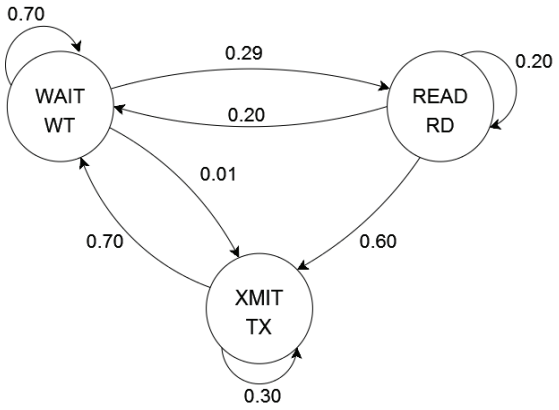

Figure 4.1: State diagram of the Markov process for the simple temperature sensor.

At each interval, the system is in exactly one state. The system may change states, moving from state i to state k with probability, P_{ik}, or may stay in state i, with probability, P_{ii}. The transition probabilities do not change with time in a stationary Markov processes [36]. A Markov process will have N states. These probabilities can be grouped into an N by N square matrix. The temperature sensor example consists of three states and the 3x3 probability matrix for this process as illustrated in (4-1).

$$P = \begin{bmatrix} P_{WTWT} & P_{WTRD} & P_{WTTX} \\ P_{RDWT} & P_{RDRD} & P_{RDTX} \\ P_{TXWT} & P_{TXRD} & P_{TXTX} \end{bmatrix} \tag{4-1}$$

The probability matrix (4-1 has an entry for all nine possible edges in the state diagram in Figure 4.1. However, not all nine edges are present in Figure 4.1. Those edges that are not shown in Figure 4.1 have a transition probability of 0. The probability matrix with transition probabilities for the simple temperature sensor is shown in (4-2).

$$P = \begin{bmatrix} 0.70 & 0.29 & 0.01 \\ 0.20 & 0.20 & 0.60 \\ 0.70 & 0 & 0.30 \end{bmatrix} \tag{4-2}$$

A set of rewards is associated with the transitions from each state i, to each state k. The reward denotes the profit, or loss, associated with transitioning from state i to state k. As with the transition probabilities, the rewards can be represented in an N by N square matrix as shown in (4-3).

$$R = \begin{bmatrix} R_{WTWT} & R_{WTRD} & R_{WTTX} \\ R_{RDWT} & R_{RDRD} & R_{RDTX} \\ R_{TXWT} & R_{TXRD} & R_{TXTX} \end{bmatrix} \tag{4-3}$$

In the case of sensor networks the reward will be the energy consumption of state k. The value is the sum of the product of all transitions and the rewards associated with each transition.

The system is in exactly one state in each interval. The probability of being in a state, i, during the nth interval is denoted as, $\pi_i(n)$ [37]. The vector π illustrated in (4-4) is defined as the collection of the π_i's.

$$\pi = \begin{bmatrix} \pi_1 & \pi_2 & \dots & \pi_N \end{bmatrix} \tag{4-4}$$

where the system in question has N different states. Since each of the $\pi_i(n)$'s denotes a probability that the system is in state i, during interval n, the sum of all π_i's must be 1. Hence,

$$\sum_{i=1}^{N} \pi_i = 1 \tag{4-5}$$

The probability that the system will be in state k during the next interval, the (n+1)th interval, if the system is in state i during the nth interval is simply the product of that the probability the system is in state i during the nth interval and the transition probability p_{ik} during the nth interval.

The system can transition into state k from any of the N states in the system. Therefore the probability of being in state k during the (n+1)th interval without regard to what state the system was in during the nth interval is simply the sum of the probabilities of entering state k

from each of the N system states. The probability of being in state k during the (n+1)th interval is given by

$$\pi_k(n+1) = \sum_{i=1}^{N} \pi_i(n) p_{ik} \tag{4-6}$$

In matrix form the π vector can be calculated using the previous π vector and the probability matrix P giving the probability that the system in is each of the N states during the (n+1)th interval

$$\pi(n+1) = \pi(n)P \tag{4-7}$$

To obtain the π vector for the nth interval the probability matrix must be raised to the nth power because

$$\begin{aligned} \pi(1) &= \pi(0)P \\ \pi(2) &= \pi(1)P = \pi(0)PP = \pi(0)P^2 \end{aligned} \tag{4-8}$$

The term π(0) is the starting vector containing the system state before the first interval. The above process continues until π(n) can be calculated using the equation below

$$\pi(n) = \pi(0)P^n \tag{4-9}$$

Thus, it is possible to determine the probabilities that the system is in a given state at any interval using (4-9). However, for systems with large numbers of states (large N) the probability matrix will grow accordingly and continually multiplying the previous π-vector and the P matrix to obtain the current π-vector becomes tedious.

A chain in a Markov process is a set of states of the Markov process. Chains are important to the analysis of Markov processes. Figure 4.2 shows a Markov process with two chains consisting of the states {1} and {2, 3, 4}.

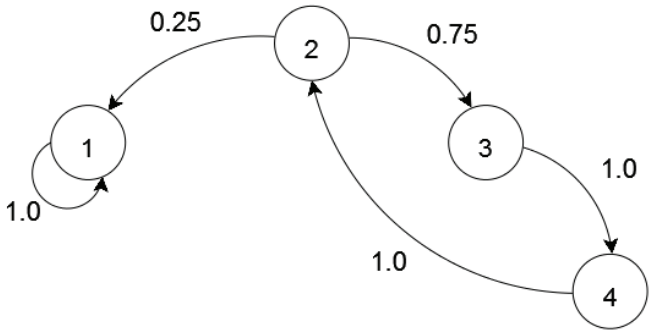

Figure 4.2: Example Markov process with two chains.

The first chain consisting of only state 1 is a recurrent chain because once state 1 is entered the process can never leave state 1. The chain consisting of states {2, 3, 4} is not recurrent because in the future it is possible to enter state 1 from state 2. No matter what state the Markov process in Figure 4.2 starts in, after a long time it will eventually become trapped in the single recurrent chain, in this example, state 1.

All Markov processes have at least one recurrent chain but may have more than one recurrent chain [37]. An example Markov process with two recurrent chains is shown in Figure 4.3.

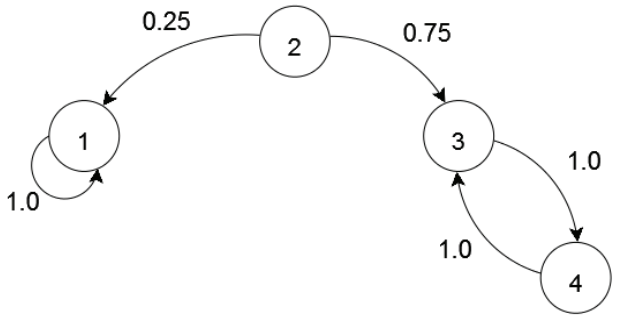

Figure 4.3: Example Markov process with two recurrent chains.

The first recurrent chain in Figure 4.3 consists of state 1, while the second recurrent chain consists of the states {3, 4}. State 2 does not belong to either chain and is a transient state since after a long time (in the above example after the first transition) the process can not be in that state [37]. Unlike the Markov process shown in Figure 4.2, the process illustrated in Figure 4.3 will be in one of two chains after one transition. The chain that the process in Figure 4.3 is trapped in depends on the starting state. For example if the process is started in state 1, then it will never leave state 1. Similarly, if the process is started in either state 3 or state 4, the process will never leave the recurrent chain made up of states 3 and 4. However, if the process is started in state 2, after one transition the process will become trapped in one of the two recurrent chains based on the transition probabilities from state 2 to either chain. In this example, it is more likely that the process will become trapped in the recurrent chain containing states 3 and 4. The Markov process shown in Figure 4.1 has only one recurrent chain.

4.1 STEADY STATE RESPONSE

In some systems, the state probabilities remain constant after some period of time. This state probability vector is called the absolute probability vector and is defined as

$$\pi = \lim_{n\to\infty} \pi(n) \tag{4-10}$$

In order for the absolute probability vector to exist, the system must have certain characteristics. The absolute probability vector exists when the probability vectors for each possible starting state converges to the same probability vector after a long time. If the π-vector found using (4-10) is identical regardless of what state the system starts in then the absolute probability vector exists. If the probability of transitioning into state j is the same from state i, P_{ij}, for all states i in the system the absolute probability vector exists.

The existence of the absolute probability vector simplifies the analysis of the system because after a long time the π-vector no longer changes and does not need to be calculated for

future intervals. Using the absolute probability vector greatly simplifies the process and reduces the number of computations required to evaluate such a system.

The absolute probability vector can be found using two different methods. The first method is simply to raise the probability matrix, P, to a high power. When the probability matrix, P, is raised to a high power, each row of P is equal to the absolute probability vector, π. If the Markov process has a single recurrent chain, then all rows of P raised to the high power will be identical. However, if the Markov process has more than one recurrent chain, then the ith row of P raised to a high power is the absolute probability vector that is obtained if the process is started in state i.

The second method to find the absolute probability vector, π, is to solve a set of N+1 linear simultaneous equations. When the absolute probability vector, π, exists the following equations holds

$$\pi = \pi P \tag{4-11}$$

From (4-11), N equations can be obtained. There are N unknown π_i elements contained in π, at least one more equation is required to find a unique solution for the π_i's. Since the absolute probability vector, π, contains the probability of being in each state in the steady state the sum of the elements of π is 1. This forms the (N+1)th equation that allows a unique solution to be obtained for π. The N+1 equations are illustrated below

$$\begin{aligned} &[\pi_1 \quad \pi_2 \quad \ldots \quad \pi_N] = [\pi_1 \quad \pi_2 \quad \ldots \quad \pi_N]P \\ &\pi_1 + \pi_2 + \ldots + \pi_N = 1 \end{aligned} \tag{4-12}$$

Solving the N+1 equations for the N unknown π_i values yields a unique solution for the absolute probability vector, π. The absolute probability vector exists if the Markov process is irreducible, aperiodic, and positive persistent [36].

If the Markov process is irreducible, aperiodic, and positive persistent the absolute state probability vector exists, and the above properties of the π-vector hold [36]. The above properties of the π-vector also hold if the Markov process contains some transient states as long

as the aperiodic property holds for all persistent states, and there is only one irreducible set of periodic states [36].

An irreducible Markov process is a process that for all pairs of states it is possible to go from state j to state k in n moves, and also from state k to state j in m moves [36]. The number of moves, n, and, m, may be different [36]. More simply, an irreducible Markov process contains only one recurrent chain [36]. Hence, the example Markov process in Figure 4.3 has two recurrent chains and is not irreducible. Conversely, the Markov process in Figure 4.2 has only one recurrent chain and is irreducible.

The period of a state in a Markov process is defined by the possible discrete times that a state j is returned to when starting at state j [36]. For a state to be periodic two conditions must hold. The first condition is that the state, k, can only be entered only on the $m*d^{th}$ move, where m can vary among the positive integers and d is particular positive integer [36]. For all other moves not equal to m*d, the probability of entering state k is 0 [36]. The value of d is the greatest integer such that it is possible to enter state k on the $m*d^{th}$ move [36]. The state k is aperiodic if the value of d is 1 [36]. An aperiodic Markov process is simply a Markov process where every state is aperiodic. An example periodic Markov process is shown in Figure 4.4.

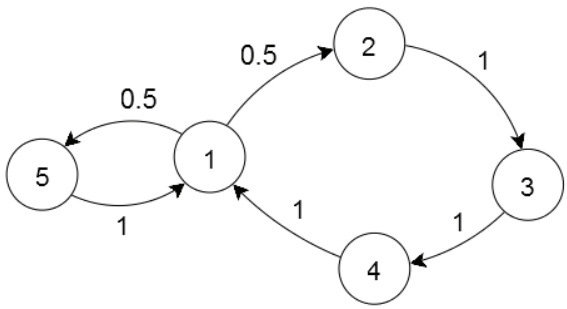

Figure 4.4: Example of a periodic Markov process.

In the example Markov process Figure 4.4, it is possible to return to state 1 when starting from state 1 at times, 2, 4, 6, 8 …. The value of d could be either 1 or 2 since for both values it is possible to find a value of m such that one of the return times to state 1 can be expressed as

m*d. However, the definition of periodic places the restriction on d that it must be largest number such that all return times can be expressed as m*d. Thus, the value of d must be 2 and state 1 is said to have a period of 2. The period of a state is simply the value of d for that state. The period of the remaining states can be found in a similar manner. The value of d is simply the greatest common devisor (producing a positive integer) of the return times. The example Markov process in Figure 4.5 is aperiodic because the only possible values of d for all states is 1.

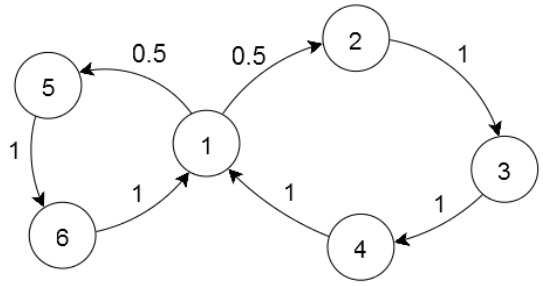

Figure 4.5: Example of an aperiodic Markov process.

It is easy to verify that states 1, 2, 3, 4, 5, and 6 are aperiodic because the return times for those states consist of both even and odd times. It is impossible to find a common divisor of an even and odd return time other than 1 so states 1, 2, 3, 4, 5, and 6 are aperiodic since d is 1 for those states. For example, the first three return times to state 1 starting in state 1 are 3, 4, 6 and the only possible value of d, that allows these three return times to be generated with m*d is 1. The remaining states can be shown to have periods of 1 using similar reasoning. If the probability P_{ii} for state i is greater than zero, state i is periodic [36].

A positive persistent state is a state in which it is possible to return to that state from that state. The probability of returning to a state, j, when in state, j, at a given time, n, is denoted as $f_{jj}^{(n)}$. If the sum of the f_{jj}'s for all n is 1, then state j is a persistent state [36]. The expected return time, μ_j, is calculated by multiplying the time step, n, by the probability of returning to state j at time n, fjj(n) and is shown in (4-13) [36].

$$\mu_j = \sum_n^{\infty}\left(n * f_{jj}^{(n)}\right) \tag{4-13}$$

The expected return time of a positive persistent state is finite [36]. A null persistent state has a expected return time of infinity [36].

Recall that for the properties of the π-vector defined above to exist, that the Markov process must be irreducible, aperiodic, and positive persistent [36]. Determining if a Markov process meets the criteria for the absolute probability vector to exist can be difficult. However, several efficient methods have been developed [36, 38, 39]. The column-sum method described by Isaachson and Madsen is one such method and that particular algorithm is used in this research to verify that the Markov processes used for the example cases are irreducible, aperiodic, and positive persistent.

4.2 ANALYSIS OF MARKOV PROCESS WITH Z-TRANSFORM

When the probability matrix P has large dimensions, computing P^n or solving the system of equations given in (4-12) to obtain the absolute probability vector, π, can be tedious and time consuming. Obtaining the π-vector can be simplified by performing operations in the z-domain. Because the Markov processes in this work are evaluated in a discrete time base they may be evaluated in the z-domain. A discrete time function is transformed into a function in the z-domain as follows, where n represents the discrete time step, f(n) is the discrete function, and f(z) is the function corresponding to f(n) in the z-domain.

$$f(z) = \sum_{n=0}^{\infty} f(n) z^n \tag{4-14}$$

It is also possible to move from the z-domain to the discrete domain by using the appropriate inverse z-transform.

Converting (4-9) from the discrete domain into the z-domain requires evaluating the following

$$Z\{\pi(n)\} = Z\{\pi(0)P^n\} \tag{4-15}$$

Where, $Z\{\ldots\}$ denotes the z-transform. The identity matrix, I, is equivalent to a scalar 1 in terms of mathematical operations with respect to matrices. A well know z-transform of a function α^n is

$$Z\{\alpha^n\} = \frac{1}{1-\alpha} \tag{4-16}$$

Modifying (4-16) for matrices results in

$$Z\{A^n\} = (I - Az)^{-1} \tag{4-17}$$

where I is the identity matrix and A is any matrix. Here the probability matrix, P, will be substituted for the matrix, A, in (4-17). Applying this to (4-15) yields [40],

$$Z\{\pi(n)\} = \sum_{n=0}^{\infty} \pi(0)P^n z^n = \sum_{n=0}^{\infty} \pi(0)(Pz)^n = \pi(0)\sum_{n=0}^{\infty}(Pz)^n = \pi(0)(I - zP)^{-1} \tag{4-18}$$

To obtain $\pi(n)$ from (4-18) the inverse z-transform of $Z\{\pi(n)\}$ must be evaluated,

$$\pi(n) = \pi(0)Z^{-1}\{(I - zP)^{-1}\} \tag{4-19}$$

Thus, $\pi(n)$ can be computed using (4-19). Computing $\pi(n)$ using (4-19) requires computing $(I-zP)^{-1}$, converting the P matrix to the z-domain, and converting the result back to the discrete domain. Both the z-transform and the inverse z-transform have been heavily studied and many conversion tables to convert functions between the z and discrete domains exist [37, 40-42]. In some cases using (4-19) to determine $\pi(n)$ is simpler than either (4-9) or (4-12). Finishing the evaluation of (4-19) yields,

$$\pi(n) = \pi(0)H(n) \tag{4-20}$$

where H(n) is the inverse z-transform of $(I\text{-}zP)^{-1}$. The matrix, H(n), contains a steady state component, that does not change with time and a transient component that disappears after a time for aperiodic processes [37]. The column sum method described by Isaacson and Madsen can be used to determine if the Markov process is aperiodic or not [36]. Hence, H(n) can be written as,

$$H(n) = S + T(n) \tag{4-21}$$

Where, S, denotes the steady state component of H(n), and T(n) denotes the transient component of H(n). For periodic Markov processes the transient term T(n) does not disappear for large n, and the steady state response, S, represents the probability of being in a given state at time, n [37]. The steady state component is related to the absolute state probability vector, π, discussed in Section 4.1 [37]. If the Markov process is irreducible, aperiodic, and positive persistent then the transient term, T(n), is 0 and π(n) only depends on S, simplifying the calculation of π(n).

4.3 ANALYSIS OF THE SIMPLE TEMPERATURE SENSOR MARKOV PROCESS

The analysis of the Markov process developed earlier to describe a simple temperature sensor is presented in this section. Recall that the simple temperature sensor can be in one of three states, wait, read, xmit. The state diagram repeated in Figure 4.6 is below. The probability matrix for the simple temperature sensor is repeated in (4-22).

$$P = \begin{bmatrix} 0.70 & 0.29 & 0.01 \\ 0.20 & 0.20 & 0.60 \\ 0.70 & 0 & 0.30 \end{bmatrix} \tag{4-22}$$

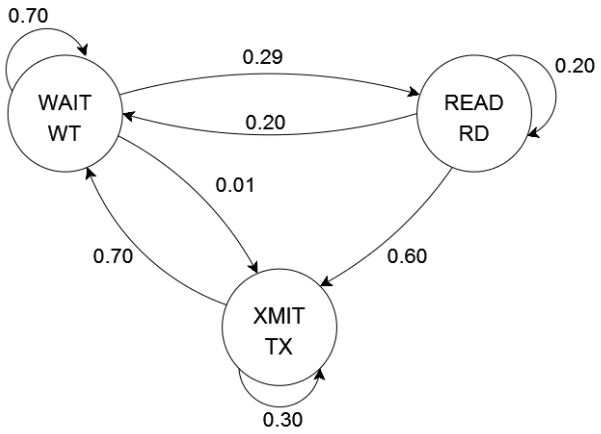

Figure 4.6: State diagram of the Markov process for the simple temperature sensor.

To determine if the above matrix, P, is aperiodic, the column-sum method will be used. The first step is to create an adjacency matrix, A, for the probability matrix P. Each element of A is either a 1 or a 0. Element A_{ij} is a 1 if the probability of transitioning from state i to state j is greater than zero, otherwise A_{ij} is a 0. The adjacency matrix A for the matrix P in (4-22) is shown below.

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix} \tag{4-23}$$

The adjacency matrix, like P, is an N by N matrix. If any column of A sums to N then the Markov process is aperioidic, irreducible, and positive persistent [36]. If no column sums to N then the matrix A is raised to the 2^kth power where k starts at one and k is increased by one each time no column sums to N [36]. If no column sums to one when,

$$2^k > \frac{N(N-1)}{2} \tag{4-24}$$

the Markov process is not aperiodic, irreducible, and positive persistent [36].

The first and third columns of matrix A shown in (4-23) sum to 3 and N equals 3 for this example. Therefore, using the column sum method on the adjacency matrix generated from the probability matrix in (4-22) the Markov process is determined to be aperiodic [37]. Because the Markov process is aperiodic, the transient component of H(n) will vanish for large n, and the absolute state probabilities can be obtained.

To obtain the absolute state probabilities the following system of equations can be solved.

$$\begin{aligned}&(\pi_1 \quad \pi_2 \quad \pi_3)=(\pi_1 \quad \pi_2 \quad \pi_3)*\begin{bmatrix}0.70 & 0.29 & 0.01\\ 0.20 & 0.20 & 0.60\\ 0.70 & 0 & 0.30\end{bmatrix}\\ &\pi_1+\pi_2+\pi_3=1\end{aligned} \tag{4-25}$$

Multiplying out the π-vector and P matrix and rearranging terms yields.

$$\begin{aligned}&0.30\pi_1-0.20\pi_2-0.70\pi_3=0\\ &-0.29\pi_1+0.80\pi_2=0\\ &-0.01\pi_1-0.60\pi_2+0.70\pi_3=0\\ &\pi_1+\pi_2+\pi_3=1\end{aligned} \tag{4-26}$$

Putting the above four simultaneous linear equations into matrix format yields

$$\left[\begin{array}{ccc|c}0.30 & -0.20 & -0.70 & 0\\ -0.29 & 0.80 & 0 & 0\\ -0.01 & -0.60 & 0.70 & 0\\ 1 & 1 & 1 & 1\end{array}\right] \tag{4-27}$$

Solving (4-27) yeilds

$$\begin{aligned}&\pi_1=0.5926\\ &\pi_2=0.2148\\ &\pi_3=0.1926\end{aligned} \tag{4-28}$$

Because the system is aperiodic, the π-vector could also have been determined by raising the P matrix to a high power, say 200.

$$P^{200} = \begin{bmatrix} 0.70 & 0.29 & 0.01 \\ 0.20 & 0.20 & 0.6 \\ 0.70 & 0 & 0.30 \end{bmatrix}^{200} = \begin{bmatrix} 0.5926 & 0.2148 & 0.1926 \\ 0.5926 & 0.2148 & 0.1926 \\ 0.5926 & 0.2148 & 0.1926 \end{bmatrix} \quad (4\text{-}29)$$

As can be seen the rows of the resulting matrix in (4-29) equal to each other and to solution found by solving the set of four equations. Therefore, either method will work to find the π-vector for an aperiodic system. However, due to the large amount of computations needed to raise P to a high power it is sometimes quicker to solve the system of equations for the π-vector.

The reward matrix, R, defines the reward for each transition. In a financial situation, the reward represents a profit or a loss occurring as a result of a transition. In this work the reward represents the energy consumed during a particular transition. The reward matrix for this example will have the following form.

$$R = \begin{bmatrix} R_{WTWT} & R_{WTRD} & R_{WTTX} \\ R_{RDWT} & R_{RDRD} & R_{RDTX} \\ R_{TXWT} & R_{TXRD} & R_{TXTX} \end{bmatrix} \quad (4\text{-}30)$$

For this example, the specific rewards denote the energy consumed in mW (milliwatts) for each transition in the Markov process describing the simple temperature sensor. The energy consumption for each transition of the Markov process describing the simple temperature sensor is shown in (4-31).

$$R = \begin{bmatrix} 2 & 6 & 3 \\ 5 & 5 & 17 \\ 4 & 0 & 17 \end{bmatrix} \quad (4\text{-}31)$$

The total possible reward that is possible during the next transition, when in a state k, is denoted by, q_k. The quantity, q_k, represents the total reward possible when a transition is made from state k to another state [37, 40].

$$q_k = \sum_{j=1}^{N} \left(p_{kj} * r_{kj} \right) \tag{4-32}$$

The column vector q denotes the total possible reward in the next transition from each of the N states in the Markov process and is composed of the q_k's found in (4-32).

$$q = \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_N \end{bmatrix} = \begin{bmatrix} \sum_{j=1}^{N} \left(p_{1j} * r_{1j} \right) \\ \sum_{j=1}^{N} \left(p_{2j} * r_{2j} \right) \\ \vdots \\ \sum_{j=1}^{N} \left(p_{Nj} * r_{Nj} \right) \end{bmatrix} \tag{4-33}$$

In an aperiodic, irreducible, and positive persistent Markov process, the transient term, T(n), is 0 and all the rows of S are equal to the π-vector. The reward for a given Markov process, g, is found by multiplying the probability that the process is in a state k by the total possible reward for state k, for all N states.

$$g = \pi q \tag{4-34}$$

The quantity, g, represents the average energy consumption for the Markov process for each transition. The energy consumption of the Markov process for n is found using (4-35).

$$E_{Total} = n \cdot g \tag{4-35}$$

Where n is the number of transitions investigated and g is the average energy consumption for one transition. Transitions represent entire tasks, or sub-tasks that make up a

larger task. Converting the number of transitions, n, into a time requires that the average time per transition, t_A, be obtained. Often a single transition represents completion of tasks and sub-tasks on multiple separate entities. First, a time matrix, T_A, is created containing the average time for all sub-tasks represented in a single transition,

$$T_A = \begin{bmatrix} t_{11} & t_{12} & \dots & t_{1N} \\ t_{21} & t_{22} & \dots & t_{2N} \\ \dots & \dots & \dots & \dots \\ t_{N1} & t_{N2} & \dots & t_{NN} \end{bmatrix} \tag{4-36}$$

Each entry, t_{ij}, in, T_A, represents the average time of the operation of all entities performing a task for the i to j transition. It is important that all entities, t_{ij}, in, T_A, are in the same time units. The average time per transition, t_A, should be weighted to those transitions that happen more frequently to obtain better accuracy in the solution. To appropriately weight, t_A, to those transitions the average time per transition q-vector, q_A, is found,

$$q_A = \begin{bmatrix} q_{A1} \\ q_{A2} \\ \vdots \\ q_{AN} \end{bmatrix} = \begin{bmatrix} \sum_{j=1}^{N} \left(p_{1j} * t_{1j} \right) \\ \sum_{j=1}^{N} \left(p_{2j} * t_{2j} \right) \\ \vdots \\ \sum_{j=1}^{N} \left(p_{Nj} * t_{Nj} \right) \end{bmatrix} \tag{4-37}$$

With, q_A, the weighted average time per transition, t_A, is simply,

$$t_A = \pi q_A \tag{4-38}$$

Once, t_A, is calculated the number of transitions is found by simply dividing the desired length of time to investigate, L_{Time}, by, t_A,

$$n = \frac{(L_{Time})}{t_A} \tag{4-39}$$

The energy consumption for the example simple temperature sensor can now be calculated. The π-vector was found, and the values for the three elements of the π-vector are listed in (4-28). The reward for each transition is listed in the reward matrix, R, in (4-31). First, the q vector is found using (4-33) is shown in (4-40).

$$q = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix} = \begin{bmatrix} 3.17 \\ 12.20 \\ 7.90 \end{bmatrix} \tag{4-40}$$

With the q vector and the π-vector, the average energy consumption of the simple temperature sensor for one transition, g, is computed using (4-35).

$$g = 6.0207 \tag{4-41}$$

Hence, the simple temperature sensor consumes 6.0207 mW (milliwatts) of energy for each transition. Assuming each transition represents an average of 1 second, thus making, t_A, 1 second, the energy consumed in one hour is simply, g, multiplied by the number of transitions in one hour, in this case, 3600. The energy consumed in one hour is given by the following equation.

$$E_{\text{Total}-1\,\text{Hour}} = n \cdot g = 3600 * 6.0207 = 21674.52\ \text{mJ} \tag{4-42}$$

The energy consumed in one day is calculated by simply adjusting the quantity, n, in (4-35) to the number of transitions in one day, in this case 86400. Thus, the energy consumed in one day is given by the following equation.

$$E_{\text{Total}-1\,\text{Day}} = n \cdot g = 86400 * 6.0207 = 520188.48\ \text{mJ} \tag{4-43}$$

The energy consumption of the simple temperature sensor can be found for one-week, one-month, one-year, etc. simply by changing the parameter, n, in (4-35) to the appropriate value. The energies consumed for four additional periods are listed in Table 4.1.

Table 4.1: Value of n from (4-35) and the energy consumed by the simple temperature sensor for four different periods of operation.

Period	Value of n	Energy Consumed (mJ)
1-Week	604800	3641319.36 mJ
1-Month (30 Days)	2592000	15605654.40 mJ
1-Year (365.25 Days)	31557600	189998842.32 mJ
10-Years (3652.5 Days)	315576000	1899988423.20 mJ

By comparison, a 60 W light bulb consumes 5,184,000,000 mJ (milli-Joules) of energy in one day.

5.0 SINGLE ENTITY MARKOV PROCESSES

A single Markov process and a set of associated input parameters defining the functionality of an entity can be used to describe the energy consumption of that entity in isolation. These Markov processes are used as the basis for describing the energy consumption of the entire network. In these Markov processes the states correspond to the tasks or stages of tasks (for more complex tasks) that the entity performs during its operation with the transitions representing the actions required to complete those tasks. The Markov processes for each entity are general in the sense that they contain chains to describe the energy consumption of all possible tasks for an entity. The general model is customizable to describe a real-world entity by altering the probability matrix and altering the reward matrix through input parameters describing the functionality of the entity, and by structurally altering the Markov process.

5.1 NODE MARKOV PROCESS

The node accounts for the majority of the entities in most sensor or RFID networks. The node contains sensors and/or actuators along with communications devices and some processing capability. The generic node must perform five basic tasks within the sensor network. First, the node must take sensor readings to monitor some phenomenon. Second, the node may contain actuators, and must operate those actuators. Third, the node must process data and received messages. Fourth, the node must receive messages from the network. Finally, the node must transmit messages to the remainder of the network.

The node may contain only sensors, only actuators, or a mixture of sensors and actuators. Nodes may contain a single sensor or multiple sensors. The node must take and process sensor readings periodically. The frequency of these readings is an input parameter to the Markov

process for the node. A high sensor read frequency describes a node that must take and process sensor readings often, and low sensor read frequency describes a node that takes and processes frequency readings very infrequently. For example, a node designed to monitor for hazardous chemicals in a chemical plant may require a reading every minute to ensure safety while a node in an HVAC system may only need to read the temperature sensor every fifteen minutes. A state describing the energy consumption of the task to take a particular reading will be present in the Markov model. Tasks may require readings from multiple sensors and the designers specify these tasks.

Similarly, nodes containing actuators must periodically adjust the actuator output. These actuators provide input into another system, for example controlling a rotating gear. The type of system dynamics determine the actuator update frequency that the actuator(s) are responsible for controlling. Some systems require rapid input changes to maintain stability while others require infrequent input changes to maintain stability. The actuator update frequency is an input into the Markov process describing the node. There will be a state to describe the energy consumption of each actuator task performed by the node. As with the sensor tasks the number of actuator task states must be limited to prevent the explosion of the dimensions of the Markov process.

The node may receive a message during operation. This message could be a command or request from a type 1 or type 2 sink, or a data message from another node. The node must process all received messages. Some messages may contain errors, and not all errors can be corrected. The node discards those messages. However, some errors can be corrected with additional processing of the message.

The transmitter may allow for the adjustment of transmission power. This control allows the node to adjust the transmission power and by extension, the range of the transmitter to use minimize the amount of power to transmit the message from the source to the destination. The node can also use this feature to overcome interference, such as a thunderstorm, in the environment. This is represented in the general Markov process for the node by having multiple transmit states, each for a different level of transmit power. The number of different transmit levels must be kept relatively small to prevent the dimensionality of the Markov process from exploding. The number of different transmit and sleep states must be kept reasonably small to prevent the dimensionality of the Markov process from exploding. The Markov process for the node is shown in Figure 5.1.

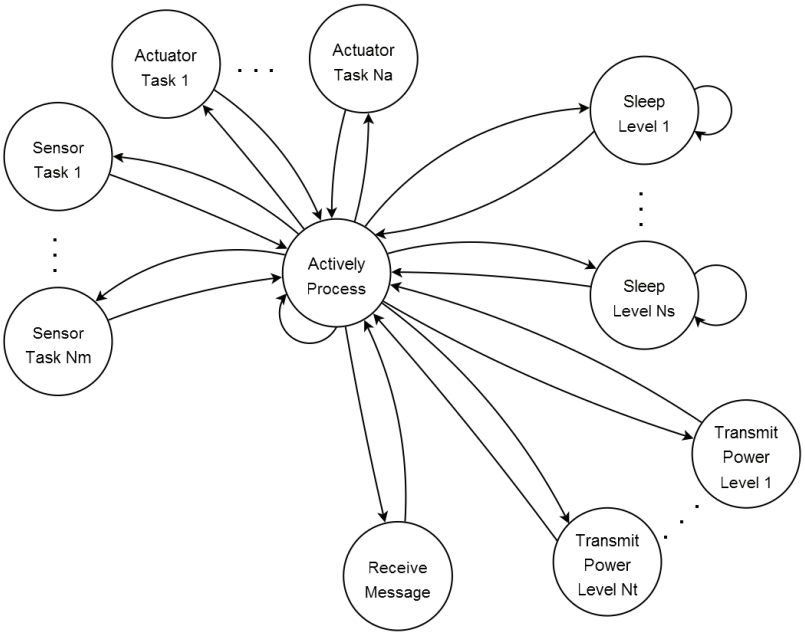

Figure 5.1: Generic Markov process for a node.

The transition probabilities in the generic process are simple to find. The probability of transitioning into one of the task states is simply the probability of that task occurring. This probability can be found using by dividing the number of times per day a particular task is executed by the number of transitions possible in a single day. For states with a loop-back transition, the same method can be applied to determine those transition probabilities.

5.2 TYPE 2 SINK MARKOV PROCESS

The type 2 sink provides a link between the outside world and the sensor network. The primary function of the type 2 sink is to listen for data messages sent by nodes within the sensor network and to transfer that information to the outside world. The type 2 sink also injects commands from the outside world into the network, however this event is rare as ideally once initialized the

sensor network will function without interaction from the outside world. Thus, the type 2 sink only transmits to the network when it receives a command from the outside world.

The type 2 sink is also equipped with a microprocessor allowing the sink to process the received data. This processing capability can be used to reduce the volume of messages sent to the outside world. Two examples of this are data fusion and data compression. Reducing the amount of data transmitted can significantly reduce the energy consumption of a sensor network, but only if the additional energy used processing the data is offset by a larger reduction in energy consumed by the transmitters.

The type 2 sink must communicate with entities inside the sensor network and outside the sensor network. In both cases, communication includes receiving and transmitting messages. Two sets of communication states will be used to account for these four tasks. The first will model the communication with the outside world and consists of one receive state and a number of transmit states. Each transmit state represents a transmission at a different power level. Communication inside the sensor network is similar, with a single receive state and a number of transmit states each representing a different transmitter power level. In both communication cases, the number of transmit states (transmitter power levels) must be kept reasonably small to prevent the dimensions of the Markov process from exploding.

The type 2 sink can process data, and a state is assigned to represent the task of actively processing data. The type 2 sink can also enter sleep mode where different components can be put into a low power state. There can be multiple levels of sleep states, and each level will be represented by a separate state. As with the two sets of transmit states the number of sleep states must be kept relatively small to keep the dimensionality of the Markov process within reasonable limits. The generic Markov process for a type 2 sink is shown in Figure 5.2.

The transition probabilities in the generic process are simple to find. The probability of transitioning into one of the task states is simply the probability of that task occurring. This probability can be found using by dividing the number of times per day a particular task is executed by the number of transitions possible in a single day. For states with a loop-back transition, the same method can be applied to determine those transition probabilities.

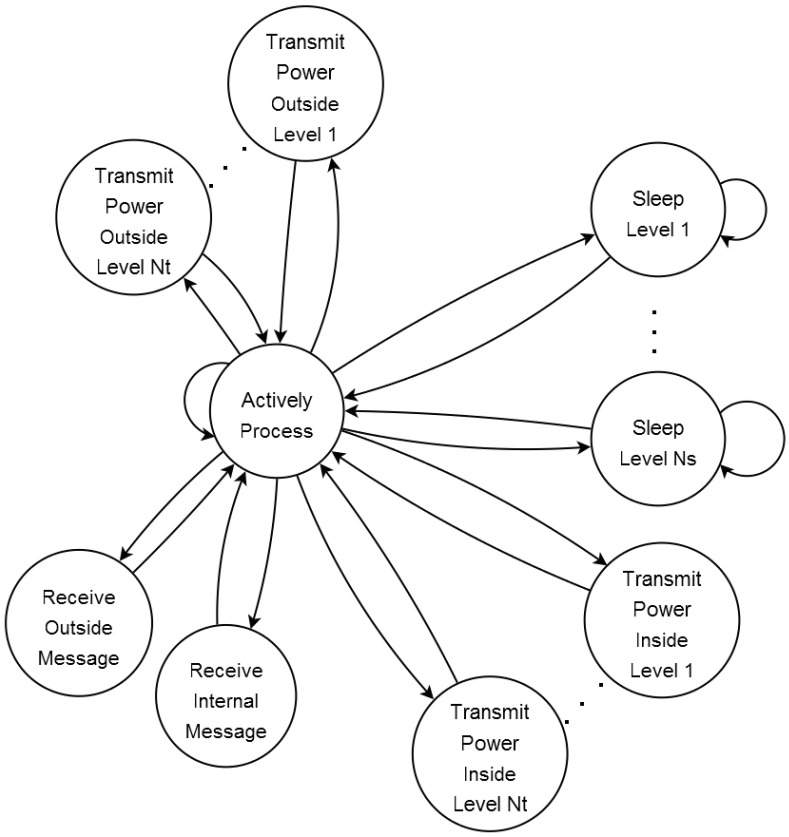

Figure 5.2: Generic Markov process for a Type 2 Sink.

5.3 TYPE 1 SINK MARKOV PROCESS

The type 1 sink can function as either a type 2 sink or as a node, and can change rolls during operation. Therefore, the type 1 sink must be able to perform all the tasks of a node and of a type 2 sink. The Markov process for the type 1 sink is obtained by combining the Markov processes for a node and for a type 2 sink. In this case the number of sensors, actuators, and transmitter power levels must be reasonably limited in order to prevent an explosion in the dimensionality of the Markov process. The generic Markov process for the type 1 sink is shown in Figure 5.3.

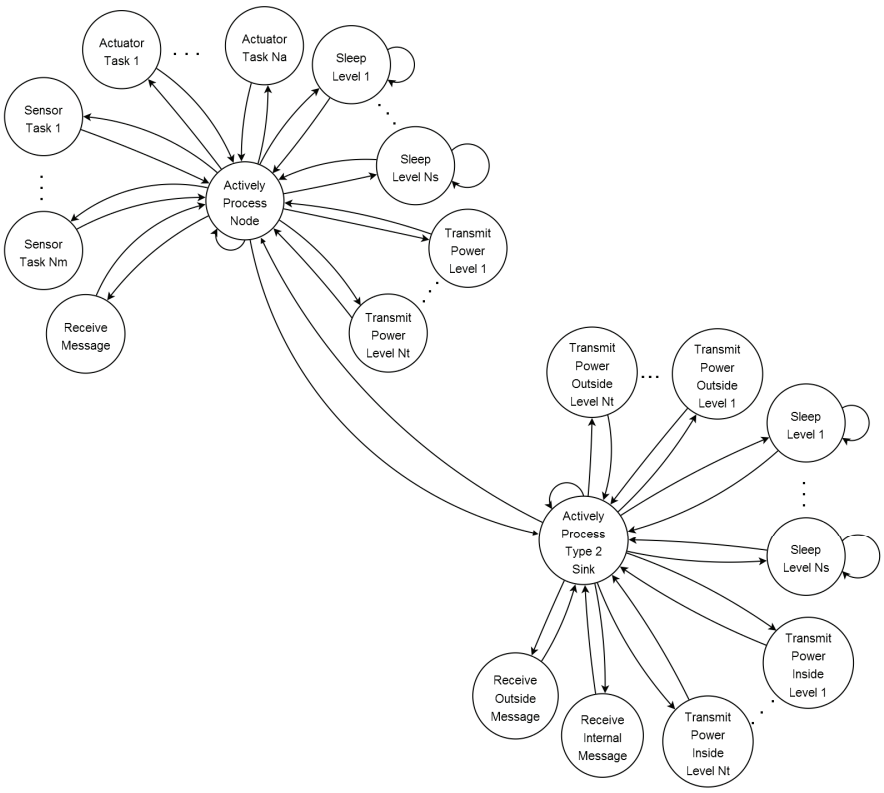

Figure 5.3: General Markov process for a Type 1 Sink.

The transition probabilities in the generic process are simple to find. The probability of transitioning into one of the task states is simply the probability of that task occurring. This probability can be found using by dividing the number of times per day a particular task is executed by the number of transitions possible in a single day. For states with a loop-back transition, the same method can be applied to determine those transition probabilities.

5.4 MARKOV PROCESSES FOR TOPOLOGICAL ENTITIES

The entities that comprise a sensor or RFID network group themselves into the topological groups described in the previous section. The topological groups enable the designer to view the sensor or RFID network at a higher, more abstracted level. Topological groups provide a means

to decompose the entire sensor or RFID network into a set of sensor or RFID network topologies. The number of sensor or RFID network topologies after the sensor or RFID network is decomposed is less than the number of base entities in the original sensor or RFID network. By investigating each of the topological entities designers are able to gain insight into the behavior and performance of the sensor or RFID network on a whole providing for quicker, more efficient simulation of the sensor or RFID network to determine energy consumption.

A Markov process similar in structure to those developed to describe the individual entities describes the topological entity. The Markov process describing a topological entity must accurately model the actions of all entities in the topology and keep the dimensionality of the Markov process to a minimum. Keeping the dimensionality at a minimum reduces the amount of computations, hence time, to evaluate the Markov process to determine energy consumption.

The Markov processes for the topological entities will focus on the tasks that each topology must perform. Sensor or RFID networks must perform a few common tasks to function and some specialized tasks based on the requirements of the individual sensor or RFID network. However, within a single or RFID sensor network, all required tasks can be determined from the requirements and specifications of that particular network. Focusing on the performance of these tasks enables the entities to be quickly and easily combined into topological entities and for the creation of the Markov process for the topological entities.

5.4.1 Tasks of a Sensor Network

Sensor and RFID networks must perform a number of different tasks. The Markov model for the sensor or RFID network must represent each of these tasks. Focusing on the tasks a sensor or RFID network topology performs limits the dimensionality of the Markov process describing that topology. Limiting the dimensionality of the Markov process is critical to ensuring that the Markov process can be evaluated within a reasonable time. The individual topology groups that combine to make up the sensor or RFID network must perform either all or a sub-set of the tasks. The Markov model for each topology group must represent those tasks that must be performed by that topology group.

The following six tasks form a basis of tasks that a topology may perform. These basic operations form a basis for the more complex tasks that a sensor or RFID network may perform. The most basic task of a sensor network is to collect data from the sensor nodes. Collecting data involves taking sensor readings and communicating those readings to the appropriate sink(s). Processing of the sensor reading by the sensor nodes or the sinks is the second task required by the sensor network. For example, the processing could be focused on fusing the data, or normalizing sensor readings to a known standard. The third task of a sensor or RFID network is for the sinks in the sensor or RFID network to communicate the collected data to the outside world. Fourth, sinks in the sensor or RFID network must receive commands from the outside world. Fifth, entities in the sensor or RFID network must process commands from the outside world or other entities. Sixth, the entities in the sensor or RFID network must distribute commands they receive. The six basic tasks that form a basis for tasks within a sensor network are shown in Figure 5.4. More complex tasks can be decomposed into a sequence of the six basis tasks, or as a customized extension of one of the six basis tasks.

The above six basis tasks provide the foundation for developing the Markov process structure describing each topological entity. The structure of the Markov process describing each topology type will be the same, but the reward and probability matrices will be customized for each topology type. A general Markov process for a topological entity performing each of the six basis tasks is shown in Figure 5.5.

The transition probabilities in the generic process are simple to find. The probability of transitioning into one of the task states is simply the probability of that task occurring. This probability can be found by dividing the number of times per day a particular task is executed by the number of transitions possible in a single day. For states with a loop-back transition, the same method can be applied to determine those transition probabilities.

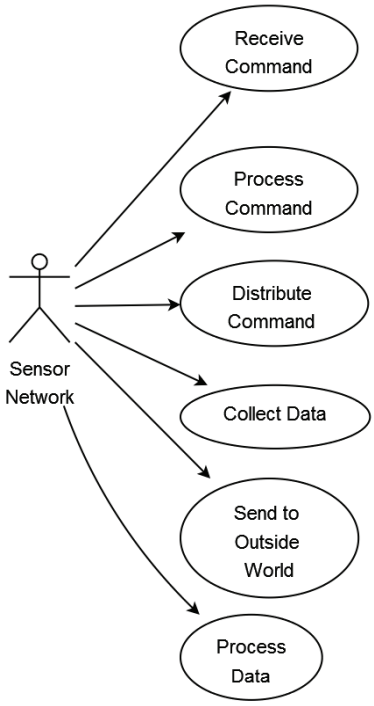

Figure 5.4: Six basis tasks forming a basis for all tasks of a sensor network.

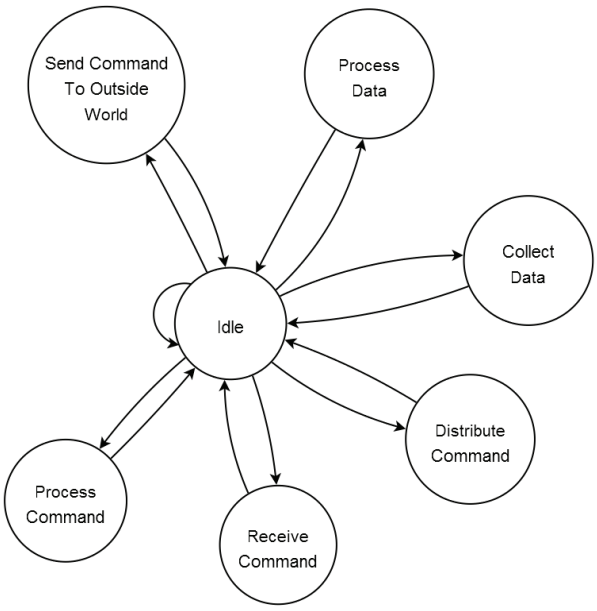

Figure 5.5: General Markov process for a topological entity.

5.5 MOBILITY

A sensor network containing entities capable of moving around may be able to reposition those assets to improve the quality of the network. For example, some propose deploying mobile energy harvesting entities as part of a sensor network capable of recharging batteries and replacing the batteries of dead entities automatically [21]. Mobile entities can be used to repair or add new communication links in networks or to arrange the sensors in a better topology for monitoring [43].

Mobility offers many useful features in sensor networks and mobility is achieved in sensor networks by providing a mobile version of a static entity [21, 43, 44]. An entity called a mobile entity will facilitate the addition of mobility to a static entity (node, type 1 sink, or type 2 sink).

Sensor networks have also been envisioned containing entities to transport stationary entities from one location to another [45]. The Networked Infomechanical System (NIMS) provides entities that can pick up, transport, and drop off stationary entities [45]. An entity mover entity will provide this capability to the networks investigated in this work. The entity mover will have the capabilities to pick up an entity, transport that entity to another location, and finally drop off the entity at the new location. With the mobile entity and the entity mover, mobility within sensor networks is covered.

This work focuses on wireless networks in which the entities are stationary, but the method developed can be employed in wireless networks containing mobile entities. As an entity moves the topology of the network changes reflecting that movement. Each network is made up of a collection of topologies that cooperate to perform a set of tasks. With mobility the makeup of the topology changes (nodes or sinks are moved from one topology to another topology). These changes in topologies can be determined or estimated beforehand and the network can then be analyzed using each set of topologies. Ideally, only those topologies that are effected by mobility must be reevaluated. Hence, mobility can be modeled by using a set of topologies that reflect the movement of entities within the network. Random movement can be modeled using random groupings of topologies to simulate the movement.

6.0 BASIC COMMUNICATION GRAPH

Interaction among entities in a sensor or RFID network is limited to communication. There are two types of communication that take place - useful and worthless. Useful communication is defined as an entity receiving a message (error free or correct) that is addressed to it. Worthless communication is defined as an entity receiving a message that is not addressed to it or a message that contains an error. Regardless of whether the communication is useful or worthless, the presence of a message in the channel will corrupt any message that is simultaneously in the channel and/or prevent an entity wishing to transmit a message by occupying the channel. Thus, no distinction need be made between useful and worthless communication, only the fact that a message that is present is required.

Graphs provide an excellent representation for many problems. Some areas where graphs are extremely useful are in cell-library binding, and network connectivity and routing [46-48]. Sensor networks can be represented using a graph. In this research, sensor and RFID networks will be represented by a basic communication graph with the vertices representing entities and edges representing communication links. The edges are each assigned a weight corresponding to the distance (or power) required for one of the two vertices connected by the edge to communicate with the other. Such a graph will be referred to as a basic communication graph.

The probability that a message is present will be determined using a graph describing communication called a basic communication graph in this research. Each entity making up the larger topology is represented by a vertex in the graph. Edges in the graph represent two vertices (entities) that are within direct communication range (point to point) of each other. A graph example of a partial mesh network containing 9 nodes where each node is within direct communication range of its neighbors is shown Figure 6.1. The nodes in Figure 6.1 can be replaced with sinks or topological entities without any modification to the concepts of the basic communication graph.

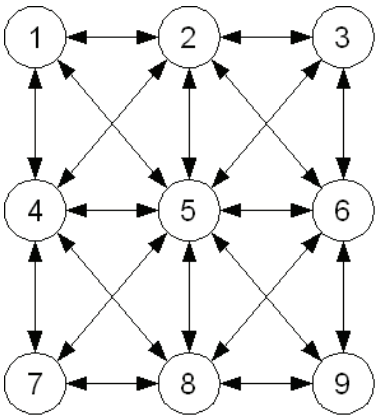

Figure 6.1: Example of a basic communication graph of a mesh network.

Each of the corner nodes (nodes 1, 3, 7, 9) is within range of only 3 other nodes, while the center node (node 5) is within range of 8 other nodes. Clearly, node 5 (center node) will have more difficulty in obtaining a clear channel to communicate, because any of the other 8 nodes transmitting will prevent node 5 from accessing the channel (medium). As a result, the probability that node 5 must wait for the communication channel to be free is higher than that for nodes 1, 3, 7, and 9 because node 5 can communicate with more nodes.

Similarly, the probability that node 5 receives an uncorrupted message is lower than for the corner nodes since any transmission from one of the other 7 nodes can corrupt the message being received at node 5 (assuming an Aloha type protocol). For example, if node 1 and node 9 transmit a message to node 5 at the same time both messages will be corrupted when received at node 5. The number of edges ending at each vertex, or the degree of each vertex, will be used to derive these probabilities. The probability of interference, F, at a particular entity, v, is found using the following equation.

$$F = \frac{1}{\deg(v)} \tag{6-1}$$

This procedure is applicable to cases where vertices represent topologies as well. The degrees will be known for those networks that have a defined topology. With the degree, it is possible to obtain the probabilities mentioned above for each entity (vertex) in the graph.

7.0 SENSOR AND RFID NETWORK TOPOLOGY IDENTIFICATION

Simulations of large sensor and RFID networks take a long time to run - often many hours or days. Configuring the simulation environment for a large sensor or RFID network is not a trivial task. The time required to setup and run one simulation typically prohibits a thorough exploration of design alternatives during the development of a sensor network. Limited exploration of design alternatives leads to sub-optimal sensor and RFID networks possibly not meeting the requirements. The large number of entities that must be configured and simulated causes a bottleneck for simulation. Reducing the number of entities that must be simulated speeds up both the configuration and execution time of the simulation allowing exploration of more design alternatives. However, reducing the number of simulated entities may reduce the accuracy of the simulation.

A network topology is an association of individual entities in a network. Some examples of network topologies are crossbar, tree, ring, and star. Topologies are normally defined based on communication links between entities. In terms of a computer network, such as Ethernet or token-ring, the topology defines how one computer can communicate with another computer as opposed to physical topology. Topologies are more important in sensor and RFID networks as the entities use topological information to communicate but also to work together to perform the required functions.

All networks, including sensor and RFID networks, can be defined by a topology. Ad-hoc sensor networks use an algorithm or a set of parameters to setup the network. A topology can always be overlaid to such an ad-hoc network. The ad-hoc network could also be programmed with the desired topology and the entities in the network will then organize themselves into such a topology. Topologies provide a means to describe the operation of a group of entities without having to simulate the operations of each individual entity contained in the topology.

At the macro-level, a sensor or RFID network can be described by a topology and a set of tasks that must be performed. The macro-level topology of a large sensor or RFID network is composed of a number of entities. Each entity represents a separate portion of the larger sensor network and has a topology. This decomposition is repeated until the final set of topologies consists only of base level entities. The decomposition of sensor and RFID networks using topologies results in a set of topologies that can be expressed using a tree structure. This tree structure will be referred to as a partition tree. A parent node in the partition tree structure represents a particular topology containing as members all child nodes of that parent node. Partitioning of the topologies continues until reaching the leaf nodes, which represent single base entities. The root of the partition tree represents the macro-level topology description of the sensor or RFID network, and the lowest level (the leaves) of the tree represents the micro-level description of the sensor or RFID network. The levels between the root and leaves represent different levels of abstraction. Topologies provide the linkage between two layers of abstraction.

The top layer contains the basic topology of the entire network. Each lower layer shows the topology of the sensor or RFID network in greater detail. At the lowest level, the topology of the sensor or RFID network with each entity in the sensor or RFID network is shown. Information obtained from an analysis performed at one level of the tree can be propagated one level immediately above or below the current level, allowing refinement to those levels.

The partition tree of a sensor or RFID network can be constructed in two ways; from the bottom up, or from the top down. The bottom up construction begins with the base level entities and groups them into small topologies. This grouping continues until the entire network is grouped together into a single topology. This method is useful when a topology must be applied to a group of base level entities that have already been deployed. The top-down method starts with the sensor or RFID network assigning a macro-level topology. The macro-level topology, is constructed using topological entities. Each topological entity is then described by a topology and the process continues until all topologies have been broken down into base level entities. The top-down method is useful when designing a sensor or RFID network that has not yet been deployed. Additionally, if the phenomena of interest are identified beforehand the top-down method provides an efficient way to design the sensor network around monitoring those phenomena of interest.

The entities contained in each topology at each level cooperate to perform a set of tasks. A general Markov process that contains chains for all tasks required in a particular sensor or RFID network topology will be used to describe the energy consumption of each topology. The general Markov process is customized to a specific behavior and task by adjusting the probability and reward matrices, or the structure of the Markov process. Focusing on tasks performed by the topology prevents the Markov process describing that task from growing to an unmanageable dimension.

Thus, the sensor or RFID network can be divided into smaller components and a Markov process with a reasonable degree describing each component. Therefore, the degree of any one Markov process is not too large to prevent performing the matrix operations or solving the set of linear equations.

7.1 BOTTOM-UP CONSTRUCTION

Constructing the partition tree using the bottom-up method continually groups entities together forming larger entities until a single entity contains the entire network. This method is best used when the sensor network has already been deployed and a topology must be fit to the network.

An example sensor network containing 25 base level entities is shown in Figure 7.1. The arrows connecting the entities denote a communication link between entities. As can be seen from Figure 7.1, there are four different spatial groupings of base level entities, top, middle right, bottom right, and bottom left.

These four groups of entities can easily communicate with each other due to the short distances between them. The communication links between entities are shown in Figure 7.2.

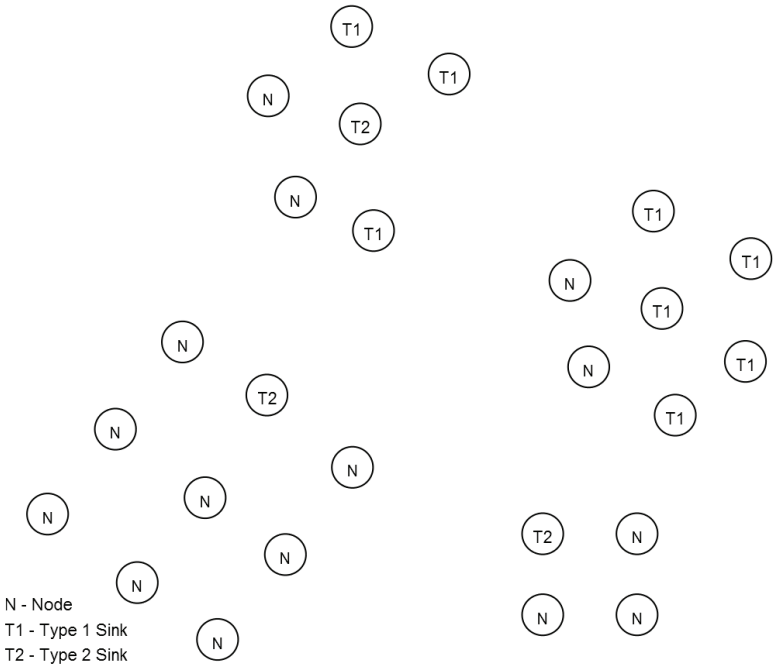

Figure 7.1: Example sensor network consisting of twenty-five entities.

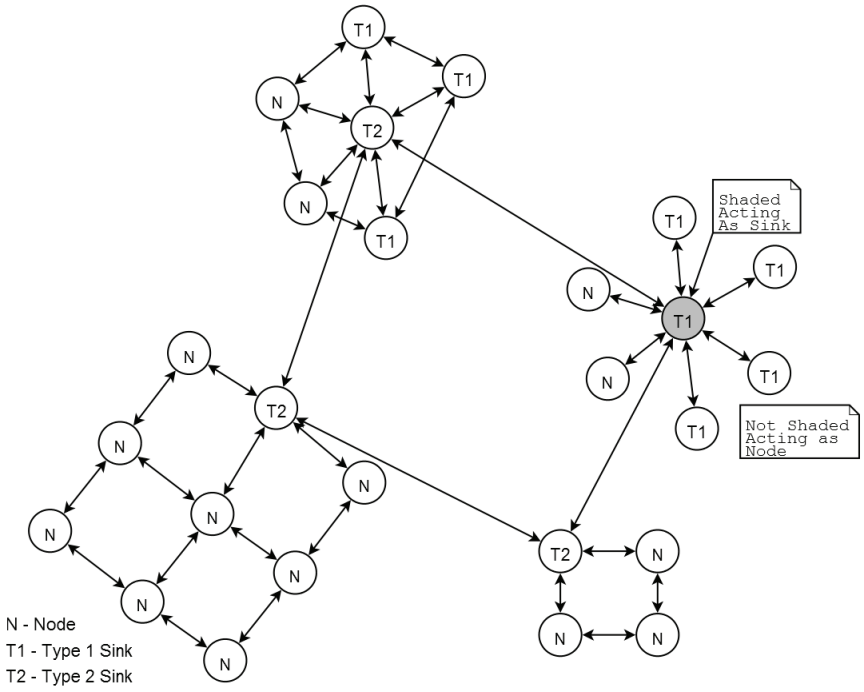

Figure 7.2: Example sensor network at entity level.

These groupings can be formed together into four separate topologies, two mesh topologies (lower right and lower left groups), a cluster topology (top group), and a star topology (middle right group). Thus, this sensor network can, at the highest level, be viewed as a mesh network containing four topologies and is shown in Figure 7.3. Following this same procedure larger sensor networks can be reduced to a simpler representation.

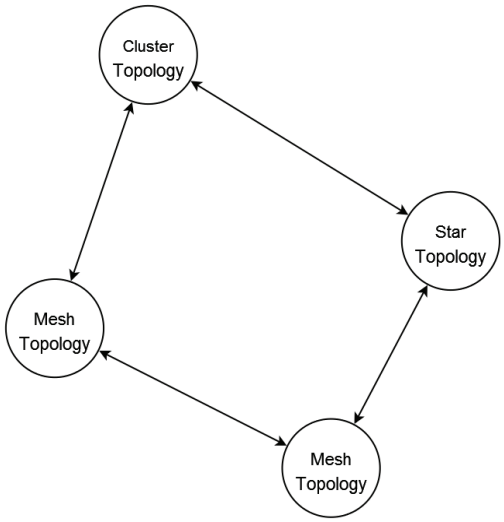

Figure 7.3: Top layer topology of the example sensor network shown in Figure 7.2.

The example sensor network can now be partitioned into a partition tree. The partition tree for this example sensor network will be constructed from using the bottom-up construction method. First, individual entities will be grouped to form small topologies. Next the small topologies will be combined with other topologies to form large topologies. The large topologies will be combined to form larger topologies. This process will continue until the sensor network can be represented by a single topology.

A partition tree will be created using the example sensor network previously described. Each node in the example sensor network is assigned a unique ID. This is purely for clarity of the example, as the ID will be used to show where each entity and topology group fit within the partition tree. The example sensor network with unique IDs is shown in Figure 7.4.

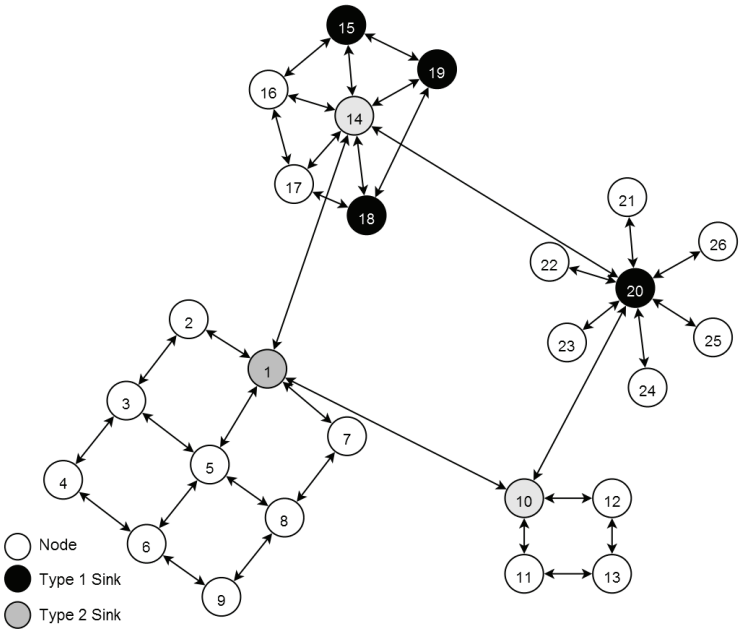

Figure 7.4: Example sensor network with each entity assigned an ID. The example tree structure will be constructed using this network.

The first step is to identify topology groups within the sensor network structure. Starting at entity 1 it is clear that entities 1 through 9 form a mesh topology, with entity 1 providing the link to the rest of the network. Therefore, entities 1 through 9 will be grouped into a mesh topology. Entities 10 through 13 form a second mesh (or ring) topology and the entity 10 links that mesh topology to the remainder of the network. Therefore, entities 10 through 13 will be grouped into a second mesh topology. Entities 14 through 19 form a cluster topology with entity 14 providing links to the rest of the network. Therefore, entities 14 through 19 will be grouped into a cluster topology. Entities 20 through 26 form a star network with entity 20 providing links to the remainder of the network. Therefore, entities 20 through 26 will be grouped together in a star topology.

The second step is to reduce the complexity of the sensor network by replacing each group of entities included in a particular topology with a single entity representing that topology. The example sensor network in Figure 7.4 is reduced by replacing the groups of entities with single topology entities as shown in Figure 7.5. The four entities contained in Figure 7.5

represent a collection of base level entities and reduce the number of entities that require analysis from twenty-five to four.

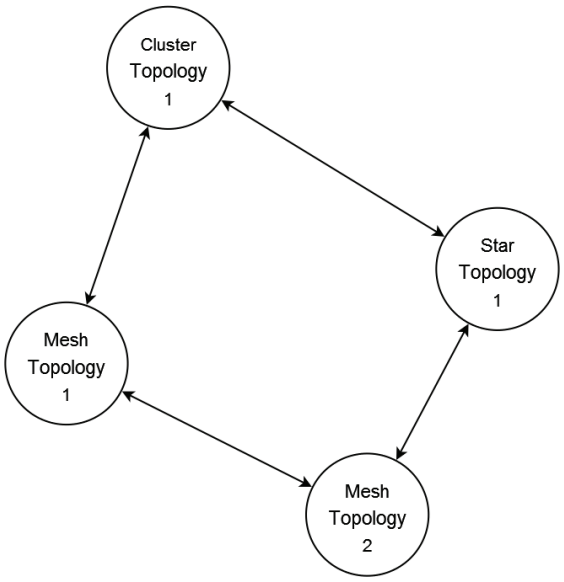

Figure 7.5: The reduced example sensor network with topology entities replacing groups of individual entities shown in Figure 7.4.

At this point, the lowest two layers of the partition tree have already been obtained. The lowest layer contains only leaves, while the higher levels may contain topologies or leaves. The leaves represent individual entities. The first two layers of the partition tree are shown in Figure 7.6.

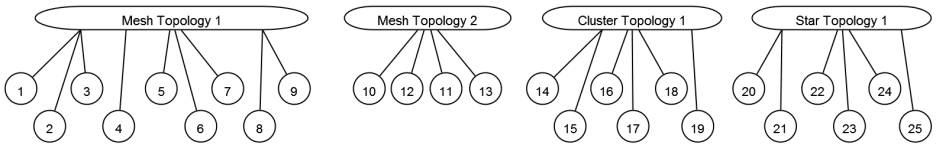

Figure 7.6: Lowest two layers of the partition tree for the example sensor network.

The second step is to combine the topological entities into a larger topology. The reduced sensor network shown in Figure 7.5 resembles a mesh network. The four topological entities can be grouped together into a single mesh network. This mesh network topology now represents the entire sensor network as a single entity indicating that the bottom up method has finished. The final partition tree of the example sensor network in Figure 7.1 is shown in Figure 7.7.

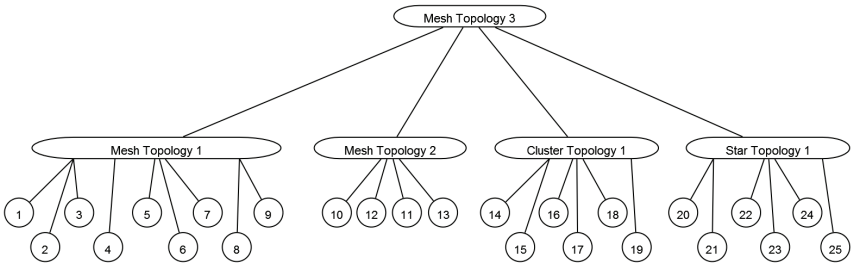

Figure 7.7: Final partition tree constructed using the bottom-up method of the example sensor network shown in Figure 7.4.

The above example illustrates how a topology can be enforced on a sensor network. In the case of sensor networks whose entities are deployed at random locations, a topology can still be assigned a priori as the entities in the sensor network can automatically organize themselves to form the required topology.

Combining individual nodes together into topological groups allows the reduction of the number of Markov processes that must be evaluated to obtain the energy consumption of a sensor network. However, care must be taken to ensure that the Markov process describing each topology does not have dimensions so large that the evaluation of the Markov process requires too much computing time and resources. If too much computing time and resources are required, at some point the benefit obtained from grouping entities into topologies will be lost as direct evaluation of the Markov process describing each entity would require no more computing time and resources. The Markov processes describing each topology will focus on the tasks the

topology must perform. Each task can be ideally represented by a single state, thus keeping the dimension of the Markov process describing a topology within reason.

The basic communication graph is useful in identifying topologies as needed for the bottom-up method. The first step in the bottom-up method is to identify the initial topologies from the large sensor network. Two methods can be used to construct the topologies in the bottom-up method. The first involves using procedures used for cell-library binding in digital CAD packages. The second method creates a set of graphs based on the basic communication graph to identify topologies within the sensor network.

7.1.1 Cell-Library Binding Method

A number of methods exist to map a digital circuit to cells in a particular technology library [46]. The cell library binding process takes such a graph and covers it with logic cells contained in the library [46]. Cell library binding algorithms generally fall into one of two categories, structural matching and Boolean matching [46]. In structural matching the cell library binding process, the circuit design is provided as a graph with vertices representing a logical operation (i.e. NAND, NOR) with edges indicating the flow of data or signals through the system [46]. Structural matching searches for matches in the circuit graph to graphs of the individual cells in the cell library [46]. In Boolean matching, the circuit and library cells are described by Boolean functions, called patterns, and the circuit is searched for patterns matching those of the library cells [46].

With respect to this research, the structural matching algorithms are applicable because sensor and RFID networks can be represented using the basic communication graph. The basic communication graph of a sensor or RFID network is very similar to the circuit graph that are inputs into cell library binding tools. Structural matching searches a pattern match in circuit graph matching a pattern in the cell library [46]. This requires that both the circuit graph and cell component graphs be translated into comparable structures [46]. Structural matching can be transformed into a graph isomorphism problem, which is NP-complete, but the tree covering or matching problem can be solved in linear time [46]. Cell library binding solutions make use of the linear time algorithms for tree covering. Therefore, if the sensor network graph can be

converted into a tree structure the same linear time cell library binding methods used for logic synthesis can be used to identify the topologies in the sensor network.

Trees are a special form of graph containing no cycles but the majority of sensor and RFID networks contain cycles [49, 50]. Constructing a new basic communication graph for a sensor or RFID network without cycles allows the use of existing solutions from cell library binding to be used to identify the topologies contained in the sensor network. Sensor and RFID networks connect to the outside world through type 1 and type 2 sinks. Type 1 and type 2 sinks are roots of trees within the sensor network. Using the type 1 and type 2 sinks as a starting point, a spanning tree or minimum spanning tree (MST) can be constructed connecting all entities in the sensor network to at least one sink entity. A spanning tree is a tree containing all the vertices in a given graph. A minimum spanning tree is a tree containing all vertices in a given graph where the sum of the edge weights is the minimum. The graph over which the spanning tree or minimum spanning tree is constructed can be a sub-graph of a larger graph. Using the minimum spanning tree focuses attention on the shortest possible links helping to group entities spatially. The spanning tree requires less computational power to generate, but does not automatically group entities together spatially. The spatial grouping of entities is related to the accuracy of the behavior models of the topological entities because entities that are close together tend to work together because the relative closeness of the entities allows them communicate using a lower transmission power level than entities that are far away.

Two greedy algorithms commonly used to find a minimum spanning tree of a graph are Prim's algorithm and Kruskal's algorithm [49, 51]. Kruskal's algorithm starts by creating a tree containing a only single vertex. Hence, for a graph G containing V vertices and E edges, Kruskal's algorithm begins by constructing V trees containing one vertex each. The edges E are then sorted into a list in increasing order of weight. The first edge in the list (edge with minimum weight) is removed from the list and added to the minimum spanning tree if and only if the edge connects two unconnected trees. If the edge connects two already connected trees, it would create a cycle violating the definition of a tree. The process above is repeated until the minimum spanning tree is formed. The run-time of Kruskal's algorithm is O(Elog2(V)), where E denotes the number of edges in the graph, and V denotes the number of vertices in the graph [49].

Prim's algorithm starts at a given vertex and grows the minimum spanning tree by adding edges to the current minimum spanning tree. During each step, the edge with minimum weight connecting one of the vertices currently in the minimum spanning tree to a vertex not in the minimum spanning tree is added to the minimum spanning tree. The run-time of Prim's algorithm O(Vlog2(V) + Elog2(V)) = O(Elog2(V)) because E > V is true in the most sensor networks [49]. The run-time of Prim's algorithm can be improved if a Fibonacci heap is used to identify the minimum weight edge that adds another vertex to the minimum spanning tree [49]. A Fibonacci heap is a special kind of heap data structure that allows for updating the key, or data, for a node within the Fibonacci heap in O(1) or constant time and allows for the node with the minimum key to be extracted in O(log2(V)) [49]. Thus, Prim's algorithm implemented to use a Fibonacci heap has a run-time of O(E + Vlog2(V)) which is better if V << E [49]. Prim's algorithm implemented with a Fibonacci heap is most likely better for finding the minimum spanning tree of the basic communication graph of a sensor or RFID network because the number of edges is usually much greater than the number of vertices (entities).

The minimum spanning tree of the sensor network may be a single tree or a set of minimum spanning trees. The latter case is encountered when portions of the sensor or RFID network are disconnected from each other. Each disconnected portion of the sensor or RFID network is connected to the outside world through one or more sinks. If a disconnected portion has no sink, it is isolated from the outside world and can not function as part of the sensor network and will not have a MST (minimum spanning tree) because all MSTs must be rooted at a type 1 or a type 2 sink. A simple example of a sensor network with three disconnected portions is shown in Figure 7.8.

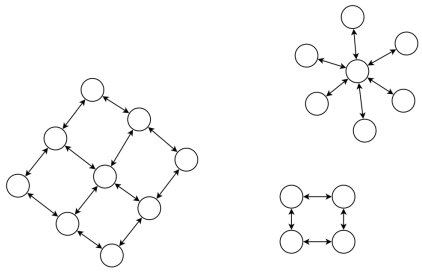

Figure 7.8: Sensor network containing three disconnected portions.

There will be at least one minimum spanning tree for each disconnected portion. Hence, the maximum number of spanning trees is equal to the total number of type 1 and type 2 sinks in the sensor network, and the minimum number of spanning trees is equal to the number of disconnected portions of the sensor network.

In some sensor and RFID networks, a non-sink entity may not have path to a sink entity. Such an entity is referred to as an isolated entity. In this case that entity cannot be covered by any spanning tree rooted at a sink entity. Therefore, in this case a spanning tree must be computed for every sink in the network before realizing that no path exists from a sink to the isolated entity. This results in the worst possible runtime for the spanning tree algorithm. The fact that an isolated entity is not included in any spanning tree is acceptable because there is no condition under which isolated entities could participate in the sensor or RFID network.

Using one of the methods above, a set of minimum spanning trees of the network can be found, and then cell library binding methods can then be used on each tree to identify the topology.

7.1.2 Distance Graph Method

Sensor and RFID network topologies normally consist of a set of entities within a relatively close distance to each other. Recall that the power required to communicate over a given distance is proportional to the distance because the received power at an antenna in free space is determined by the following equation.

$$P_r = \frac{P_t \lambda^2}{(4\pi)^2 d^{\alpha}} \tag{7-1}$$

Where λ is the wavelength of the carrier frequency, d is the distance between the transmitter and the receiver, α is the power that the distance, d, is raised to, P_t is the transmit power, and P_r is the received power. The free space model uses 2 for α, while in a real world environment α will be larger due to interference. Thus, the received power decreases quickly as distance increases.

Hence, by grouping entities close together into topologies, the energy consumed by intra-topology communication is kept small. In addition, the duties of the sensor and RFID network over a small area are likely to be similar. Hence, the Markov process describing the topology more accurately describes the functions of the individual entities contained in the topology.

The basic communication graph, G, can be used to help identify topologies contained in the sensor or RFID network. The edges of the basic communication graph have weights assigned to them that correspond to the distance between the two vertices at the two ends of the edge. The identification of topologies will be demonstrated to illustrate the construction of the partition tree from the bottom-up. The basic communication graph of the example sensor network with edge weights denoting the distance between entities in meters is shown in Figure 7.9.

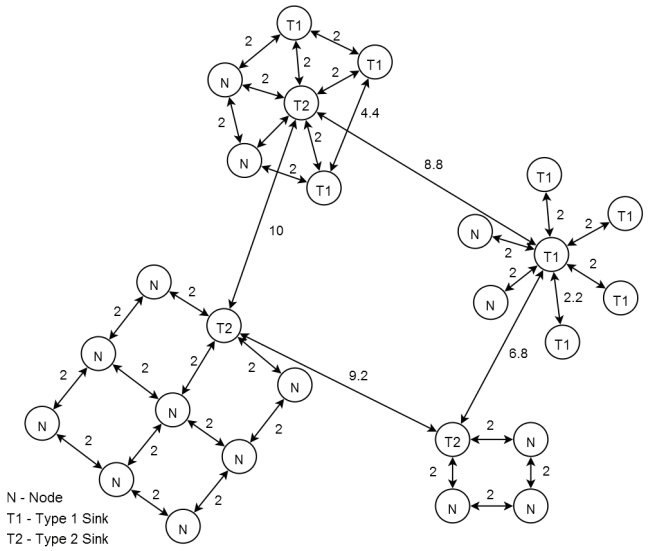

Figure 7.9: Basic communication graph, G, of the example sensor network. The numbers next to each edge are the edge weight representing the distance between the two end points in meters.

An arbitrary measure of distance, the distance unit, can be defined over the sensor network. One distance unit is defined as the minimum value of all edge weights, or the distance between the two closest base entities. The minimum edge weight in Figure 7.9 is 2, making 1

distance unit equal to 2 meters. Normalizing the edge weights simplifies the analysis. With the definition of a single distance unit, a new graph, the distance unit graph, G_D, can be created that is identical in structure to G with the exception that the edge weights are now given in terms of distance units, rather than pure distance (meters). The distance unit graph, G_D, created by dividing all edge weights in Figure 7.9 by 2 converting meters into distance units. The distances do not need to be normalized but normalizing the distances makes the analysis easier. The resulting distance unit graph G_D, is shown in Figure 7.10.

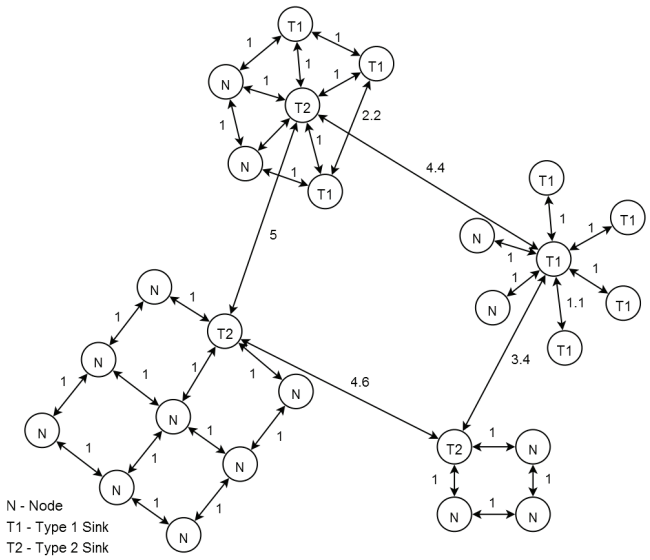

Figure 7.10: The distance unit graph, G_D, created from the graph G in Figure 7.9.

A set of graphs can be created from G_D with restrictions based on the maximum edge weight (maximum distance between two entities). Each graph, G_{Di} contains all vertices in G_D but only those edges with weight less than D_i, where D_i is in units of distance units. The set of graphs G_{Di} relate spatial information about the communication structure of the sensor or RFID network. Each graph in G_{Di} represents the sensor or RFID network with communication links of no greater length than D_i. Thus, the maximum weight of any edge in G_{Di} less than or equal to the number of edges in G_{Di+1}. Similarly, the number of edges in G_{Di} is always less than or equal to the number of edges in G_{Di+1}. Fewer computations are required to search G_{Di} to identify

topologies than to search G_{Di+1} making it desirable to try to cover the sensor or RFID network using the lowest possible distance graph. Defining G_{Di} as being the distance graph containing all edges of G_D with edge weight less than or equal to i, where i = 1, 2, 3, … The graphs G_{D1} through G_{D3} are shown in Figure 7.11 through Figure 7.13.

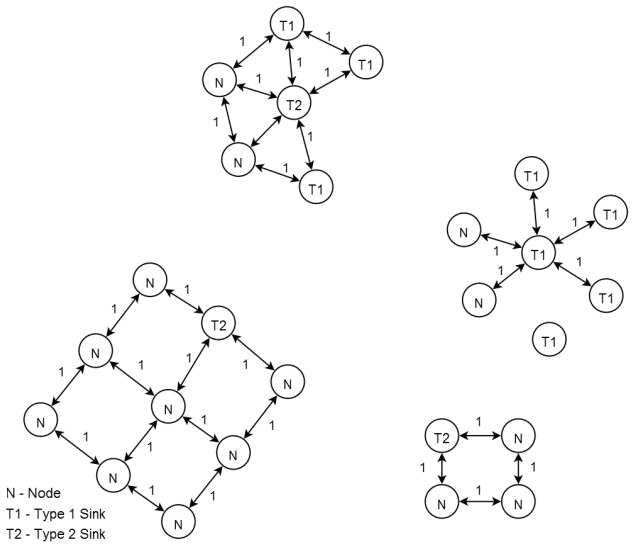

Figure 7.11: Distance graph G_{D1} showing only communication links of 1 distance unit or less.

Starting with G_{D1}, shown in Figure 7.11, it is easy to match the mesh topology pattern to the mesh topology consisting of four entities in the lower right corner. The larger mesh topology in the lower left corner requires matching of a larger mesh topology pattern. The star topology in the upper right corner can be matched to a star topology pattern containing a central entity connected to five other entities. The sixth entity, the unconnected type 1 sink, is not covered by the star topology or any other topology. The cluster topology in the upper left corner is incomplete as two nodes are not directly connected together. A star topology could be used to cover this topology if the links connecting the nodes are ignored. In this example it is assumed that the topology pattern must match exactly without ignoring any edges, however this is not necessary. Allowing edges to be ignored when identifying topologies may result in the topological covering being identified more quickly, but at the cost of accuracy. Hence, the graph

G_{D1} cannot be covered by base topologies, and the next distance graph in the set, G_{D2}, shown in Figure 7.12 must be employed.

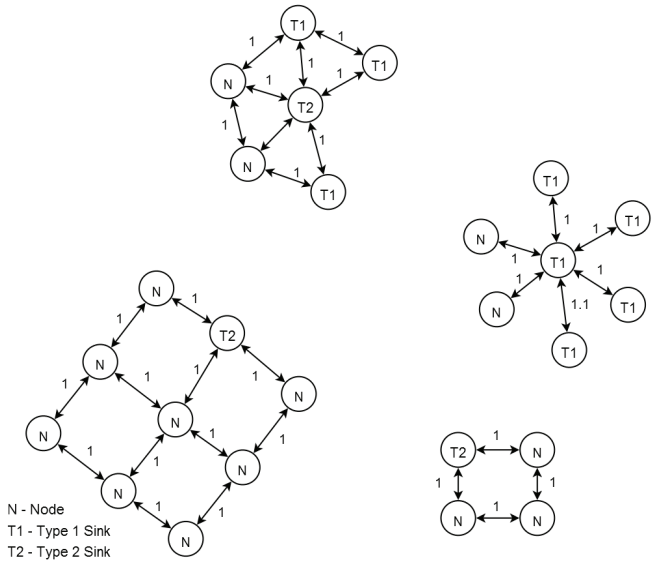

Figure 7.12: Distance graph G_{D2} showing only communication links of 2 distance units or less.

The final communication link of the star topology has been added in G_{D2} and thus the star topology in the upper right corner can be identified. If edges can be ignored then the cluster topology in the upper left corner could be covered with a star topology and all entities would be covered by base topologies indicating the procedure is complete. However, because in this example edges are not ignored, the cluster topology in the upper left corner is still unidentified so G_{D3}, shown in Figure 7.13 must be employed. Ignoring edges may decrease the accuracy of the assigned topology.

The final edge, linking the two nodes in the cluster topology in the upper left corner is included in G_{D3}. Hence, the cluster topology in the upper left can now be identified and all entities are covered by base topologies so we are done. The entities in G_{D3} can be replaced with the topological entities that were identified to cover them to create a new basic communication graph, G'. This new graph G' represents the sensor network at the first level of topological

abstraction and is shown in Figure 7.14. The edge weights in the basic communication graph G' in Figure 7.14 represent distance between topological entities in meters.

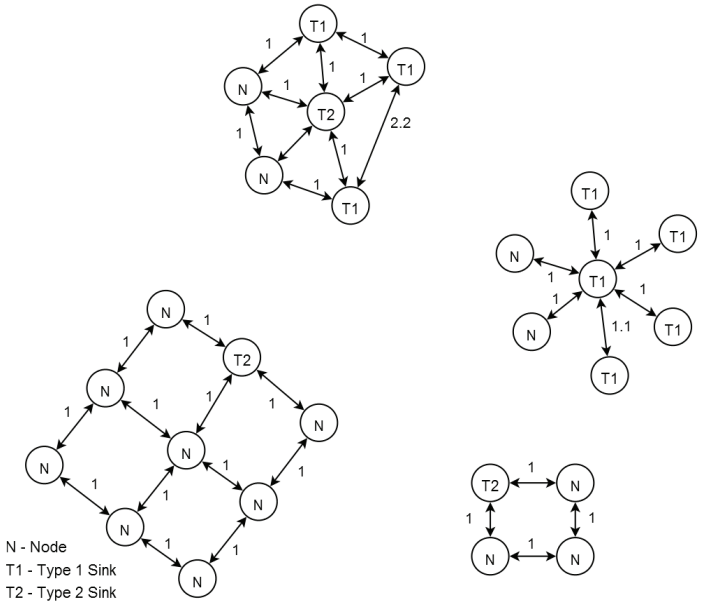

Figure 7.13: Distance graph G_{D3} showing only communication links of 3 distance units or less.

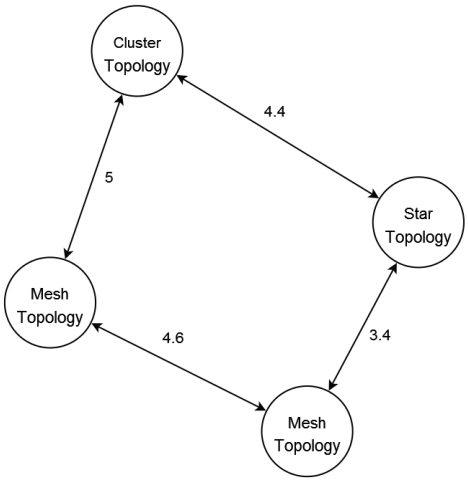

Figure 7.14: Basic communication graph, G', showing the example sensor network at the first level of topological abstraction.

To construct the distance graph, G_D', from G', one distance unit is defined as the minimum edge weight of 3.4 meters and the edge weights are converted from meters to distance units. The graph G_D' is shown in Figure 7.15.

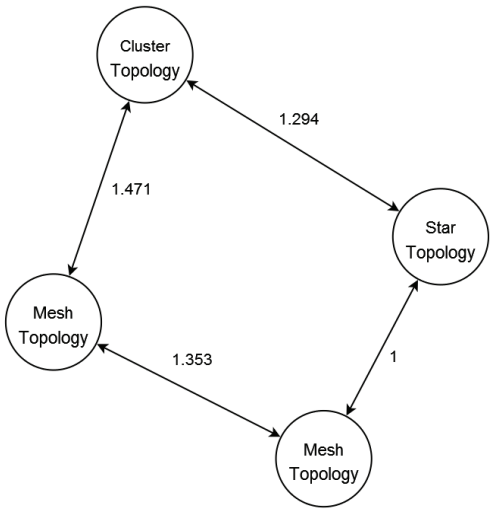

Figure 7.15: Distance unit graph, G_D' with one distance unit equal to 3.4 meters (or feet).

The distance graph G_{D1} is created from G_D' and is shown in Figure 7.16. The two topological entities on the left side are not unconnected making it impossible to cover the graph in Figure 7.16 with base topologies. Therefore, the graph G_{D2}' shown in Figure 7.17 must be employed.

With the distance unit graph, G_{D2}' it is possible to cover all four topological entities using the mesh topology. After applying the mesh topology to graph G_{D2}', a new basic communication graph, G'' can be created by representing the four topological entities with a single mesh topological entity. Because the basic communication graph, G'', contains only one entity no further reduction is possible and the bottom-up construction is complete.

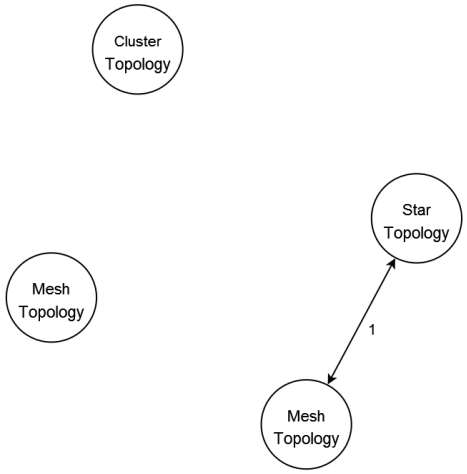

Figure 7.16: Distance graph G_{D1}'.

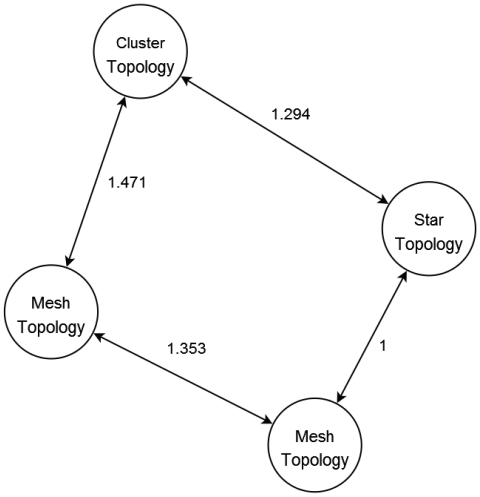

Figure 7.17: Distance graph G_{D2}'.

Searching for topologies, starting with G_{D1} and only searching G_{D2}, G_{D3}, …, if the smaller graph cannot be covered, yields topologies that are spatially close together. The spatial closeness of the entities in each topology produces a Markov process of the topology that more accurately represents the energy consumption of the individual entities contained within the topology, because entities that are closer together tend to work together.

7.2 TOP-DOWN CONSTRUCTION

The process described above can be performed in top-down order. Using the above example, the process to decompose the sensor network using the top-down methodology will be presented. The first step with the top-down methodology is to identify phenomena of interest the sensor network will monitor. Take as an example a small chemical production plant, the footprint of which is shown in Figure 7.18.

In this example, the sensor network must monitor the concentration of ethylene oxide within each of the four buildings and report that concentration to the control building. The phenomenon that will be monitored in this example is the indoor concentration of ethylene oxide.

With this information it is now possible to identify the locations that must be covered by the sensor network. The interior of each of the four buildings must be monitored using a sensor network and each network must be able to communicate with the control building. Based on the building shapes, a mesh network will provide good coverage in the two square shaped buildings, and either a cluster or a star network will provide good coverage in the circular buildings. The top-level topology showing the high level entities and high level communication links is shown in Figure 7.19.

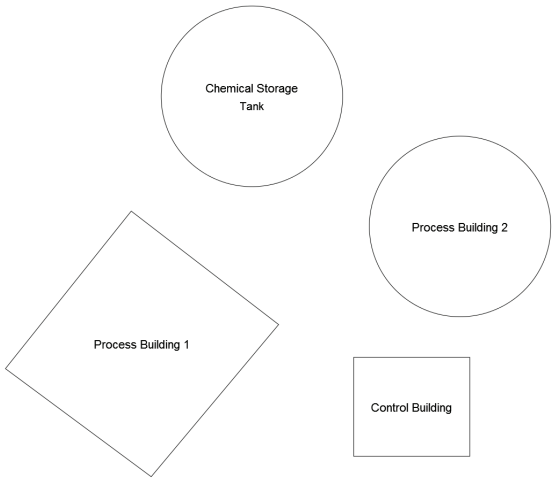

Figure 7.18: Footprint of simple chemical production plant consisting of four buildings.

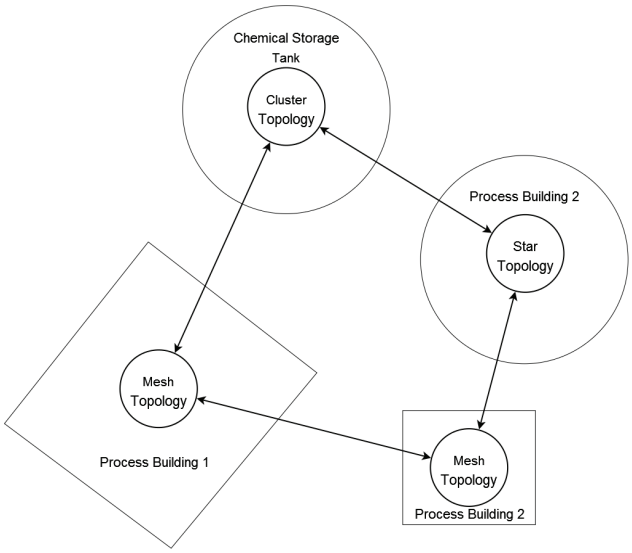

Figure 7.19: Chemical plant with top-level sensor network topology shown.

Each building now has a sensor network topology defined to monitor for ethylene oxide. Now each of the four top-level topologies can be again expanded to a more detailed level and studied in isolation from the other three topologies. Expanding the four topologies to the next level of detail results the sensor network described at the base entity level and this is shown in Figure 7.20.

With the detail to the base entity level, it is now possible to investigate the energy consumption of individual entities in the sensor network. The partition tree is built from the top-down when using the top-down methodology shown in Figure 7.21 and is identical that built using the bottom-up method shown in Figure 7.7.

The top-down methodology provides for a better designed sensor network as the phenomena to be monitored are identified and located first. The sensor network is then designed around the phenomena to provide the needed coverage. Using the top-down method it becomes possible to deploy topologies of entities to monitor the phenomena of interest rather than to try to fit topologies to a set of base networks to monitor the phenomena. Further, in the case of a sensor network with mobile entities the positioning of those entities and their future movements can easily be determined and anticipated based on the desired topology.

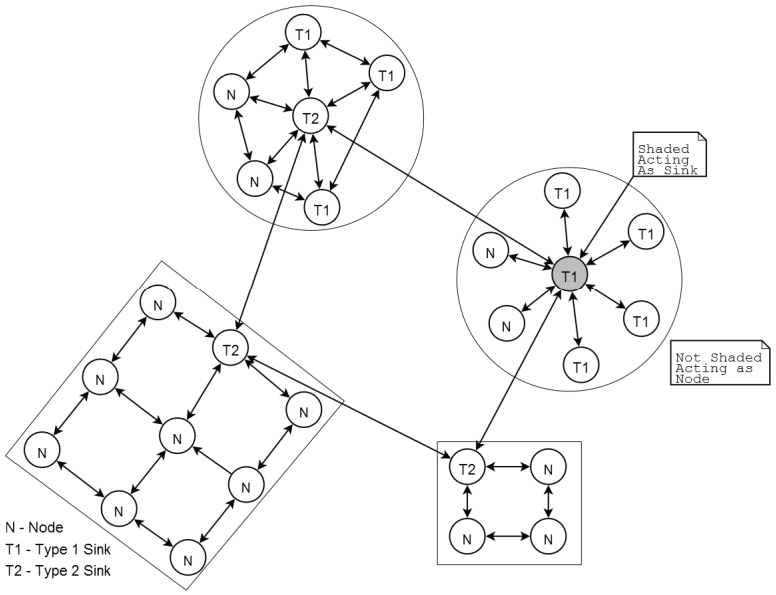

Figure 7.20: Sensor network for the chemical plant to monitor ethylene oxide at the basic entity level of detail.

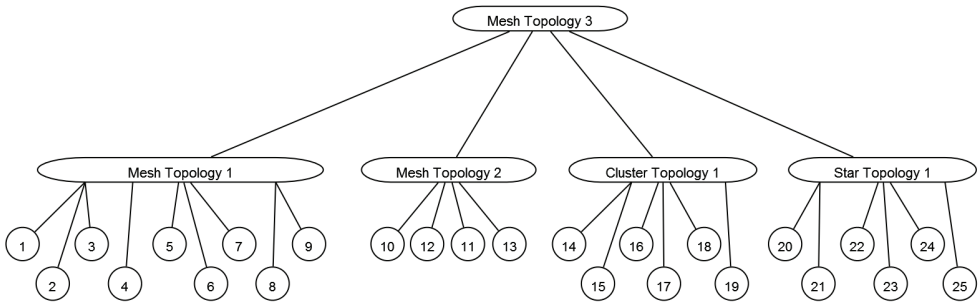

Figure 7.21: Final partition tree constructed using the top-down method of the example sensor network shown in Figure 7.4.

8.0 BASE SENSOR AND RFID NETWORK TOPOLOGIES

In the previous section, topologies were used to group together base entities responsible for performing similar tasks. Grouping the base entities into topological entities reduced the number of entities that need to be simulated resulting in a speedup in evaluation of the energy consumed by the network. This process can be continued and topological entities can themselves be grouped together into topologies. At each stage in this process the number of entities in the simulation is reduced. Sensor and RFID networks are composed of a set of base topologies that perform a similar role as the base entities. Existing sensor or RFID networks can be decomposed into collections of base topologies and new sensor or RFID networks can be built using the base topologies. The individual entities in a base topology could be base entities (nodes or sinks) or topologies containing base and/or topological entities.

Graph theory provides a convenient means to determine the resistance to a breakdown in the communication with a network or topology due to link or entity failures. Unattended sensor networks will most likely experience link or entity failures. Hence, sensor and RFID networks must have resistance to link and entity failures. As shown, each topology can be represented by a graph with the edges being the communication links and the vertices being the entities that make up the topology. The connectivity of a graph is a measure of how many edges or vertices need to be removed to disconnect the graph. In terms of a sensor or RFID network topology, connectivity measures how many communication link or entity failures a topology can sustain before becoming disconnected and failing. The larger the connectivity of a topology, the more failures that must occur before that topology fails.

8.1 MESH TOPOLOGY

There are many different variations of the mesh topology. A fully connected mesh is a topology in which each node is connected to every other node. While the fully connected mesh topology allows for direct communication between any two entities, the number of communication links and, hence, cost grows astronomically. The very large number of links required to implement a fully connected mesh topology limits its use to smaller networks. The ZigBee protocol employs a fully connected mesh, where each ZigBee entity can communicate with any other ZigBee entity within range to increase resistance to link or entity failures [52]. A crossbar mesh or Manhattan topology is another variety of the general mesh topology. The crossbar or Manhattan mesh is laid out as a 2D-grid with each entity communicating with another entity to the north, south, east, and west. In the crossbar mesh topology, each node communicates with at most four other nodes. This type of topology is common in parallel computing and provides a number of independent routes from source to destination [53]. An example of a 3x3 crossbar mesh topology is shown in Figure 8.1.

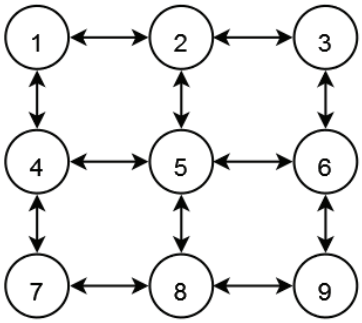

Figure 8.1: A 3x3 mesh topology.

In the crossbar mesh, there are a number of completely independent paths sharing no common edges between two entities in a mesh topology. A fully connected mesh contains the most independent paths between two entities. For example, in the crossbar mesh topology in Figure 8.1 there are two completely independent paths between entities 7 and 3;

Path 1: 7,8,9,6,3

Path 2: 7,4,1,2,3

For the crossbar mesh network topology in Figure 8.1 to become disconnected at least two links would need to fail (i.e. 7-4, and 7-8). Thus, the mesh topology provides a strong resilience in the face a link failures. The presence of multiple paths between two entities enables information to travel through an alternate route if the primary route becomes disconnected. This is important in a sensor network because as entities fail communication between the remaining entities could be adversely affected if alternate paths are not present in the network. As sensor networks must be resilient against failures of a few entities the mesh topologies are popular with sensor networks.

8.2 STAR TOPOLOGY

In a star topology, there is one link between the central entity and all other entities in the star network. Figure 8.2 shows simple example of the star topology. Thus, in a star topology containing n entities, there are (n-1) links.

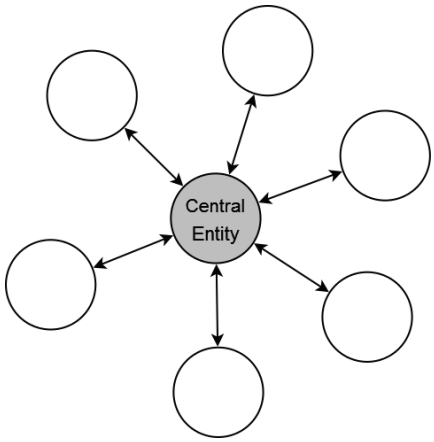

Figure 8.2: General depiction of the star topology, all entities in the star topology are connected to the central entity.

The primary drawback of the star topology is the absence of redundant paths to the central entity. Thus, the star topology has a connectivity of 1. If any link in the star topology fails at least one entity will be isolated from the central entity. A benefit of the star topology is that if any link fails only one entity is disconnected from the star. In other topologies, the failure of a single link could result in the disconnection of multiple entities that were using the failed link to communicate with the central entity. Further, if the central entity is removed, then all remaining entities in the star topology are isolated from the rest of the network.

The star topology has the benefit of easily scheduling communications between the central entity and other entities. The central entity knows exactly how many entities are connected to it and can assign each entity a timeslot or CDMA code to use to transmit their data. This results in a negligible number of message collisions and increased throughput, thus increasing the efficiency of the topology in terms of throughput and number of message collisions. Another advantage of the star topology is that the central entity can more efficiently divide the workload for the required tasks among the entities it oversees. This should result in an increase in the lifetime of the topology. The star topology is one of the topologies available in a ZigBee network [54].

In a sensor or RFID network, the central entity is assumed to connect the star topology it exists in to the rest of the network or to the outside world. The central entity must transmit the data collected from the other members of the star topology to the outside world and must distribute commands and requests received from the outside world to the other members of the star topology. Therefore, the central entity is either a type 1 or a type 2 sink. The other entities can be either nodes or type 1 sinks acting like nodes. Star topologies are a good mechanism for use in networks that employ data fusion because the central entity can fuse the data before sending the data to the rest of the network or the outside world.

8.3 CLUSTER TOPOLOGY

The cluster topology is related to the star topology. The network generated using only the cluster topology contains a set of groups called clusters. Each cluster contains a number of entities within the cluster. Like the star topology, each cluster has a controlling entity. The controlling

entity is responsible for directing the operation of the cluster and for communicating with the larger network outside of the cluster. In the cluster topology, the entity performing the controlling entity role can change over time. The ability to change which entity is acting as the controlling entity provides added functional resilience to entity failures. Within the cluster, individual members communicate directly with each other in addition to the controlling entity. Figure 8.3 shows an example of a single cluster in a larger cluster topology.

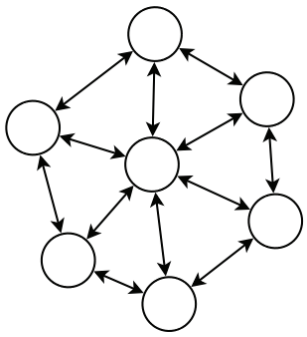

Figure 8.3: Example of a single cluster within a cluster topology.

The cluster in Figure 8.3 contains seven entities, any of which could perform the controlling entity role for the cluster. In a star topology, the central entity controls the rest of the star and performs the energy intensive tasks long-haul communication. Once the central entity in a star fails, the data from the other members of the star are unable to communicate their results to the rest of the network. In the star topology, none of the other members can take over the responsibilities of the central entity. The ability to pass the role of controlling entity between the members of the cluster enables the cluster as a whole to function longer than a comparable star topology because multiple entities can be the controlling entity.

The cluster topology has a greater connectivity than the star topology and provides a greater resilience to a communications link failure than the star topology. In the star topology, each time a link fails, one entity is disconnected from the central entity. However, in the cluster topology, if a link fails between two entities there is the chance that the traffic can be rerouted through the other links.

8.4 TREE TOPOLOGY

A tree topology organizes the entities into a structure resembling the branches of a tree. A tree is simply a graph containing no cycles. Some examples of tree topologies are a binary tree, and an n-ary tree. Single paths are common in sensor or RFID networks, but they fall under the category of a tree topology because all paths eventually meet at a common node - usually a sink. Thus, a sensor or RFID network consisting of numerous single paths from nodes to sinks is really a network of tree topologies with each tree rooted at a sink.

The tree topology is useful in cases where data traveling to the sink can be fused or processed at each hop on the message path. As previously described, data fusion can greatly reduce the amount of information that must be transmitted. Reducing the amount of information that must be transmitted provides a potential energy savings.

Data fusion performs more computational operations on the data, but in exchange, the amount of data transmitted by each entity is reduced. The tree topology is ideally suited to data fusion. At each level, the vertices of the tree wait until receiving data from all children. The parent vertex then fuses their data and their children's data. The fused data is then sent to the higher-level parent entity. Compression and averaging of data are just two examples of data fusion methods.

A simple example of data fusion using a binary tree structure is one in which parent nodes at each level average the temperature readings reported by their children. A simple network of fifteen temperature sensors is shown in Figure 8.4.

Each vertex in the binary tree represents a single sensor and with data flowing in the direction of the arrow. The number inside each vertex is the temperature reading taken at that particular sensor. The number next to the edges is the temperature reading that each vertex sends to its parent. This temperature is the average of the temperatures received from the children and the parent's temperature reading. Because the first level all are leaf nodes, the average sent to each parent is identical to the temperature reading.

Once each parent has received temperature messages from both children, the average temperature to send to the next highest level can be computed. This process is shown in Figure 8.5.

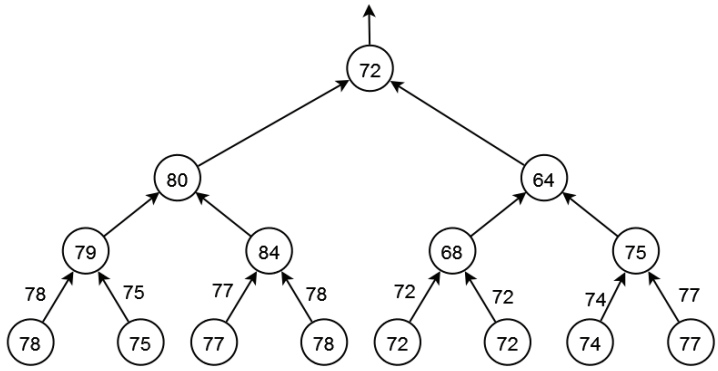

Figure 8.4: Data fusion in a network of temperature sensors arranged in a binary tree topology. All sensor nodes have taken readings and the lowest level (leaf nodes) have transmitted their readings to the next highest level.

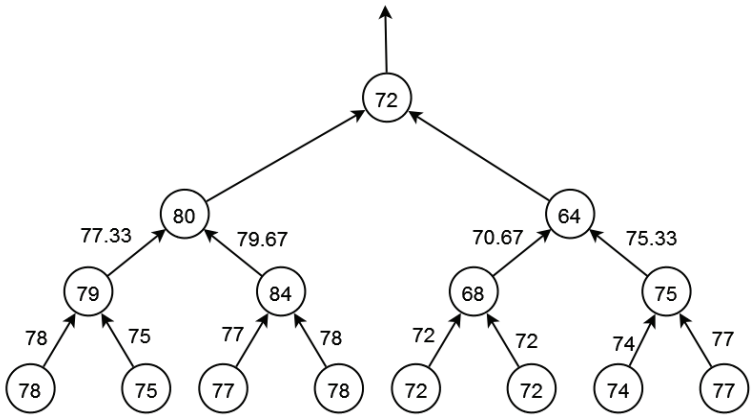

Figure 8.5: The data fusion process started in Figure 8.4 has progressed to the next higher level.

The process continues with the next highest level and is shown in Figure 8.6. At this point only the sensor node that is the root of the tree has yet to contribute data to the average temperature.

Once the sensor node at the root of the tree receives information from the two children, it can add the final temperature reading to the average. The average temperature data is then sent to the next level, which is assumed in this case to be a sink. However, the tree structure could

continue, and the process for the larger tree structure is identical to the process for the smaller tree structure shown here. The completed data fusion process is shown in Figure 8.7.

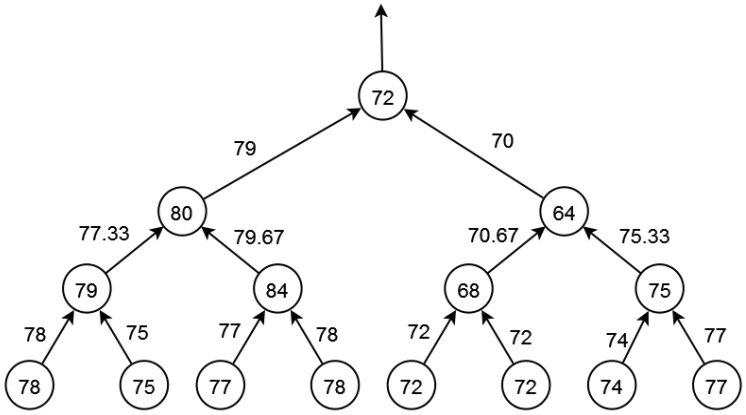

Figure 8.6: The data fusion process has processed from Figure 8.5 with only the sensor node that is the root of the tree remaining.

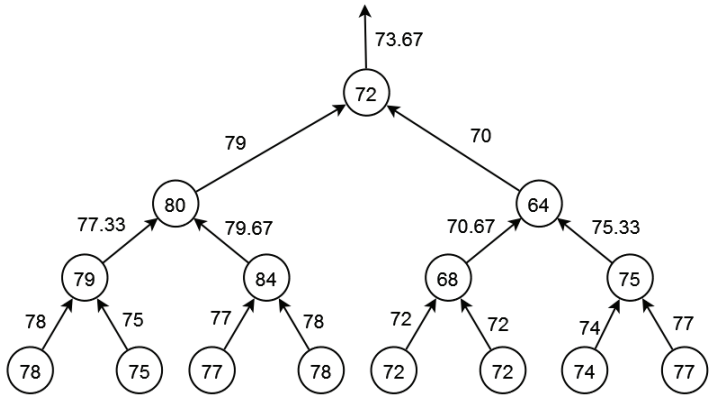

Figure 8.7: The data fusion process is complete and the sensor node at the root of the tree has sent the data to the sink (or next highest level).

In the above example, only a single temperature was sent over each link. Without data fusion each vertex must transmits its own temperature reading as well as the temperature

readings of all descendents. The higher-level vertices must transmit increasingly larger numbers of readings. Thus, data fusion provides substantial savings in the amount of messages sent for a tree consisting of many levels. Without data fusion, a total of 49 messages are sent as compared to only 15 messages when data fusion is employed.

8.5 SINGLE LINK TOPOLOGY

The single link topology is the basis topology from which all other topologies can be constructed. The single link has a connectivity of 1 and is extremely vulnerable to failure in the face of a single link or entity failure. This vulnerability to failure is magnified if the single link topology is used to connect two larger topologies (i.e. to connect two 3x3 mesh topologies). The single link topology is included for completeness and to allow existing networks to be completely covered by a set of topologies. Usually the single link can be absorbed into a tree rooted at a sink entity. In some cases, only a few single links exist in a sensor or RFID network and the single link topology enables those links to be covered without reconstructing another topology. The communication link between the two entities can be unidirectional or bidirectional, and an example of a single link topology is shown in Figure 8.8.

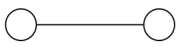

Figure 8.8: Example single link topology.

The single link topology is not meant to be used on a large scale, because it does not provide much reduction in the number of entities in the system. The single link topology should only be employed when absolutely necessary to cover an existing sensor or RFID network with a topological entities when using the bottom up method. When the designer is using the top-down design methodology, the single link topology should never be used.

9.0 TOPOLOGICAL ENTITIES GENERATION ALGORITHM

The method developed in this work uses topological entities to group multiple entities into a single entity with the goal of reducing the number of simulation entities present in the network. The top-down and bottom-up construction methods can then be used to tile the network with the topological entities, but those entities must first be defined. The following ten-step algorithm can be used to define the topological entities present in a given sensor or RFID network.

First, the base level entities making up the network must be identified. Second, the tasks that each base level entity must perform must be identified. These tasks will be used to construct the Markov processes describing each type of base level entity and to assist in identifying interactions with other entities. Third, a Markov process describing each base level entity based on the tasks it performs must be developed. Expressions for the probability and reward matrices must be obtained. These expressions define the parameters that describe each base level entity. Fourth, using the tasks the base level entities perform, the interactions between the base level entities can be determined. During this step, a basic communication graph can be developed to assist the designer, but that is not necessary. Fifth, using the interactions between the base level entities and the tasks the base level entities must work together to perform, the first level topological entities can be defined. The first level topological entities group the base level entities together based on interactions between the base level entities. Several different types of first level topologies are possible within the same network because portions of the network may have different responsibilities.

With the first level topologies, the network can now be tiled using these topologies. In the top-down approach, the network is built from the top down starting with larger topological entities and working down using increasingly smaller topological entities, until at the final step base level entities are used. In the bottom-up approach the network is already deployed, so the basic communication graph must be constructed and searched for topological groups. With the

fundamental topological entities identified existing algorithms can be used to search the basic communication graph to identify these topological entities. These algorithms form the basis for CAD tools for integrated circuit design [46]. Once a topological entity is identified it is replaced with the topological entity. This replacement reduces the order (number of entities) of the basic communication graph. Larger topological entities can be constructed using the first level topological entities. Again, in the bottom up approach the larger topological entities can be identified and replaced with a single topological entity by employing existing algorithms.

The following steps describe how to identify larger topological entities constructed from smaller topological entities. Sixth, the tasks that each type of topological entity performs must be identified. These tasks can be derived from the tasks of the entities within the topological entity and the interactions between entities within the topology. Seventh, using the tasks identified in step six, the Markov process and expressions from the probability and rewards matrices can be developed. To obtain expressions for the rewards matrix the task can be broken down into a sequence of subtasks that the entities within the topology must perform to complete the larger topological entity task. These expressions will define the parameters that describe the operation and energy consumption of the topological entity. Eighth, the interactions between topological entities must be identified. Again, constructing a basic communication graph of the topological entities is helpful but not necessary. These interactions will be used to define the next higher level topological entities. The network can now be tiled with these topological entities.

Ninth, the next higher level topological entities are identified based on the interactions between the topological entities at the current level. Tenth, by repeating steps six through nine, the topological entities continue to grow in size and fewer topological entities are required to cover the network after each pass through steps six through nine. In the ideal case, the topological entity will grow large enough to cover the network using a single topological entity. The examples in Sections 10 and 11 will illustrate the use of the method developed in this work.

The algorithm for the procedure described above is given below.

Step 1: Identify the base level entities that make up the network

Step 2: Identify the tasks each base level entity must perform

Step 3: Develop a Markov process and expressions for the probability and rewards matrices for each base level entity using the tasks from Step 2

Step 4: Identify the interaction between base level entities using the tasks the base level entities perform

Step 5: Identify the first level topological entities from the interactions and tasks identified in Step 4 and Step 2

Step 6: Identify the tasks each topological entity must perform

Step 7: Develop Markov processes and expressions for the probability and rewards matrices for each type of topological entity

Step 8: Identify the interactions between topological entities using information from Step 6

Step 9: Identify the next higher level topological entities based on information from Step 8 and Step 6 – If new topological entities are identified Repeat Steps 6 through 9 for those topological entities

Step 10: If number of topological entities to tile the network is too large GOTO step 6, OTHERWISE Stop

10.0 ISO 18000-7 RFID NETWORK EXAMPLE

Current commercially available RFID tags generally fall into one of two categories, passive tags or active tags. Passive tags have no on-board power source and are powered by the reader's interrogation signal. Passive tags communicate using backscatter and generally contain only a unique identifier and a few bytes of additional information. Active tags have an on-board power source, usually a battery, which powers a processor, an active receiver, and an active transmitter. Active tags may even include sensors or other similar devices and can communicate over greater distances than passive tags [55-58]. The lifetime of an active tag is limited to the lifetime of the active tag's on-board power supply (battery). Conserving energy is critical for active tags because a reduction in energy consumption yields an increase in lifetime.

An International Organization for Standardization (ISO) has produced the standard ISO 18000-7, describing the physical, MAC, and communication protocol for active RFID tags communicating at 433 MHz. An ISO 18000-7 network consists of two types of entities, RFID tags, and ISO 18000-7 readers. In the ISO 18000-7 standard, communication is always initiated by the reader. Tags send a reply only in response to a reader's inquiry. Communication only occurs between tags and readers because there is no tag to tag communication. Tags receiving messages from other tags simply discard those messages.

The ISO 18000-7 standard defines two types of communication. The first type is communication between a reader and tag(s). The tags communicate only with readers and only in response to a reader's inquiry and the reader always initiates the communication (reader talks first). The second type of communication is reader to reader and this may be achieved using an external network such as an internal LAN to link the readers. ISO 18000-7 assigns each reader a unique ID, which the tag includes in the reply to that reader's inquiry [59]. It is possible that the reader to which the reply is addressed did not receive that reply, but that another reader did. Using reader-to-reader communication, the reader receiving the reply can forward that reply to

the reader specified in the reader ID field of the reply. The reader-to-reader communication increases the reply success rate of the system. The readers are assumed to be connected to the facility's power supply, and their energy consumption is not an issue.

Two types of commands are defined by ISO 18000-7, broadcast and point-to-point. The type of command is distinguished by a flag bit in the inquiry sent from the reader [59]. Broadcast commands are not addressed to a particular tag, and any tag receiving a broadcast command from a reader will send a reply. Three commands are defined as broadcast commands; 1) collect the tag ID; 2) collect the tag ID with some data; and 3) collect the tag ID and the user ID [59]. The remaining eleven commands are point-to-point commands. Point-to-point commands contain a tag and group ID in the command packet, and only the tag with a matching tag or group ID will respond to the command [59]. Point-to-point commands are primarily used to control the tag (i.e. set tag password, lock tag, unlock tag, put tag to sleep, etc.) [59].

Battery powered devices are often put to sleep to conserve energy and extend lifetime. The ISO 18000-7 standard allows for tags to go to sleep on their own or be put to sleep by a reader and defines a wake-up tone, a 30 kHz square wave, to wake-up all tags receiving this tone [59]. The wake-up tone is used to wake tags up from the sleep state so they can receive the message. The sleep state allows the tags to power down to conserve energy during dormant periods.

An ISO 18000-7 network falls under the respond to commands category of sensor and RFID networks. Hence, the network is constructed from a number of tree or star topologies with a reader being the central entity or the root in each case. It is possible for a tag to be within communication range of more than one reader and belong to multiple tree or star topologies. In this case, the tag would be a member of the star or tree topology for each reader such that the tag in range.

Markov processes will first be developed to model the energy consumption of a generic ISO 18000-7 active RFID tag and for a generic ISO 18000-7 reader. The Markov processes will be customized to each tag and reader (number of each type of command sent/received will be specified) and the energy consumption of the network will be computed by evaluating the Markov process for each entity (each tag or reader). The energy consumption for each entity will be obtained by evaluating the Markov process for that entity. Once the Markov processes for all base entities have been evaluated the energy consumption of the entire ISO 18000-7

network is simply the sum of the energy consumption found for each entity. The time required to evaluate the network and determine the energy consumption will be recorded. The time required to evaluate each model will be used to determine how well the method scales with respect to the size of the network being analyzed.

The entities in the example network will then be grouped into topologies. The energy consumption of the network will then be found using topological entities and compared to the energy consumption found using the base entities. Each topology will consist of a single reader and all tags that are within range of that reader. Using this topology will reduce the number of simulation entities from the number of tags plus the number of readers to simply the number of readers. In the majority of RFID systems, the number of tags far exceeds the number of readers. A Markov process will be developed to represent the energy consumption of this topology. The Markov processes for the individual tree (or star) topologies will model the tasks that the topology performs. Again, the energy consumption of the network will be determined by evaluating the Markov process associated with each topology and then summing the results. The time required to determine the energy consumption of the network will also be recorded.

Next, the topological entities will be grouped into a single topological entity, and the energy consumption of the network found using this single topological entity. The energy consumption of the network will be found using this topology and the energy consumption obtained using the base entities and smaller topology will be compared. In addition, the time required to evaluate the network using the base entities and the topological entities will be compared to determine the speedup provided by the topological entities.

10.1 EXAMPLE ISO 18000-7 NETWORK

Conceptually, this network contains three levels. The first level is the individual tags, the second level is the interaction of each reader with all tags within the reader's range, and the third level is the reader-to-reader communication. The first level contains the largest number of entities, having an entity for each tag and reader requiring the longest execution time to evaluate.

Topological entities are introduced at the second level in order to reduce the number of simulation entities. At the second level, the readers and tags are grouped together to form either

a star topology or a tree topology with the reader playing the role of the central entity in the star topology or the root in the tree topology. The tree topology can be used with the restriction that it contains a single root, the reader, with all tags within range of the reader (root) are children of the root. Grouping a reader and associated tags together reduces the number of simulation entities in the system providing a speedup in evaluation of energy consumption. A simple example of the star and tree topologies for an ISO 18000-7 network is shown in Figure 10.1.

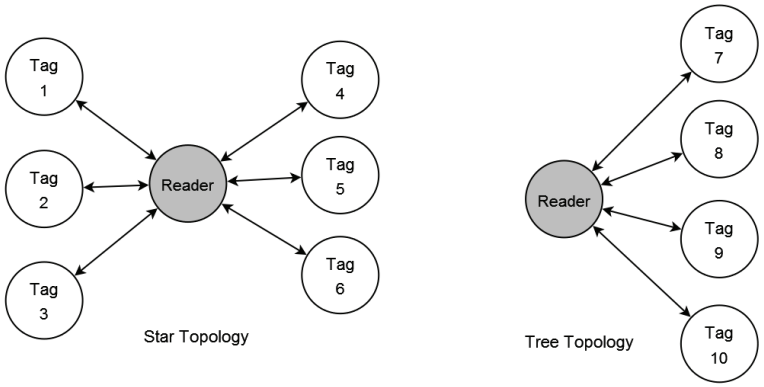

Figure 10.1: Topologies used to group a single ISO 18000-7 reader and associated tags together.

The cluster, and mesh topologies cannot be used as topologies at the second level because the reader and tag communication is point-to-point and there is no tag-to-tag communication. The cluster and the mesh topologies imply the possibility of tag-to-tag communication to relay messages to and from the reader. The single link topology is used only when a reader is within range of at most one tag, in all other cases the tree or the star topology can be applied to reduce the number of simulation entities. The single link topology will not be used in this example because everywhere where the single link topology could be employed a star or tree topology could also be employed. The star or tree topologies are the better choice because they group more entities together than the single link topology.

It is possible that a tag is within range of more than one reader. A tag that is within range of more than one reader will be duplicated in the topology based around each reader with which it can communicate.

The third level is the highest level of abstraction looking only at the readers in the network and has the fewest number of entities. At this level, every entity can represent multiple readers and all tags within those readers' range. Readers may communicate with other readers so all topologies are possible for the reader-only portion of the network.

10.2 ANALYSIS OF ISO 18000-7 NETWORK

The analysis of the energy consumption of the ISO 18000-7 example case is presented in this section. The subsections in this section cover the steps in the algorithm to identify the topological entities for the example network. The energy consumption found for each of the three levels, base level entities, single reader and associated tags topological entities, and multi-reader topological entity are included in the relevant subsections.

10.2.1 Step 1: Identification of the Base Entities

The first step, step 1, of the algorithm is to identify the base entities making up the network. The base entities in the ISO 18000-7 example network follow from the ISO 18000-7 specification. In an ISO 18000-7 network two types of base level entities exist, tags and readers. The tags provide information to the readers. The readers link the network with the outside world.

10.2.2 Step 2: Identification of Tasks of the Base Level Entities

The tag must perform only one task, reply to received commands. The reader must perform five tasks. First, the reader must send inquiries to the tags to solicit information. Second, the reader must listen for tag replies. Third, the reader may need to communicate with another reader, either to forward a reply or receive a reply forwarded from another reader. Fourth, the reader must communicate with the outside world (internet or company's private LAN) to receive instructions and transfer data received to the outside world. Fifth, the reader must process the reply.

A Markov process describing the energy consumption of an ISO 18000-7 RFID tag can be developed based on the single task the tag performs. The reader may send any of eighteen possible commands to the tag. These commands are listed and given abbreviations in Table 10.1.

Table 10.1: ISO 18000-7 commands and abbreviations.

Command	**Abbreviation**	**Command**	**Abbreviation**
Collection	Co	Read Owner ID	OIDR
Collection with Data	CD	Write Owner ID	OIDW
Collection with User ID	CID	Firmware Revision	FR
Sleep	SL	Model Number	MN
Status	ST	Read Memory	RM
Read User ID Length	IDLR	Write Memory	WM
Write User ID Length	IDLW	Set Password	SP
Read User ID	UIDR	Read Set Password Protect	SPP
Write User ID	UIDW	Unlock	UNL

The ISO 18000-7 used in this example consists of N_{Tag} tags and N_{Reader} readers. The values of these parameters are given in Table 10.2.

Table 10.2: Number of entities in example ISO 18000-7 network.

N_{Tag}	20
N_{Reader}	4

The example network consists of twenty RFID tags and four RFID readers. All readers are identical to each other and all tags are identical to each other. Each reader can communicate with any tag within its range. A tag can communicate with (reply to) any reader that is within its communication range. Each tag is assigned a unique ID number for discussion and a 'T' to indicate it is a tag prefixes this ID number. Similarly, readers are assigned a unique ID number starting with a 'R' followed by the unique ID number. The example network is shown below in Figure 10.2 where the grey circles represent readers and the white circles represent tags. Each reader has a communication range denoted by a dotted circle in Figure 10.2 and the reader can communicate with any tag within the circle, as shown in the basic communication graph of the example ISO 18000-7 network in Figure 10.3.

Each reader sends out a number of inquiries each day. The number of each type of inquiry sent by each of the four readers per day is listed in Table 10.3.

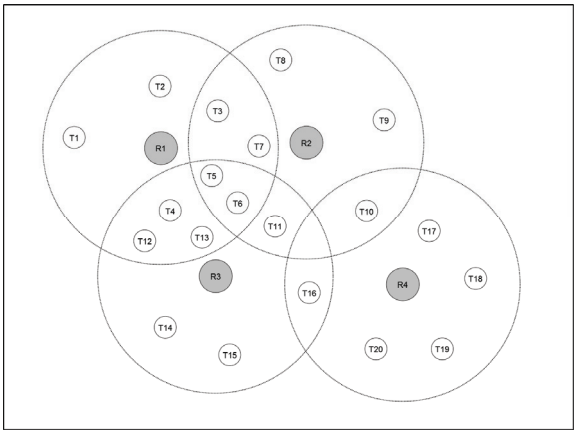

Figure 10.2: Example ISO 18000-7 network.

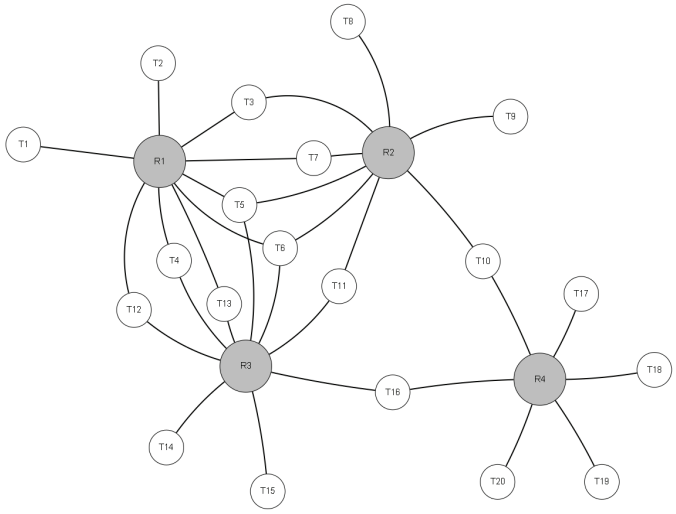

Figure 10.3: Basic communication graph of the ISO 18000-7 example network.

Table 10.3: Number of each inquiry sent by each of the readers in the ISO 18000-7 example.

Inquiry	R1	R2	R3	R4
Co	23	17	15	25
CD	8	9	14	20
CID	6	8	12	24
SL	12	15	17	9
ST	10	13	8	15
IDLR	0	0	0	0
IDLW	0	0	0	0
UIDR	1	1	3	0
UIDW	0	0	0	0
OIDR	1	4	6	3
OIDW	0	0	0	0
FR	1	1	3	1
MN	1	4	2	0
RM	4	6	9	1
WM	1	3	4	5
SP	0	0	0	0
SPP	0	0	0	0
UNL	0	0	0	0

10.2.3 Step 3: Development of Markov Process and Probability and Reward Matrices for the Base Level Entities

The tasks that the tag must perform (identified in Section 10.2.2) form the basis for constructing the Markov processes for the tag and reader. The Markov processes and expressions for the probability and reward matrices for the base level entities (tag and reader) are determined in step 3 of the algorithm. The development of the Markov process and expressions for the probability and rewards matrices for the ISO 18000-7 tag is presented in Section 10.2.4. The development of the Markov process and expressions for the probability and rewards matrices for the ISO 18000-7 reader is presented in Section 10.2.5. The energy consumption for a period of 1 day for each of the base level entities is presented at the end of each of the following two subsections. The energy consumption is compared against the energy consumption found using the topological entities presented in later sections.

10.2.4 Step 3 for the Single ISO 18000-7 Tag

The tasks that the tag must perform (identified in Section 10.2.2) form the basis for constructing the Markov processes for the tag, and are repeated for clarity. The tag must perform only one task, reply to received commands. This task can be broken into five steps. First, the tag enters the sleep state remaining dormant between inquiries to conserve energy, and the tag must listen for the wake-up tone. Ideally the tag will spend the majority of it's lifetime in the sleep state and in this example, the tags will be put to sleep after the reader has completed the inquiry, and the reader will transmit the wake-up tone before issuing an inquiry. Second, the tag must listen for messages after being wakened by the wake-up tone. Third, the tag must receive, decode, and process received messages. Manchester encoding is used in ISO 18000-7 networks to provide a communications channel with less errors [59]. An active receiver converts the RF signal to a digital signal. Fourth, the tag must decode the Manchester encoded message, verifying that the message is addressed to itself and if so, generate the appropriate reply. The fifth task that the tag must perform is to transmit the reply to the reader. The data portion of the reply must be encoded using Manchester encoding. An active transmitter will convert the reply from a digital

signal into an RF signal. The Markov process for the ISO 18000-7 RFID tag is illustrated in Figure 10.4.

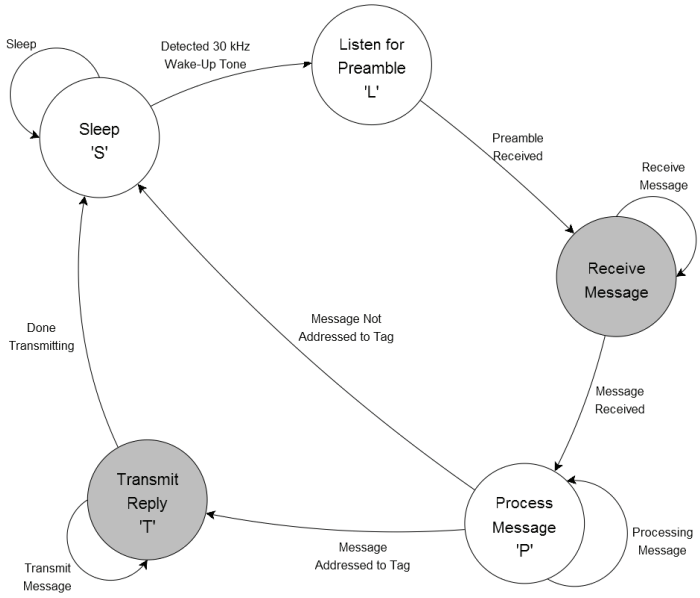

Figure 10.4: Markov process describing the energy consumption of an ISO 18000-7 RFID tag based on the tasks performed.

It is assumed that the tags will immediately go back to sleep after sending the reply. A probability and a reward matrix are associated with the Markov process in Figure 10.4. The probability matrix, P, will have the form,

$$P = \begin{bmatrix} P_{SS} & P_{SL} & 0 & 0 & 0 \\ 0 & 0 & P_{LR} & 0 & 0 \\ 0 & 0 & P_{RR} & P_{RP} & 0 \\ P_{PS} & 0 & 0 & P_{PP} & P_{PT} \\ P_{TS} & 0 & 0 & 0 & P_{TT} \end{bmatrix} \tag{10-1}$$

The reward matrix will have the following format shown in (10-2).

$$R = \begin{bmatrix} R_{SS} & R_{SL} & 0 & 0 & 0 \\ 0 & 0 & R_{LR} & 0 & 0 \\ 0 & 0 & R_{RR} & R_{RP} & 0 \\ R_{PS} & 0 & 0 & R_{PP} & R_{PT} \\ R_{TS} & 0 & 0 & 0 & R_{TT} \end{bmatrix} \quad (10\text{-}2)$$

The period of one day will be used to determine the transition probabilities and the rewards will represent the energy consumed for each transition (action). The day can be divided into a number of periods, with each period having a length τ. In this case, τ is 2.5 ms (milliseconds). The 2.5 ms (milliseconds) period was chosen as that is a good approximation for the time for the shortest task to be performed - processing one byte of a received message. The time period τ allows the determination of the energy consumed for the reward matrix. The total number of periods, T, in one day ($8.64*10^7$ milliseconds in one day) is given by (10-3).

$$T = \frac{8.64*10^7}{\tau} \quad (10\text{-}3)$$

For τ equal to 2.5 ms (milliseconds), there are a total of 34,560,000 τ-periods within one day.

First, the transition probabilities for the sleep state will be determined. The number of times per day that the tag is interrogated, I, is used to determine the transitions starting at the sleep state. It is assumed that the tag wakes up after observing the 30 kHz wake-up tone for, W, continuous τ-periods, and also observing that every interrogation is successful in waking up the tag. In this case, W, is equal to 4, requiring that the 30 kHz wake-up tone must be observed for 10 ms (milliseconds) for the tag to transition from state S to state L. The number of possible S to L transitions, denoted by, T_{SL}, is found using (10-4).

$$T_{SL} = \frac{8.64*10^7}{W*\tau} \quad (10\text{-}4)$$

Using the values for τ and W previously defined, T_{SL} is 8,640,000. The number of times per day that each of the eighteen commands is received by the tag are input parameters into the

model and are denoted by, I_{xx}, where, xx, is the abbreviation for each command found in Table 10.1. The total number of messages received by a tag is the sum of the, I_{xx}, variables and is denoted by, I. The probabilities P_{SL} and P_{SS} are found using (10-5) and (10-6) respectively

$$P_{SL} = \frac{I}{T_{SL}} \tag{10-5}$$

$$P_{SS} = 1 - P_{SL} \tag{10-6}$$

The tag, when in state L, is listening for the preamble and sync pulse of an ISO 18000-7 command. Once the preamble and sync pulse are detected, the tag will transition into the receiving message state.

The reader sends the wake-up tone for 2.5 seconds, and the tag wakes up and enters state L after observing the wake-up tone for 10 ms [59]. Thus, the tag must listen for the preamble for approximately T_{Wake} seconds. In this example T_{Wake} is 2.49 seconds. The transition probability from state L to state the receive message state, R, is , P_{LR}, is given by (10-7).

$$P_{LR} = 1 \tag{10-7}$$

Some commands have a variable length based on the parameters associated with each command. For those commands with variable length, there is a transition back into the receiving message state, state R, to account for the variable length. In the ISO 18000-7 standard, the only commands having a variable length are the write memory and write user ID commands [59]. The base length of each of the eighteen commands includes all non-variable portions of the message are denoted as, L_{Mxx}, where, xx, is the abbreviation for the one of the commands in Table 10.1. For those messages having variable length components the length of those components is denoted by, L_{Exx}, and again, xx, is the abbreviation for the one of the commands in Table 10.1. For commands without variable length parameters, L_{Exx}, equals zero.

The probability that the tag stays in the receive message state (still more bytes of message to receive), is given by (10-8).

$$P_{RR} = \left(\frac{\sum_{\forall xx} \left(I_{xx} \cdot \left(L_{Mxx} + L_{Exx} - 1 \right) \right)}{\sum_{\forall xx} \left(I_{xx} \cdot \left(L_{Mxx} + L_{Exx} \right) \right)} \right) = 1 - \left(\frac{\sum_{\forall xx} \left(I_{xx} \right)}{\sum_{\forall xx} \left(I_{xx} \cdot \left(L_{Mxx} + L_{Exx} \right) \right)} \right) \tag{10-8}$$

The probability that the tag receives the last byte of the inquiry and transitions from state R, to the process message state, P, is given by (10-9).

$$P_{RP} = 1 - P_{RR} \tag{10-9}$$

The tag, in state P, processes the message and determines what action, if any, to take. The received message could be either a broadcast message or a point-to-point message. For a broadcast message all tags receiving that message will respond, while for a point-to-point message only the specified tags will respond. The probability that the received message is a broadcast command is denoted by, M_B, and M_P, denotes the probability that the command is point-to-point and addressed to the tag. It is assumed that the tag can determine if the message is a broadcast or point-to-point command, checking the address in the case of a point-to-point command, within, W_P, τ-periods. In this example, W_P, is assumed to be 1. The tag requires time, T_P, to process each byte of the command and each byte of the reply. Hence, the probability that the tag remains in state P is given by (10-10).

$$P_{PP} = \left(\frac{\sum_{\forall xx} \left(I_{xx} \cdot \left(L_{MRxx} + L_{ERxx} - 1 \right) \right)}{\sum_{\forall xx} \left(I_{xx} \cdot \left(L_{MRxx} + L_{ERxx} \right) \right)} \right) = 1 - \left(\frac{\sum_{\forall xx} \left(I_{xx} \right)}{\sum_{\forall xx} \left(I_{xx} \cdot \left(L_{MRxx} + L_{ERxx} \right) \right)} \right) \tag{10-10}$$

The probability of transitioning from state P to state T is simply,

$$P_{PT} = 1 - \left(P_{PP} \right) * \left(\frac{\sum_{xx \in B} I_{xx}}{I} + M_P * \frac{\sum_{xx \in PtP} I_{xx}}{I} \right) \tag{10-11}$$

The probability that the message is not addressed to the tag and the tag returns to sleep is given by (10-12).

$$P_{PS} = 1 - \left(P_{PT} + P_{PP}\right) \tag{10-12}$$

where, B, denotes the set of all broadcast commands, and PtP denotes the set of all point-to-point commands.

The tag must transmit a reply to all eighteen commands, and the replies have varying lengths. Two of the eighteen replies, read user ID and read memory, have variable length portions [59]. The base lengths of each of the eighteen replies includes all non-variable portions of the message and are denoted as, L_{MRxx}, where, xx, is the abbreviation for the one of the commands in Table 10.1. For those replies having variable length components, the length of those components is denoted by, L_{ERxx}, again, xx, is the abbreviation for the one of the commands in Table 10.1. For replies without variable length parameters, L_{ERxx}, equals zero. The probability that the tag must continue to transmit and remains in the transmit state is given by (10-13).

$$P_{TT} = \left(\frac{\sum_{\forall xx} \left(I_{xx} \cdot \left(L_{MRxx} + L_{ERxx} - 1 \right) \right)}{\sum_{\forall xx} \left(I_{xx} \cdot \left(L_{MRxx} + L_{ERxx} \right) \right)} \right) = 1 - \left(\frac{\sum_{\forall xx} \left(I_{xx} \right)}{\sum_{\forall xx} \left(I_{xx} \cdot \left(L_{MRxx} + L_{ERxx} \right) \right)} \right) \tag{10-13}$$

The probability of the tag returning to sleep after transmitting the last byte of the reply transitioning from state T to state S is given by (10-14).

$$P_{TS} = 1 - P_{TT} \tag{10-14}$$

The reward matrix will now be determined. The reward for each transition in the Markov process represents the energy consumed to perform the particular operation the transition represents. The following definitions will be used to determine the energy consumption for each transition. The reward matrix has the following format as shown in (10-15).

$$R = \begin{bmatrix} R_{SS} & R_{SL} & 0 & 0 & 0 \\ 0 & 0 & R_{LR} & 0 & 0 \\ 0 & 0 & R_{RR} & R_{RP} & 0 \\ R_{PS} & 0 & 0 & R_{PP} & R_{PT} \\ R_{TS} & 0 & 0 & 0 & R_{TT} \end{bmatrix} \tag{10-15}$$

The reward for staying in the sleep state, S, is simply the power consumed by the tag in sleep mode multiplied by the time for a possible transition into state L as shown in (10-16).

$$R_{SS} = PW_{SS} * \tau \tag{10-16}$$

where, PW_{SS}, denotes the tag's power consumption while in the sleep state.

Likewise, the reward for transitioning from state S into state L is simply.

$$R_{SL} = PW_{SL} * W * \tau \tag{10-17}$$

Where the power consumption of the tag while waking up from sleep mode during the transition from state S to state L is represented by, PW_{SL}. The power consumption of the tag while listening for and receiving the preamble is denoted by, PW_{LL}. The reward for transitioning to the receive message state after detecting a preamble and sync pulse is simply:

$$R_{LR} = PW_{LL} * T_{Wake} \tag{10-18}$$

The tag will remain in the receive message state until the entire command has been received. The term, PW_R, denotes the power consumed by the tag while receiving a message, and the term, T_B, denotes the time required to receive one byte of the message. The energy required to receive one byte of the command is given by (10-19).

$$R_{RR} = PW_R * T_B \tag{10-19}$$

Upon receiving the last byte of the command, the tag will transition into the processing message state. The energy consumed during this transition is equal to that of receiving one byte of the command. Hence, R_{RR}, and R_{RP} are equal.

$$R_{RP} = R_{RR} \tag{10-20}$$

The tag must process each byte of the message and then generate each byte of the reply. The tag requires time, T_P, to process one byte of the command or to generate one byte of the reply and consumes, PW_P, Watts while processing the message. The energy consumed for each byte processed is simply.

$$R_{PP} = PW_P * T_P \tag{10-21}$$

The transition from state P to state S requires the same energy as that from state P back to state P. Hence,

$$R_{PS} = R_{PP} \tag{10-22}$$

The transition from state P to state T represents processing of the final byte of the message or reply. During this time the transmitter wakes up and transmits the preamble. The tag consumes, PW_T, Watts while transmitting information. The time required to transmit the preamble is denoted by, T_{PT}. Hence, R_{PT} is simply:

$$R_{PT} = PW_P * T_P + PW_T * T_{PT} \tag{10-23}$$

Transmitting each byte of the reply requires time, T_B. While transmitting the tag remains in state T, and the reward for remaining in state T is given by (10-24).

$$R_{TT} = PW_T * T_B \tag{10-24}$$

The transition from state T to state S occurs when the final byte of the reply is transmitted, and then the transmitter and processor or controller return to sleep mode. The energy consumption for this transition is given by (10-25).

$$R_{TS} = PW_T * T_B \tag{10-25}$$

The tag developed by Cho and Baek will be used to obtain the power consumption numbers for the tag components [60]. Cho and Baek's tag utilizes an Atmega 128L processor, a XEMICS XE1203F transceiver, and a RICOH Rx5C348A real-time clock [60]. In this example the Atmega 128L processor and XEMICS transceiver will be used for the tag, but the real-time clock will not be part of the tag. The time parameters will be derived by estimating the amount of time each task requires. The ATmega processor operating at 4MHz used by Cho and Baek draws at most 5.5 mA when active and 2.5 mA when idle [60, 61]. The transmitter portion of the XE1203F transceiver draws 40 mA when active, the receiver portion draws 17 mA when active, and the transceiver draws 1 μA (micro-Amps) in when asleep [60, 62]. The tag operates at 3.3 V [60]. The values for the power consumption parameters for the ISO 18000-7 tag are listed in Table 10.4.

Table 10.4: Values of power consumption parameters tag developed by Cho and Baek.

Parameter	Power Consumption (mW)
PW_{SS}	74.2533 mW
PW_{SL}	74.2533 mW
PW_{LL}	74.2533 mW
PW_{R}	74.2533 mW
PW_{P}	74.2533 mW
PW_{T}	150.1533 mW

The time to transmit one byte of encoded data, T_B, is defined as the number of symbols required to transmit one byte of encoded information multiplied the time to transmit one symbol. The ISO 18000-7 standard employs Manchester encoding in order to encode a synchronizing signal with the data reducing errors due to loss of synchronization between the transmitter and receiver [48, 59]. As Manchester encoding requires a transition in the middle of the bit time, transmission of one bit of information actually requires the transmission of two symbols, with each symbol being transmitted for half of the bit time. ISO 18000-7 defines a '1' bit as represented by a low to high mid-bit transition and a '0' bit as a high to low mid-bit transition. The Manchester encoding defined by the ISO 18000-7 standard is illustrated in Figure 10.5.

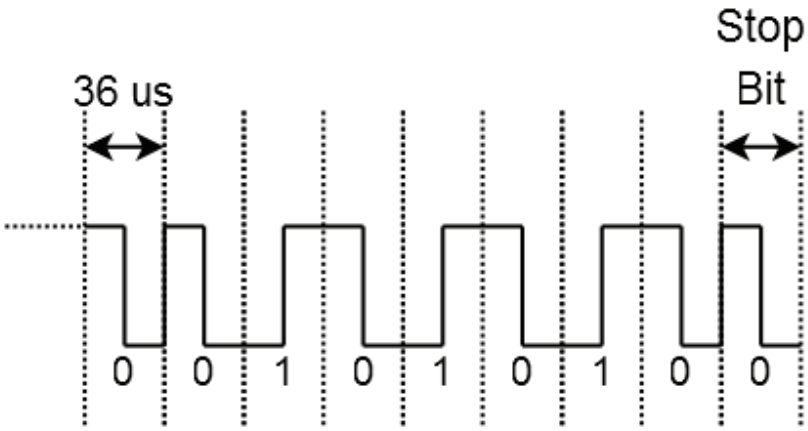

Figure 10.5: Example of Manchester encoding used in ISO 18000-7, the byte '00101010' is illustrated.

Each byte of data in an ISO 18000-7 standard is followed by a stop bit, requiring that each data byte actually contains nine bits (8 data bits plus 1 stop bit) [59]. The data rate in ISO 18000-7 is defined as 27.7 kilobits per second meaning that the symbol rate is twice that, or 55.4 kilobits per second [59]. Hence, the time to transmit one byte, T_B, is found using (10-26).

$$T_B = 9 * \left(\frac{1}{27700} \right) \tag{10-26}$$

Assuming that, N_I, instructions must be executed by the processor to decode and process one byte of information on a processor capable of executing, C_P, instructions per second the time to process a single byte, T_P, can be found using the following equation.

$$T_P = \left(\frac{N_I}{C_P} \right) \tag{10-27}$$

In the ISO 18000-7 examples, N_I, is assumed to be 10,000, and C_P, is assumed to be 4 MHz. Thus, using (10-27), T_P, is found to be 2.5 ms.

The preamble consists of twenty, 60 μs pulses with a 50% duty cycle (30 μs high then 30 μs low) followed by a sync pulse of 96 μs (for the tag reply) [59]. The total time required to transmit the preamble, T_{PT}, is 1,296 μs. The non-power parameters for the ISO 18000-7 tag are listed in

Table 10.5: Non-power parameters for the ISO 18000-7 tag.

Parameter	Value
T_{Wake} (milliseconds)	2490 ms
T_{PT} (milliseconds)	0.1296 ms
T_B (milliseconds)	0.3249 ms
T_P (milliseconds)	2.5 ms
N_I (instructions)	10000
C_P (instructions per second)	4 MHz
τ (milliseconds)	2.5 ms
W (τ periods)	4

With these equations and the user input parameters, the probability and reward values can be calculated for each tag. The values of L_{Mxx}, L_{Exx}, L_{MRxx}, and L_{ERxx} for each of the eighteen commands are listed in Table 10.6.

Table 10.6: Message and reply length parameters.

Command	L_{Mxx}	L_{Exx}	L_{MRxx}	L_{ERxx}
Co	17	0	14	0
CD	20	0	14	32
CID	16	0	14	16
SL	14	0	0	0
ST	14	0	13	0
IDLR	14	0	12	16
IDLW	14	16	12	0
UIDR	14	0	13	0
UIDW	15	0	12	0
OIDR	14	0	15	0
OIDW	14	3	12	0
FR	14	0	13	0
MN	14	0	14	0
RM	18	0	13	46
WM	18	46	12	0
SP	18	0	12	0
SPP	15	0	12	0
UNL	18	0	12	0

With the probability and reward values for each tag the energy consumption of each tag in a one day period is obtained and listed in Table 10.7.

Table 10.7: Energy consumed by each tag in one day.

Tag ID	Energy Consumed (mJ)	Tag ID	Energy Consumed (mJ)
T1	6415639.19 mJ	T11	6415908.60 mJ
T2	6415639.19 mJ	T12	6415876.13 mJ
T3	6415823.54 mJ	T13	6415876.13 mJ
T4	6415876.13 mJ	T14	6415725.57 mJ
T5	6416056.37 mJ	T15	6415725.57 mJ
T6	6416056.37 mJ	T16	6415990.16 mJ
T7	6415823.54 mJ	T17	6415755.40 mJ
T8	6415672.36 mJ	T18	6415755.40 mJ
T9	6415672.36 mJ	T19	6415755.40 mJ
T10	6415937.95 mJ	T20	6415755.40 mJ

10.2.5 Step 3 for Single ISO 18000-7 Reader

The tasks that the tag must perform (identified in Section 10.2.2) form the basis for constructing the Markov processes for the reader, and are repeated for clarity. The reader must perform five tasks. First, the reader must send inquiries to the tags to solicit information. Second, the reader must listen for tag replies. This task is broken into two separate tasks, listening for the preamble that begins every reply and receiving the data portion of the message. Third, the reader may need to communicate with another reader, either to forward a reply or receive a reply forwarded from another reader. Fourth, the reader must communicate with the outside world (internet or company's private LAN) to receive instructions and transfer data received to the outside world. Fifth, the reader must process the reply.

It is assumed that communications across the network connecting the readers to each other and the network connecting the readers to the outside world are on separate networks and do not interfere with the RFID communication. It is assumed that the readers are connected

through an Ethernet (or other infrastructure) to each other and the outside world. The Markov process for an ISO 18000-7 reader is shown in Figure 10.6.

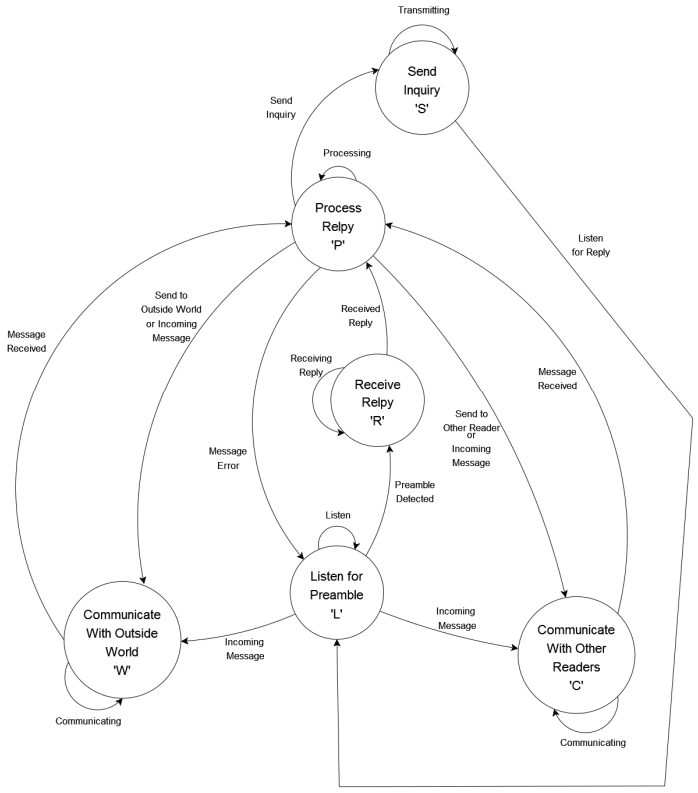

Figure 10.6: Markov process for an ISO 18000-7 reader.

The probability matrix for the ISO 18000-7 reader will have the following form.

$$P = \begin{bmatrix} P_{LL} & 0 & P_{LW} & P_{LC} & P_{LR} & 0 \\ P_{SL} & P_{SS} & 0 & 0 & 0 & 0 \\ 0 & 0 & P_{WW} & 0 & 0 & P_{WP} \\ 0 & 0 & 0 & P_{CC} & 0 & P_{CP} \\ 0 & 0 & 0 & 0 & P_{RR} & P_{RP} \\ P_{PL} & P_{PS} & P_{PW} & P_{PC} & 0 & P_{PP} \end{bmatrix} \tag{10-28}$$

The ISO 18000-7 standard does not define the communication links between readers or between the readers and the outside world. In this example, it is assumed that the readers will periodically receive commands from the outside world that cause them to send an inquiry. Further, it is assumed that the readers will send back information they collect to the outside world and that the readers are preprogrammed to issue some inquiries periodically (i.e. issue a collection command once per hour). The number of each type of inquiry issued by a reader is specified by the user.

Again, the period of one day will be used, and the τ-period of 1.25 ms will be used to obtain some of the transition probabilities. The number of inquiry messages received by the reader from the outside world is denoted by, M_{Ixx}, where, xx, is the abbreviation of the inquiry from Table 10.1. The length of the base reply to each command is denoted by, L_{Mxx}, where, xx, is the abbreviation of the inquiry from Table 10.1. The length of the variable portion of the command is denoted as, L_{Exx}, where, xx, is the abbreviation of the inquiry from Table 10.1. For commands without a variable length portion, L_{Exx}, is 0. The number of messages that the reader receives from the outside world in one day is denoted as M_{OUT}, and the length of these messages is denoted by, L_{OUT}. The transition probabilities for the Communicate with Outside World state, W, will be found first. The probability, P_{WW}, of entering state W from state W is simply.

$$P_{WW} = \left(\frac{\sum_{\forall xx} \left(M_{Ixx} * \left(L_{Mxx} + L_{Exx} - 1 \right) \right) + M_{Out} * L_{Out} - 1}{\sum_{\forall xx} \left(M_{Ixx} * \left(L_{Mxx} + L_{Exx} \right) \right) + M_{Out} * L_{Out}} \right) = 1 - \left(\frac{\sum_{\forall xx} \left(M_{Ixx} \right) + 1}{\sum_{\forall xx} \left(M_{Ixx} * \left(L_{Mxx} + L_{Exx} \right) \right) + M_{Out} * L_{Out}} \right) \quad (10\text{-}29)$$

The only other transition that originates in state W is to state P, hence the probability, P_{WP}, of entering state P from state W is simply.

$$P_{WP} = 1 - P_{WW} \quad (10\text{-}30)$$

The Communicate with Other Readers state, state C, is very similar to the Communicate with the Outside World state. The reader communicates with, N_T, tags, receiving, M_T, messages from the, N_T, tags. Some of the messages received by the reader contain an error and the percentage of messages received by the reader containing an error is denoted by, M_{Error}. The

reader receives a number of replies from other readers, and the average length of reach message received from another reader is, $L_{Avg\text{-}R}$. The reader-to-reader network requires some overhead (i.e. framing) and the length of that extra overhead is, L_{Reader}. When in state C, the probability of remaining in state C is given by (10-31).

$$P_{CC} = \frac{\left(2 * L_{Avg-R} + L_{\text{Reader}} - 1\right) * \left(1 - M_{Error}\right) * \left(M_T - N_T \sum_{\forall xx} \left(I_{xx}\right)\right)}{\left(2 * L_{Avg-R} + L_{\text{Reader}}\right) * \left(1 - M_{Error}\right) * \left(M_T - N_T \sum_{\forall xx} \left(I_{xx}\right)\right)} \tag{10-31}$$

The probability of transitioning from state C into the Process Reply state, state P, is given by (10-32).

$$P_{CP} = 1 - P_{CC} \tag{10-32}$$

The transition probabilities originating in the Receiving Reply state, state R, can be found using similar methods as those originating in states W and C. The number of replies a reader receives is denoted by, I_{xx}, where, xx, is the abbreviation of the inquiry from Table 10.1. The base length of each reply is denoted by, L_{MRxx}, where, xx, is the abbreviation of the inquiry from Table 10.1. The length of any variable portions of the reply are denoted by, L_{ERxx}, where, xx, is the abbreviation of the inquiry from Table 10.1. For replies that have no variable length components, the value of, L_{ERxx}, is 0 for those replies. When the reader is receiving a reply from a tag, the probability that the reader remains in state R is given by the following equation.

$$P_{RR} = \left(\frac{\sum_{\forall xx} \left(I_{xx} * \left(L_{MRxx} + L_{ERxx} - 1\right)\right)}{\sum_{\forall xx} \left(I_{xx} * \left(L_{MRxx} + L_{ERxx}\right)\right)} \right) = 1 - \left(\frac{\sum_{\forall xx} \left(I_{xx}\right)}{\sum_{\forall xx} \left(I_{xx} * \left(L_{MRxx} + L_{ERxx}\right)\right)} \right) \tag{10-33}$$

The probability of entering the processing reply state, state P, from state R is simply.

$$P_{RP} = 1 - P_{RR} \tag{10-34}$$

The transition probabilities originating in the send inquiry state, state S, can be found using a similar strategy as those for the receive reply state. The probability that there are additional bytes of the inquiry to send, or of remaining in state S is,

$$P_{SS} = \left(\frac{\sum_{\forall xx} \left(I_{xx} * \left(L_{MRxx} + L_{ERxx} - 1 \right) \right)}{\sum_{\forall xx} \left(I_{xx} * \left(L_{MRxx} + L_{ERxx} \right) \right)} \right) = 1 - \left(\frac{\sum_{\forall xx} \left(I_{xx} \right)}{\sum_{\forall xx} \left(I_{xx} * \left(L_{MRxx} + L_{ERxx} \right) \right)} \right) \tag{10-35}$$

The probability that the byte just sent is the last byte of the inquiry causing a transition into the listen to preamble state, state L, is given by (10-36)

$$P_{SL} = 1 - P_{SS} \tag{10-36}$$

The transition probabilities for transitions originating in the Process Reply state, state P, are now determined. While in state P, the reader can receive or send messages to other readers or the outside world, decide to transmit an inquiry transitioning to state S, or can return to state L if the message contains an error or is from a reader. The probability that the reader remains in the processing state is equal to the probability that the reader is processing a reply from a tag that is addressed to itself, or a message from another reader, or a message from the outside world. The reader must periodically send information to the outside world. The number of times in one day that the reader sends information to the outside world is denoted by, S_{OUT}. The number of messages that a given reader receives from other readers per day is denoted by M_{Reader}. The probability that the reader starts to send a message to or receives a message from the outside world, P_{PW}, is given by (10-37).

$$P_{PW} = \frac{M_{OUT} + S_{OUT}}{\sum_{\forall xx} \left(I_{xx} \right) + M_{OUT} + S_{OUT} + M_{READER} + M_T} \tag{10-37}$$

The probability that the reader begins to send an inquiry by transitioning into state S is given by (10-38).

$$P_{PS} = \frac{\sum_{\forall xx}(I_{xx})}{\sum_{\forall xx}(I_{xx}) + M_{OUT} + S_{OUT} + M_{READER} + M_T} \tag{10-38}$$

The probability that the message is from another reader or contains an error, causing a transition into state L is given by (10-39).

$$P_{PL} = \frac{(M_{Error})*\left(M_T - N_T \sum_{\forall xx}(I_{xx})\right)}{\sum_{\forall xx}(I_{xx}) + M_{OUT} + S_{OUT} + M_{READER} + M_T} \tag{10-39}$$

The probability that the reader starts to send a message to or receives a message from another reader is given by (10-40).

$$P_{PC} = \frac{(1 - M_{Error})*\left(M_T - N_T \sum_{\forall xx}(I_{xx})\right)}{\sum_{\forall xx}(I_{xx}) + M_{OUT} + S_{OUT} + M_{READER} + M_T} \tag{10-40}$$

The probability of remaining in state P, P_{PP}, is given by (10-41).

$$P_{PP} = 1 - (P_{PS} + P_{PW} + P_{PC} + P_{PL}) \tag{10-41}$$

The transition probabilities for those transitions originating in state L are now determined. The period of one day can be divided into τ_R time slots. Each time slot represents the time required to detect the preamble of a tag's reply. The number of time slots, τ_R, is found with the following equation.

$$\tau_R = \frac{S_{\text{Day}}}{S_{\text{Preamble}}} \tag{10-42}$$

Where S_{Day} is the number of seconds in one day and $S_{Preamble}$ is the number of seconds required to detect the reply's preamble. In ISO 18000-7 the preamble for a tag reply is 1296 μs (microseconds) [59].

The transition probabilities are then simply the ratios of the various actions that can occur in state L to τ_R. The probability of detecting the preamble of the reply and transitioning into state R is given by the following equation.

$$P_{LR} = \frac{M_T}{\tau_R} \tag{10-43}$$

The probability that a message is received from the outside world, causing a transition into state W is found using (10-44).

$$P_{LW} = \frac{M_{Out}}{\tau_R} \tag{10-44}$$

The probability that a message is received from another reader causing a transition into state C is found using the following equation.

$$P_{LC} = \frac{M_{\text{Reader}}}{\tau_R} \tag{10-45}$$

The probability that the reader remains in state L can be determined using (10-46), and it is assumed that $(M_T + M_{OUT} + M_{Reader}) << \tau_R$.

$$P_{LL} = 1 - (P_{LR} + P_{LC} + P_{LW}) \tag{10-46}$$

The rewards for each transaction represent the energy consumed in performing the action that particular transaction represents. The reward matrix for an ISO 18000-7 reader has the following format.

$$R = \begin{bmatrix} R_{LL} & 0 & R_{LW} & R_{LC} & R_{LR} & 0 \\ R_{SL} & R_{SS} & 0 & 0 & 0 & 0 \\ 0 & 0 & R_{WW} & 0 & 0 & R_{WP} \\ 0 & 0 & 0 & R_{CC} & 0 & R_{CP} \\ 0 & 0 & 0 & 0 & R_{RR} & R_{RP} \\ R_{PL} & R_{PS} & R_{PW} & R_{PC} & 0 & R_{PP} \end{bmatrix} \quad (10\text{-}47)$$

The reader developed by Cho and Baek uses an Atmega128L processor, a XE1203F transceiver and an RS-232 interface connecting the reader to the outside world operating at 9.6 kbps [60]. The processor is assumed to operate at 8 MHz operating at 5 V draws 11 mA when active and 15 μA (micro-Amps) when asleep (watchdog timer enabled) [61]. The processor consumes 55 mW when active and 75 μW (micro-Watts) when asleep. The transmitter portion of the XE1203F transceiver draws 40 mA when active, the receiver portion draws 17 mA when active, and the transceiver draws 1 μA(micro-Amps) in when asleep [60, 62]. The transceiver operates at 3.3 V, and the transceiver consumes 132 mW when transmitting, 56.1 mW when receiving, and 3.3 μA (micro-Amps) when asleep [60, 62]. The power consumption for the reader under different conditions are listed in Table 10.8.

Table 10.8: Power consumption for the reader under different conditions.

Parameter	Power Consumption (mW)
PW_L	111.1 mW
PW_S	187 mW
PW_{SL}	111.1 mW
PW_W	111.1 mW
PW_C	111.1 mW
PW_R	111.1 mW
PW_P	111.1 mW
PW_{PS}	187 mW

The reward values for those transitions that originate in state L are found first. The energy consumed while remaining in state L, listening for the preamble is given by the following equation.

$$R_{LL} = S_{\text{Preamble}} * PW_L \tag{10-48}$$

The term, PW_L, in (10-48) represents the power consumption of the reader in state L. Specifically, PW_L, includes the power consumption of the active receiver (RFID tag), processor or controller, transceiver for inter-reader communications, and the transceiver for communication with the outside world. The power consumption of those devices in sleep mode is also included in, PW_L, and in any future power variables unless specifically mentioned otherwise. The remaining components are asleep to conserve power.

When a message is detected from the outside world, the reader transitions into state W in order to receive the message. Similarly, when the reader detects an incoming message from another reader it will transition into state C to receive that message. The power consumption during the transition caused by detection of an ISO 18000-7 preamble or a message from the outside world or another reader is equal to, PW_L, because no additional components must be activated to receive either message. Likewise, no additional hardware must be activated to receive an incoming any of these three types of messages. The incoming ISO 18000-7 message could be from a tag or from another reader, but that is dealt with in state P after the message has been received. Hence, the energy consumption for the transition from state L into state R, W, or C is equal to the energy consumption of remaining in state L.

$$R_{LL} = R_{LR} = R_{LW} = R_{LC} \tag{10-49}$$

The energy consumption for the actions that may occur while the reader is in the Send Inquiry state, state S is now found. While in state S, the reader will transmit bytes of the inquiry command one by one until the complete command has been transmitted. The power consumption of the tag while in state S is denoted by, PW_S. The term, PW_S, represents the power consumption of the active transmitter (RFID tag), active transmitter, the two transceivers connecting the reader to the outside world and to other readers, and the processor or controller

unit. In state S, all components are active and the time required to transmit one byte of the message is denoted as, $T_{Byte\text{-}S}$. The time to transmit or receive one byte of an ISO 18000-7 message, $T_{Byte\text{-}S}$, is given by (10-26). The energy consumed to send one byte of the message other than the last byte is given by (10-50).

$$R_{SS} = T_{Byte-S} * PW_S \tag{10-50}$$

When the reader sends the last byte of the inquiry message, it transitions from state S into state L to listen for the reply. During this transition, the transmitter is powered down and the active receiver (RFID tag) is wakened. The power consumption, for powering down the active transmitter (RFID tag), is denoted by, PW_{SL}, and the time required to power down the active transmitter is denoted by, T_{SL}. It is assumed that, T_{SL}, is less than or equal to, $T_{Byte\text{-}S}$. The energy consumed transitioning from state S into state L is given by the following equation.

$$R_{SL} = T_{Byte-S} * PW_S + T_{SL} * PW_{SL} \tag{10-51}$$

While the reader is in state W, it can communicate with the outside world staying in state W or transition into state P to process the message after receiving the entire message. In state W the transceiver for communication with the outside world, the transceiver for reader-to-reader communication, active receiver, and the processor or controller components are active. The other components are asleep in order to conserve energy. The power consumption of active components of the reader while in state W is denoted by, PW_W, and is equal to, PW_L. The reader requires time, $T_{Byte\text{-}W}$, to transmit or receive one byte to or from the outside world. The link to the outside world is an RS-232 link operating at 9.6 kbps [60]. In this example it is assumed that the RS-232 link transmits a byte of information using 8 data bits, 1 start bit, 1 stop bit, and no parity bit. Hence, each byte of information transmitted or received over the RS-232 link requires 10-bits. Thus, $T_{Byte\text{-}W}$, is given by the following equation.

$$T_{Byte-W} = 10 * \left(\frac{1}{9600}\right) = 1.0416 \text{ ms} \tag{10-52}$$

Hence, the energy consumption of remaining in state W is given by the following equation.

$$R_{WW} = T_{\text{Byte-W}} * PW_W = T_{\text{Byte-W}} * PW_L \tag{10-53}$$

When the last byte of the message to the outside world has been successfully sent or received the reader will transition into state P, but no additional components must be activated. Thus, the energy consumption for the W to P transition is equal to that for the transition from state W back into itself.

$$R_{WP} = R_{WW} = T_{\text{Byte-W}} * PW_L \tag{10-54}$$

The energy consumption of the reader as a result of possible actions while the reader is communicating with another reader in state C is similar to that when the reader is in state W. When communicating with another reader, the processor or controller, active receiver, and the reader-to-reader and reader-to-outside world transceivers must be active. The other components are placed in sleep mode to conserve energy. The power consumption of the processor or controller and the reader-to-reader transceiver is denoted by, PW_C, and is equal to both, PW_W, and PW_L. The reader requires, $T_{\text{Byte-C}}$, to transmit or receive one byte to or from the other reader. Because the same communication link is used to connect the readers to each other and the outside world, $T_{\text{Byte-C}}$, equals, $T_{\text{Byte-W}}$. Hence, while transmitting of receiving a message to or from another reader, the energy consumed is simply,

$$R_{CC} = T_{\text{Byte-C}} * PW_C = T_{\text{Byte-C}} * PW_L \tag{10-55}$$

When the last byte of the message has been sent or received, the reader will transition into state P. During this transition, no additional hardware components must be activated. The energy consumed when transitioning from state C to state P is equal to that consumed during the transition from state C to state C.

$$R_{CP} = R_{CC} = T_{\text{Byte–C}} * PW_L \tag{10-56}$$

The energy consumption for the receive reply state, state R, is similar to that in states L, W, and C. The reader requires, $T_{\text{Byte-R}}$, time to receive one byte of the ISO 18000-7 message, and $T_{\text{Byte-R}}$, is equal to, $T_{\text{Byte-S}}$. During reception of an ISO message the processor or controller, the active receiver, and transceivers connecting the reader to the outside world and the other readers are required. The remaining components are kept in sleep mode to reduce energy consumption. The power consumption of the reader with these components powered-up is denoted by, PW_R, and is equal to, PW_L. The energy consumption to receive one byte of the message while remaining in state R (there are more bytes of message left to receive) is given by the following equation.

$$R_{RR} = T_{\text{Byte–R}} * PW_R = T_{\text{Byte–R}} * PW_L \tag{10-57}$$

The energy consumption of the reader when transitioning into state P from state R is equal to the energy consumption of the reader while remaining in state R.

$$R_{RP} = R_{RR} = T_{\text{Byte–R}} * PW_L \tag{10-58}$$

The power consumption of the reader in state P is determined from the power consumption of the processor or controller, active receiver, and the two transceivers connecting the reader to the outside world and to other readers. The active receiver and active transmitter are both put into sleep mode to reduce power consumption. The power consumption of the reader when in state P is denoted by, PW_P, which is equal to, PW_L. To process one byte of a message requires, $T_{\text{Byte-P}}$, time and can be found using the following equation.

$$T_{\text{Byte–P}} = \left(\frac{N_I}{C_P} \right) \tag{10-59}$$

Where, N_I, is the number of instructions required to process one byte, and C_P, is the number of instructions per second that the processor executes. In this case, N_I is assumed to be 10,000 and C_P is assumed to be 8 MHz. Thus, using (10-27), $T_{Byte\text{-}P}$, is found to be 1.25 ms. The energy consumption for remaining in state P, R_{PP}, is given by (10-60).

$$R_{PP} = T_{\text{Byte}-\text{P}} * PW_P = T_{\text{Byte}-\text{P}} * PW_L \tag{10-60}$$

When a message is detected from or sent to the outside world, the reader transitions into state W to communicate with the outside world. In state W the active receiver is activated in state W, but it is not activated during the transition into state W from state P. Hence, the energy consumption for the transition from state P to state W is given below.

$$R_{PW} = R_{PP} = T_{\text{Byte}-\text{P}} * PW_P = T_{\text{Byte}-\text{P}} * PW_L \tag{10-61}$$

Similarly, the energy consumed during the transition from state P to state C and from state P to state L is identical to that for the P to W transition.

$$R_{PC} = R_{PL} = R_{PW} = R_{PP} = T_{\text{Byte}-\text{P}} * PW_L \tag{10-62}$$

When an inquiry must be sent to the tags, the active transmitter must be wakened. The reader must first transmit the wake-up tone for 2.5 seconds and then transmit the first byte of information. Hence, T_{Wake}, is 2.5 seconds. The power consumption of active transmitter during the wake-up process is denoted by, PW_{PS}. The energy consumption for the P to S transition is given in (10-63).

$$R_{PS} = T_{\text{Byte}-\text{P}} * PW_P + T_{Wake} * PW_{PS} \tag{10-63}$$

Expressions to compute all the reward values have been defined and the reward matrix can be calculated for any given ISO 18000-7 reader that fits the model's assumptions. The average length of a tag's reply that is overheard by a reader other than the receipient of reply is

denoted by, L_{Avg-R}. The parameters specifically for the readers for this example are listed in Table 10.9.

Table 10.9: Parameters for the single reader model.

Parameter	Reader – R1	Reader – R2	Reader – R3	Reader – R4
M_{OUT} (Messages)	10	24	48	20
M_{READER} (Messages)	1749	1773	1625	380
L_{READER} (Bytes)	8	8	8	8
M_T (Messages)	1401	1302	1523	792
M_{ERROR} (Percentage)	6 %	7 %	3 %	0 %
S_{OUT} (Messages)	15	54	32	37
L_{OUT} (Bytes)	20	20	20	20
L_{Avg-R}	18.2778	18.2778	18.2778	18.2778
$S_{Preamble}$ (μs)	1296	1296	1296	1296
T_{Byte-P} (ms)	1.25	1.25	1.25	1.25
T_{Byte-R} (ms)	0.3249	0.3249	0.3249	0.3249
T_{Byte-S} (ms)	0.3249	0.3249	0.3249	0.3249
T_{SL} (ms)	0.100	0.100	0.100	0.100
T_{Byte-C} (ms)	1.0416	1.0416	1.0416	1.0416
T_{Byte-W} (ms)	1.0416	1.0416	1.0416	1.0416

The number of inquiries that each reader transmits per day is listed in Table 10.3 and the lengths of the set and variable parts of each command and reply are listed in Table 10.6. The number of messages of each type of inquiry, M_{Ixx}, that the reader receives from the outside world (instructing reader to transmit that inquiry) is assumed to be half of the total inquiries issued by the reader which are listed in Table 10.3. The energy consumption of each of the four readers for one day is listed in Table 10.10.

Table 10.10: Energy consumption of the four readers operating for one day.

Reader	Energy Consumption (mJ)
R1	9910931.11 mJ
R2	9911577.09 mJ
R3	10137390.92 mJ
R4	9937375.96 mJ

Summing the energy consumption of all 20 tags and 4 readers that make up this example network, the energy consumed by the entire network for one day is calculated. The energy consumed by this example network for a period of one day is 161644185.89 mJ.

10.2.6 Step 4: Identification of Interactions between the Base Level Entities

The interactions between the base level entities are found in this section. This is step 4 in the algorithm. The tags respond to reader inquiries and the reader initiates communication. Because of this an ISO 18000-7 network falls under the category of Responds to Commands network. The interactions are limited to reader-tag exchanges with the reader initiating the exchange and the tag simply providing the requested information in reply. These interactions will be used along with the tasks to identify the single reader and associated tags topological entities contained in the example ISO 18000-7 network.

10.2.7 Step 5: Identification of the Single Reader and Associated Tags Topological Entities

The single reader and associated tags topological entities are identified in this section. This is step 5 in the algorithm. Based on the interactions between the tags and readers, the simplest topology group for an ISO 18000-7 network contains a single reader and all ISO 18000-7 tags within range of that reader. Tags that are within range of more than one reader will be present in the topology built around each reader the tag is within range of. Four readers are in the example

network. Therefore, four single reader and associated tags topologies are needed to represent the entire network. The entities contained in the topology built around reader R1 are shown in Figure 10.7. This topology is referred to as TOP1.

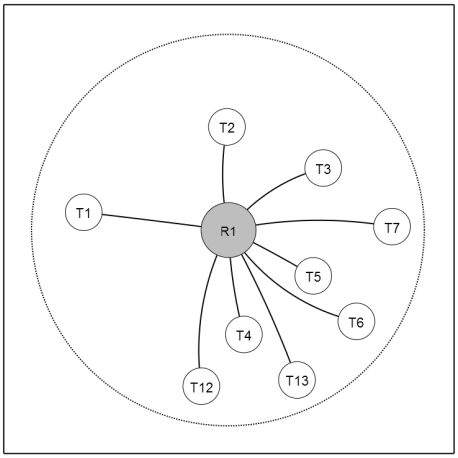

Figure 10.7: Entities contained in topology, TOP1, built around reader R1.

The topology built around reader R2 is shown in Figure 10.8 and is referred to as TOP2.

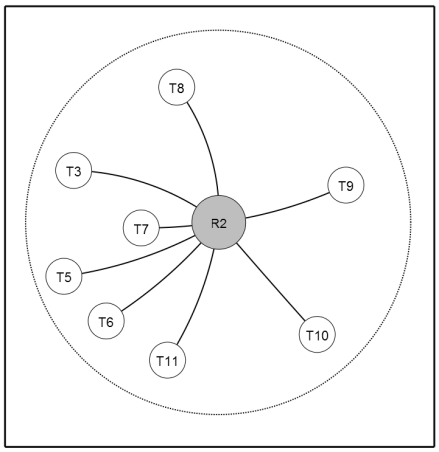

Figure 10.8: Entities contained in topology, TOP2, built around reader R2.

The entities contained in the topology centered around reader, R3 are shown in Figure 10.9 and are referred to as TOP3.

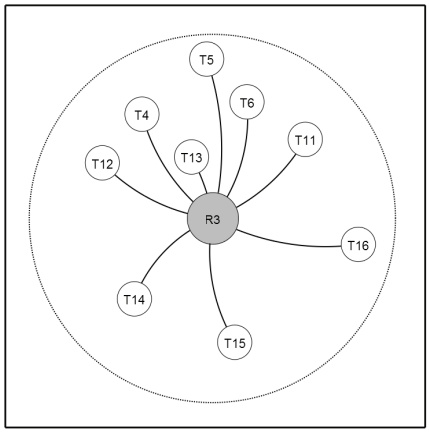

Figure 10.9: Entities contained in topology, TOP3, built around reader R3.

The final topology, constructed around reader, R4, is shown in Figure 10.10 and is referred to as TOP4.

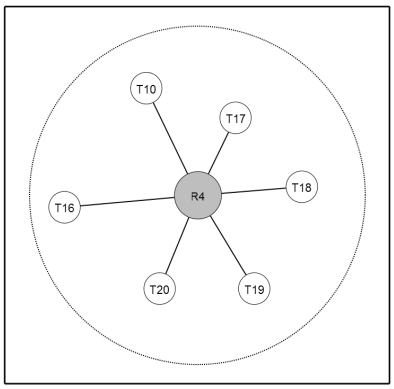

Figure 10.10: Entities contained in topology, TOP4, built around reader R4.

Applying this topology to an ISO 18000-7 network reduces the total number of simulation entities from twenty-four, the total number of entities in the network ($N_{Tag} + N_{Reader}$) to, four, the number of readers (N_{Reader}) in the network. This reduces the number of simulation entities that must be evaluated. Hence decreasing the time required to determine the energy consumption of the network. Once a single reader and associated tags topological entity is identified it is replaced with the topological entity reducing the order of the basic communication graph. Unless otherwise noted reducing the order means a reduction in the number of Markov processes that must be evaluated not in reduction of the dimensionality of the Markov process. With the single reader and associated tags fundamental topological entity identified existing algorithms can be used to search the basic communication graph to identify these topological entities. These algorithms form the basis for CAD tools for integrated circuit design [46]. The reduced network with using the four single reader and associated tags topological entities (TOP1, TOP2, TOP3, and TOP4) to cover the network shown in Figure 10.2 is shown in Figure 10.11.

Tags that are in more than one topology (connected to more than one reader in the basic communication graph shown in Figure 10.3) must be taken into account when the energy consumption of the topology is determined. A tag will respond to any reader that is within range. Therefore, it is possible for the reader in one topology to overhear a reply from a tag within that same topology that is addressed to another reader. In this case, the reader forwards the overheard reply to the reader to which the reply is addressed. Tags are placed in the topology built around each reader that they can communicate with.

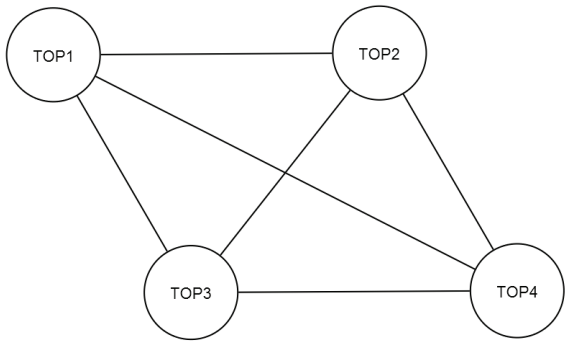

Figure 10.11: Example ISO 18000-7 network from Figure 10.2 covered using four single reader and associated tags topological entities.

10.2.8 Step 6: Identification of Tasks for the of Single Reader and Associated Tags Topological Entity

The tasks of the single reader and associated tags topological entities are found in this section. This is step 6 in the algorithm. This topological entity must perform three basic tasks. The first task is for the reader interrogate the tags for control purposes and to gather information held by the tags. The reader issues two types of inquiries, broadcast and point-to-point. The second task is for the reader to communicate with other readers when a reply addressed to another reader is received. The third task is for the reader to communicate with the outside world for control purposes and to send collected information to the outside world.

10.2.9 Step 7: Markov Process for the Single Reader and Associated Tags Topological Entity

The Markov process for the single reader and associated tags topological entity and expressions for the probability and rewards matrices are found in this section and are based on the tasks identified during step 6 of the algorithm in Section 10.2.8. This is step 7 in the algorithm. The topology may contain tags that are within range of more than one reader. This causes those tags to transmit a reply to inquiries from a reader other than the reader that the topology is centered around. The reader at the center of the topology will hear replies sent by tags that are within range of more than one reader. To account for this the Overhear Tag Reply state, state O, is included in the Markov process. When not performing one of these tasks the entire topology is idle attempting to conserve power. The Markov process for this topology is shown in Figure 10.12 below.

The Markov process for this topology contains only five states, the same number of states as the Markov process for a single tag, and one less state than the Markov process for a single reader. The number of computations required to evaluate the Markov process for this topology is, at worst, no more than that required to evaluate the Markov process for a single tag or a single reader. The probability matrix for this topology has the following form.

$$P = \begin{bmatrix} P_{SS} & P_{SI} & P_{SR} & P_{SW} & P_{SO} \\ P_{IS} & P_{II} & 0 & 0 & 0 \\ P_{RS} & 0 & P_{RR} & 0 & 0 \\ P_{WS} & 0 & 0 & P_{WW} & 0 \\ 0 & 0 & P_{OR} & 0 & P_{OO} \end{bmatrix} \tag{10-64}$$

The reward matrix for the single reader and associated tag topology has the following form.

$$R = \begin{bmatrix} R_{SS} & R_{SI} & R_{SR} & R_{SW} & R_{SO} \\ R_{IS} & R_{II} & 0 & 0 & 0 \\ R_{RS} & 0 & R_{RR} & 0 & 0 \\ R_{WS} & 0 & 0 & R_{WW} & 0 \\ 0 & 0 & R_{OR} & 0 & R_{OO} \end{bmatrix} \tag{10-65}$$

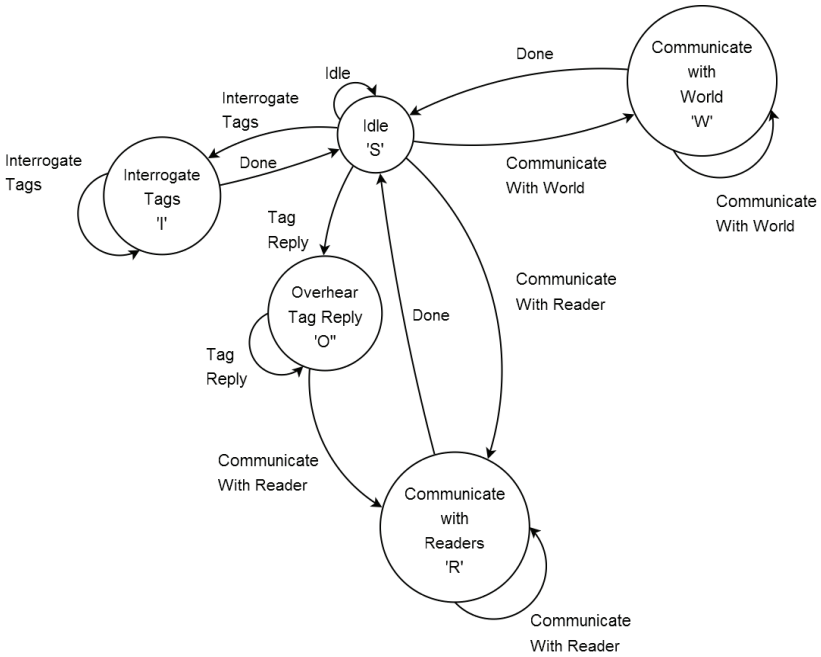

Figure 10.12: Markov process for a topological entity containing one reader and all tags associated with that reader.

Each single reader and associated tags topology contains one reader, and N_T, tags. The reader issues, I, inquiries per day, and the number of each type of inquiry issued is listed in Table 10.3. The reader communicates with the outside world receiving, M_{OUT}, control messages from the outside world and sends, S_{OUT}, information messages to the outside world. The reader overhears, M_{OVER}, tag replies generated by other readers within range of one of the tags within this topology. The number of replies that the reader overhears is determined by the number of times that are within range of more than one reader respond to another reader's inquiry. This can be calculated using the following equation,

$$M_{OVER} = \sum_{x \in CT} \left(\sum_{y \in PtP} \left(I_{x_{y,a}} * M_{P_x} \right) + \sum_{y \in B} \left(I_{x_{y,a}} \right) \right) \tag{10-66}$$

Where, CT, denotes the set of all tags within a single reader and associated tags topology that are within range of at least one other reader: M_P, is the probability that a point-to-point command is addressed to a particular tag, x; PtP,denotes the set of all inquiries that are point-to-point, and B denotes the set of all inquiries that are broadcast. The parameter, y, represents the set of readers that are within range of tag x. The parameter, a, denotes which command the overheard reply is for, the commands are listed in Table 10.1. The reader receives, M_{Reader}, messages per day from other readers. The number of each type of command that the reader issues is denoted by, M_{Ixx}, where, xx, is the abbreviation for the one of the commands in Table 10.1. The value of, M_{Reader}, is obtained using (10-67).

$$M_{\text{Reader}} = \sum_{x \in CT} \left(\deg(x) - 1 \right) \left(\sum_{\forall xx} \left(M_{Ixx} \right) \right) \tag{10-67}$$

The basic communication graph is used to determine how many additional readers will overhear tag x's response to the reader's inquiry. The term, deg(x), in (10-67) denotes the degree of the vertex representing tag x in the basic communication graph, and one edge is subtracted as that edge represents the link between the reader that issued the inquiry and tag x in a given topology. Each reader that can communicate with tag x must be taken into account. A tag belongs to the set of tags that can communicate with multiple readers, CT, if it has a degree

greater than one. The basic communication graph for this network is shown in Figure 10.13 and is used to determine the degree for tag in the network.

The transition probabilities for those transitions originating in state S are determined. The period of one day is divided into, T, periods, with each period being, τ, seconds long. As before for the single reader case, the value of, τ, will be 1.25 milliseconds, and T, is given by (10-3). The system will transition into state O when a tag sends a reply to another reader's inquiry and the probability of this occurring, P_{SO}, is given by (10-68).

$$P_{SO} = \frac{M_{\text{OVER}}}{T} \tag{10-68}$$

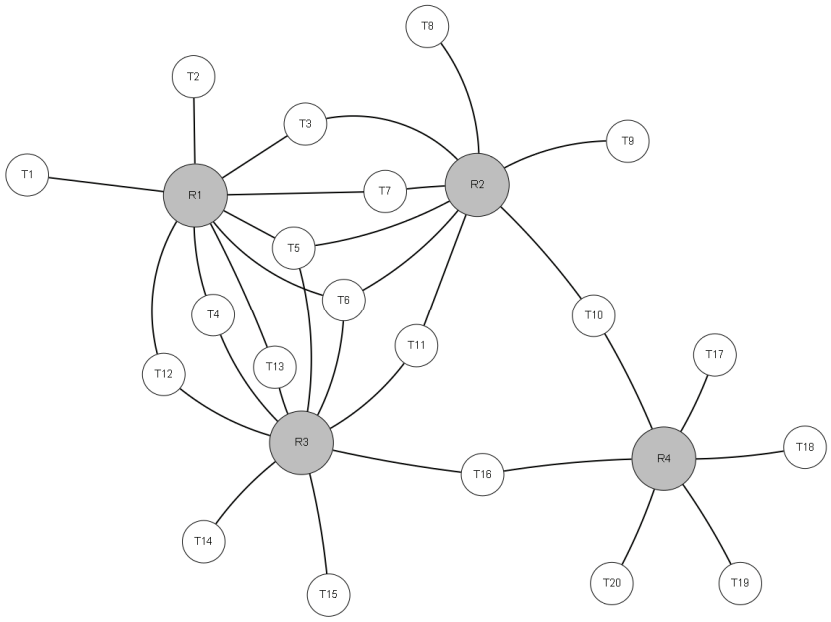

Figure 10.13: Basic communication graph of the example ISO 18000-7 network (readers are shaded and tags are not shaded).

Similarly, the probability that the system transitions into state I is the probability that the reader issues an inquiry. The probability, P_{SI}, of the reader issuing an inquiry is given below.

$$P_{SI} = \frac{I}{T} = \frac{\sum_{\forall xx} M_{Ixx}}{T} \tag{10-69}$$

The probability that the reader receives or sends a message to the outside world transitioning into state W, P_{SW}, is given by the following equation,

$$P_{SW} = \frac{M_{\text{OUT}} + S_{\text{OUT}}}{T} \tag{10-70}$$

The probability that the reader receives a message from another reader is computed using the following equation, where, M_{Reader}, denotes the number of messages received from the other readers in one day.

$$P_{SR} = \frac{M_{\text{Reader}}}{T} \tag{10-71}$$

The probability that the topology remains dormant, or in state S, is simply the probability that there is no event for a given τ-period, and P_{SS}, is simply,

$$P_{SS} = 1 - \left(P_{SI} + P_{SW} + P_{SO} + P_{SR}\right) \tag{10-72}$$

The reader communicates with the outside world by sending or receiving messages. The reader sends, M_{OUT}, messages per day and each message contains, L_{Out}, bytes. The base lengths of each of the eighteen replies includes all non-variable portions of the message and are denoted as, L_{MRxx}, where, xx, is the abbreviation for the one of the commands in Table 10.1. For those replies having variable length components the length of those components, is denoted by, L_{ERxx}, again, xx, is the abbreviation for the one of the commands in Table 10.1. The number of each type of command received from the outside is denoted by, M_{IOxx}, where, xx, is the abbreviation for one of the commands listed in Table 10.1. In this example, half of the reader's total inquiries are received from the outside world.

$$M_{IOxx} = 0.5 * M_{Ixx} \tag{10-73}$$

The probability that the reader is still sending or receiving a message, P_{WW}, is given by (10-74).

$$P_{WW} = \left(\frac{\sum_{\forall xx} \left(M_{IO_{xx}} * \left(L_{Mxx} + L_{Exx} - 1 \right) \right) + M_{Out} * L_{Out}}{\sum_{\forall xx} \left(M_{IO_{xx}} * \left(L_{Mxx} + L_{Exx} \right) \right) + M_{Out} * L_{Out}} \right) = 1 - \left(\frac{\sum_{\forall xx} \left(M_{IO_{xx}} \right)}{\sum_{\forall xx} \left(M_{IO_{xx}} * \left(L_{Mxx} + L_{Exx} \right) \right) + M_{Out} * L_{Out}} \right) \tag{10-74}$$

The probability that the reader receives or sends the last byte of the message is simply,

$$P_{WS} = 1 - P_{WW} \tag{10-75}$$

The probability that the system is overhearing a reply from a tag addressed to another reader is given by the following equation.

$$P_{OO} = \left(\frac{\sum_{\forall xx} \left(M_{I_{xx}} * \left(L_{MRxx} + L_{ERxx} - 1 \right) \right)}{\sum_{\forall xx} \left(M_{I_{xx}} * \left(L_{MRxx} + L_{ERxx} \right) \right)} \right) = 1 - \left(\frac{\sum_{\forall xx} \left(M_{I_{xx}} \right)}{\sum_{\forall xx} \left(M_{I_{xx}} * \left(L_{MRxx} + L_{ERxx} \right) \right)} \right) \tag{10-76}$$

The probability of the reader receiving the last byte of the reply is simply,

$$P_{OR} = 1 - P_{OO} \tag{10-77}$$

After overhearing the reply to another reader's inquiry, the reader in the system will forward the reply to the appropriate reader. The total number of each type of inquiries, $M_{Ixx\text{-}b}$, received by a tag, b, in the set of tags, CT, is simply the sum of each type of command issued by the set of readers, CR_b, that is within range of tag b. The set of readers, CR_b, does not contain the reader that the topology tag b belongs to is centered around because replies to those commands sent by the reader within the topology are not forwarded.

$$M_{Ixx-b} = \sum_{\substack{\forall c \in CR_b \\ \forall xx}} M_{Ixx} \tag{10-78}$$

Each message forwarded to another reader has overhead of, L_{Reader}, bytes. The probability that the reader is still transmitting the reply and remains in state R, P_{RR}, is given by (10-79).

$$\begin{aligned} P_{RR} &= \left(\frac{\sum_{\substack{\forall xx \\ \forall b \in CT}} \left(M_{I\,xx\text{-}b} * \left(L_{MRxx} + L_{Reader} + L_{ERxx} - 1 \right) \right)}{\sum_{\substack{\forall xx \\ \forall b \in CT}} \left(M_{I\,xx\text{-}b} * \left(L_{MRxx} + L_{Reader} + L_{ERxx} \right) \right)} \right) = 1 - \left(\frac{\sum_{\forall xx} \left(M_{I\,xx\text{-}b} \right)}{\sum_{\substack{\forall xx \\ \forall b \in CT}} \left(M_{I\,xx\text{-}b} * \left(L_{MRxx} + L_{Reader} + L_{ERxx} \right) \right)} \right) \\ &= 1 - \left(\frac{M_{Reader}}{\sum_{\substack{\forall xx \\ \forall b \in CT}} \left(M_{I\,xx\text{-}b} * \left(L_{MRxx} + L_{Reader} + L_{ERxx} \right) \right)} \right) \end{aligned} \tag{10-79}$$

The probability that the reader transmits the last byte of the message and returns to state S, P_{RS}, is simply,

$$P_{RS} = 1 - P_{RR} \tag{10-80}$$

When the reader issues an inquiry to the tags within the topological entity centered on the reader, the reader must first send the inquiry, then the tag must process the inquiry and send the reply. The probability that the topology is performing the interrogation task, P_{II}, is given by the following equation.

$$\begin{aligned} P_{II} &= \left(\frac{N_T * \sum_{\forall xx} \left(\left(M_{I\,xx} * \left(L_{Mxx} + L_{Exx} + L_{MRxx} + L_{ERxx} - 1 \right) \right) \right)}{N_T * \sum_{\forall xx} \left(M_{I\,xx} * \left(L_{Mxx} + L_{Exx} + L_{MRxx} + L_{ERxx} \right) \right)} \right) \\ &= 1 - \left(\frac{N_T * \sum_{\forall xx} \left(M_{I\,xx} \right)}{N_T * \sum_{\forall xx} \left(M_{I\,xx} * \left(L_{Mxx} + L_{Exx} + L_{MRxx} + L_{ERxx} \right) \right)} \right) \end{aligned} \tag{10-81}$$

where, N_T, is the number of tags in the topology. The transition probability that the topology is performing the final operations for the interrogation task (the reader finishes processing the reply) and returns to state S, P_{IS}, is simply,

$$P_{IS} = 1 - P_{II} \tag{10-82}$$

The reward matrix has the following form.

$$R = \begin{bmatrix} R_{SS} & R_{SI} & R_{SR} & R_{SW} & R_{SO} \\ R_{IS} & R_{II} & 0 & 0 & 0 \\ R_{RS} & 0 & R_{RR} & 0 & 0 \\ R_{WS} & 0 & 0 & R_{WW} & 0 \\ 0 & 0 & R_{OR} & 0 & R_{OO} \end{bmatrix} \tag{10-83}$$

The time, τ, is defined to be 1.25 ms (milliseconds). The energy consumption of the topology when remaining in the idle state, R_{SS}, is,

$$R_{SS} = \tau * PW_S \tag{10-84}$$

where, PW_S, denotes the power consumption of the topology in the idle state. In the idle state, the reader power consumption, $PW_{S\text{-}Rdr}$, consists of the power consumption of the processor, transceivers for communicating with other readers and the outside world, the active receiver that receives tag replies, and the power consumed by the active receiver while in the dormant state. Unless otherwise noted all power consumption parameters include the power consumption of the dormant state of those components that are not active. When the topology is in the idle state, each tag within the topology is in the dormant state. In the dormant state, the power consumption of the tag, $PW_{S\text{-}Tag}$, includes the power consumption of the active receiver and the processor that is monitoring for the wake-up pulse. There are, N_T, tags in each topology, the power consumption for the topology in the idle state, PW_S, is given by the following equation.

$$PW_S = \left(PW_{S-Rdr} + N_T * PW_{S-Tag}\right) \tag{10-85}$$

When the reader communicates with other readers, the outside world, or overhears a response from another tag the necessary components are already active. Thus, the energy consumption for the transitions from state S to either state R, W, or O are equal. This transition is assumed to take one τ-period. The energy consumption, R_{SR}, R_{SW}, R_{SO}, for the S-R, S-W, and S-O transitions respectively, are equal and are given by the following equation.

$$R_{SR} = R_{SW} = R_{SO} = \tau * \left(PW_{S-Rdr} + N_T * PW_{S-Tag}\right) = R_{SS} \tag{10-86}$$

When the reader sends an inquiry the topology transitions into state I to interrogate the tags. During the transition from state S to state I the reader activates the transmitter and issues the wake-up pulse for 2.5 seconds, denoted as, T_{WAKE}, to wake-up all tags within range [59]. The power consumption of the reader with the processor, transceivers connecting the reader to the outside world and other readers, active receiver and active transmitter is denoted by, $PW_{SI\text{-}Rdr}$. The tag hears the wake-up tone and begins to listen for the preamble, but no additional components must be activated for this so the power consumption, $PW_{SI\text{-}Tag}$, is equal to, $PW_{S\text{-}Tag}$. All, N_T, tags wake-up and consuming, $PW_{SI\text{-}Tag}$, Watts and begin to listen for the preamble. The time that is required to transmit the preamble, 1,296 μs, is denoted by, $T_{Preamble}$. The energy consumed turning the transition from state S to state I is,

$$R_{SI} = \left(T_{WAKE} + T_{\text{Preamble}}\right) * \left(PW_{SI-Rdr} + N_T * PW_{SI-Tag}\right) \tag{10-87}$$

When the reader issues an inquiry, the reader must first transmit the inquiry and the tags simultaneously receive the reply. After receiving the inquiry, the tags process the inquiry to determine if the command is addressed to them and if so to generate the appropriate response. The tags then send the response and go back to sleep. The reader will receive and then process the tag's reply, finally returning to the idle state. The reader only activates the active transmitter while it is sending the wake-up pulse or the inquiry, the rest of the time the active transmitter is dormant. Each transition from state I back into state I represents an operation on one byte of the

message. The message can be received, transmitted, or processed, and the time to perform each of these three operations on one byte of data is denoted by T_B, T_B, and T_P, respectively. The times to receive and transmit one byte of the message are identical because the data rate is the same for both operations in the ISO 18000-7 standard [59]. The reader and tag could have different processors (most likely the processor on the reader will be more powerful), and the time required to process one byte of the message on the reader will be denoted as, $T_{P\text{-}Rdr}$, and the time to process one byte of the message on the tag will be denoted as, $T_{P\text{-}Tag}$. The time, T_B, can be found using (10-26). The times for, $T_{P\text{-}Rdr}$, and $T_{P\text{-}Tag}$, can be determined by estimating the number of instructions required to process one byte of the message and the clock speed of the processor. The number instructions required for the reader to process one byte of the message is denoted by, $N_{I\text{-}Rdr}$, while the tag requires, $N_{I\text{-}Tag}$, instructions to process one byte of the message. The clock speed of the reader's processor is, C_{Rdr}, and the clock speed of the tag's processor is, C_{Tag}. The values of, $T_{P\text{-}Rdr}$, and $T_{P\text{-}Tag}$, can be found using the two equations below.

$$T_{P-Rdr} = \frac{N_{I-Rdr}}{C_{Rdr}} \tag{10-88}$$

$$T_{P-Tag} = \frac{N_{I-Tag}}{C_{Tag}} \tag{10-89}$$

The probability that the inquiry that the reader issues is a broadcast command, M_B, is calculated using the following equation,

$$M_B = \frac{\sum_{\forall xx \in B} I_{xx}}{\sum_{\forall xx} I_{xx}} \tag{10-90}$$

where B denotes all inquiries that fall into the broadcast category.

The power consumption of the reader while transmitting the inquiry, $PW_{II\text{-}Rdr}$, is equal to $PW_{SI\text{-}Rdr}$ because no additional components are required to be active. The power consumption of the tag while receiving an inquiry, $PW_{II\text{-}Tag}$, is equal to, $PW_{SI\text{-}Tag}$, because all the necessary components are active. The tag must activate the active transmitter to send the response the to

the command, and the power consumption of the tag during the transmission of the reply is denoted by, $PW_{TX\text{-}Tag}$, during which time the processor, active receiver, and active transmitter are active. The energy consumed for the one interrogation within the topology is given by (10-91).

$$\begin{aligned} R_{II} = {} & T_B * \left(N_T * PW_{II-Tag} + PW_{II-Rdr}\right) + T_{P-Tag} * N_T * PW_{II-Tag} + T_{P-Rdr} * PW_{S-Rdr} \\ & + T_B * \left(N_T * PW_{TX-Tag} + PW_{S-Rdr}\right) + \left(T_{\text{Preamble}} * \left(N_T * PW_{TX-Tag} + PW_{S-Rdr}\right)\right) \end{aligned} \tag{10-91}$$

The energy consumption for the transition from state I to state S, R_{IS}, represents operating on the final byte and is therefore identical to, R_{II}.

$$R_{IS} = R_{II} \tag{10-92}$$

The reader does not need to activate any additional components to communicate with another reader or the outside world, or to overhear the reply addressed to another reader. Therefore, the power consumption of the reader in states R, W, and O is equal to, $PW_{S\text{-}Rdr}$. The number of symbols that can be sent across the reader-to-reader network in one second is denoted by, D_{RtR}, and the number of symbols in one byte is denoted by, $N_{Byte\text{-}RtR}$. Similarly, D_{Out} denotes the number of symbols that can be sent across the network connecting the reader to the outside world in one second, and the number of symbols in one byte is denoted by, $N_{Byte\text{-}Out}$. Hence, the time to transmit one byte across the reader-to-reader network, $T_{Byte\text{-}RtR}$, and the time to transmit one byte across the network connecting the reader to the outside world, $T_{Byte\text{-}Out}$, are found using the following two equations.

$$T_{Byte-RtR} = \frac{D_{RtR}}{N_{\text{Byte-RtR}}} \tag{10-93}$$

$$T_{Byte-Out} = \frac{D_{Out}}{N_{\text{Byte-Out}}} \tag{10-94}$$

The energy consumed while receiving a byte of a message from another reader, R_{RR}, is given by the equation below.

$$R_{RR} = T_{Byte-RtR} * \left(PW_{S-Rdr} + PW_{S-Tag} \right) \quad (10\text{-}95)$$

During the transition from state R to state S, the reader communicates the last byte of the message to or from the other reader. Thus the energy consumption for the transition from state R to state S, R_{RS}, is equal to, R_{RR}.

$$R_{RS} = R_{RR} \quad (10\text{-}96)$$

The energy consumed while the reader is communicating with the outside world, R_{WW}, is given in the following equation.

$$R_{WW} = T_{Byte-Out} * \left(PW_{S-Rdr} + PW_{S-Tag} \right) \quad (10\text{-}97)$$

The reader communicates the last byte of information with the outside world during the transition from state W back to state S. Hence, the energy consumption for the transition from state W to state S, R_{WS}, is equal to, R_{WW}.

$$R_{WS} = R_{WW} \quad (10\text{-}98)$$

The energy consumption of the reader while overhearing a reply from a tag addressed to another reader, R_{OO}, is found using (10-99).

$$R_{OO} = T_B * \left(PW_{S-Rdr} + PW_{S-Tag} \right) \quad (10\text{-}99)$$

During the transition from state O to state R, the reader overhears the last byte of the reply. Therefore, the energy consumption for the transition from state O to state R, R_{OR}, is equal to, R_{OO}.

$$R_{OR} = R_{OO} \quad (10\text{-}100)$$

It is assumed that all tags in the network belong to the same group. Hence, they all respond to any point-to-point command. Therefore, M_P, is equal to 1 for the four topologies in this example. First, the set of tags, CT_x, for each topology is determined (x denotes which of the four topologies CT_x represents). The basic communication graph in Figure 10.13 is used to identify these sets because a tag with degree greater than one is in the set, CT_x, for each reader, x, connected to it. The set, CT, for each of the four topologies is shown below.

$$\begin{aligned} CT_1 &= \{\text{T3}, \text{T4}, \text{T5}, \text{T6}, \text{T7}, \text{T12}, \text{T13}\} \\ CT_2 &= \{\text{T3}, \text{T5}, \text{T6}, \text{T7}, \text{T10}, \text{T11}\} \\ CT_3 &= \{\text{T4}, \text{T5}, \text{T6}, \text{T11}, \text{T12}, \text{T13}, \text{T16}\} \\ CT_4 &= \{\text{T10}, \text{T16}\} \end{aligned} \tag{10-101}$$

Using (10-66), M_{OVER}, is calculated for each of the four topologies and is listed in Table 10.11. The degree of each of the tags is determined from the basic communication graph shown in Figure 10.13 is listed Table 10.12.

The number of messages forwarded to each reader, M_{Reader}, is computed using (10-67), and is shown in Table 10.13.

Table 10.11: Value of M_{OVER} for each of the four topologies.

Topology	M_{OVER} (Number of Messages)	Topology	M_{OVER} (Number of Messages)
TOP1	789	TOP3	686
TOP2	654	TOP4	174

The power consumption parameters only require determining the power consumption of a single tag or reader for a particular task (or step of a task). The power consumption is determined by the summing the power consumption of each component for each task (some components are dormant others are active). In this case, most of the power consumption parameters can be found using the power consumption parameters found for the single tag and

single reader models. The values of the power consumption parameters used in this example are listed in Table 10.14.

Table 10.12: Degree of each tag in the example network.

Tag	Degree	Tag	Degree
1	1	11	2
2	1	12	2
3	2	13	2
4	2	14	1
5	3	15	1
6	3	16	2
7	2	17	1
8	1	18	1
9	1	19	1
10	2	20	1

Table 10.13: Value of M_{Reader} for each topology.

Topology	M_{Reader}	Topology	M_{Reader}
TOP1	1137	TOP3	984
TOP2	976	TOP4	174

The values of the time parameters used in the single reader and associated tags topology are listed in Table 10.15. Because the tags and readers are identical to those studied in the previous example (single tags and single readers) the time for the reader and tag to process one byte of information is the same for both cases because the number of instructions to process one byte and the clock speeds have not changed. Similarly, the time to transmit one byte on the ISO

18000-7 reader-tag network has not changed and is identical to the previous cases. Again, the time required to transmit or receive one byte of information on the reader-to-reader network and the reader-to-outside world network is the same as the previous case.

Table 10.14: Value of the power consumption parameters for a reader and a tag.

Parameter	Power Consumption (mW)
$PW_{S\text{-}Rdr}$	111.1 mW
$PW_{S\text{-}Tag}$	74.2533 mW
$PW_{II\text{-}Rdr}$	187 mW
$PW_{II\text{-}Tag}$	74.2533 mW
$PW_{TX\text{-}Tag}$	150.1533 mW
$PW_{SI\text{-}Rdr}$	187 mW
$PW_{SI\text{-}Tag}$	74.2533 mW

Table 10.15: Values of the time parameters for the single reader and associated tags topology.

Parameter	Time (milliseconds)
$T_{Preamble}$	0.1296 ms
T_{WAKE}	2500 ms
$T_{P\text{-}Rdr}$	1.25 ms
$T_{P\text{-}Tag}$	2.5 ms
T_{B}	0.3249 ms
$T_{Byte\text{-}RtR}$	1.0416 ms
$T_{Byte\text{-}Out}$	1.0416 ms

Each reader in the network sends, S_{Out}, messages to the outside world. The values of, S_{Out}, and M_{Out}, for each reader is taken from the single reader example, and the values of, S_{Out},

and M_{Out}, for each topology (reader because each topology, TOPx, is built around reader, Rx) are listed in Table 10.16. The remaining parameters for this example are listed in Table 10.17. The energy consumed over the period of one day for each of the four topologies and the total energy consumed by the network (sum of energy consumed of each topology) are listed in Table 10.18. The energy consumed by the entire network calculated using topologies is within 4 % of that found by evaluating each tag and reader separately.

Table 10.16: Number of messages each topology sends to the outside world per day.

Parameter	**Topology TOP1**	**Topology TOP2**	**Topology TOP3**	**Topology TOP4**
S_{Out} (Number of Messages Per Day)	15	54	32	37
M_{OUT} (Messages)	10	24	48	20

Table 10.17: Miscellaneous parameter values for single reader and associated tags topology.

Parameter	**Value**
L_{Reader} (Bytes)	8 Bytes
L_{Out} (Bytes)	20 Bytes

Table 10.18: Energy consumed over 1 day for each of the four single reader and associated tag topologies and for the entire network.

Topology	Energy Consumed in One Day (mJ)
TOP1	44347884.42 mJ
TOP2	40488969.57 mJ
TOP3	44702959.56 mJ
TOP4	32104372.34 mJ
Entire Network	161644185.89 mJ

10.2.10 Step 8 for the Single Reader and Associated Tags Topological Entities

The interactions between the four single reader and associated tags topological entities are defined in this section. This is step 8 in the algorithm. The interactions will be used to help to identify the multi-reader topological entity in the following section. The basic communication graph of the ISO 18000-7 example network is repeated in Figure 10.14 for clarity.

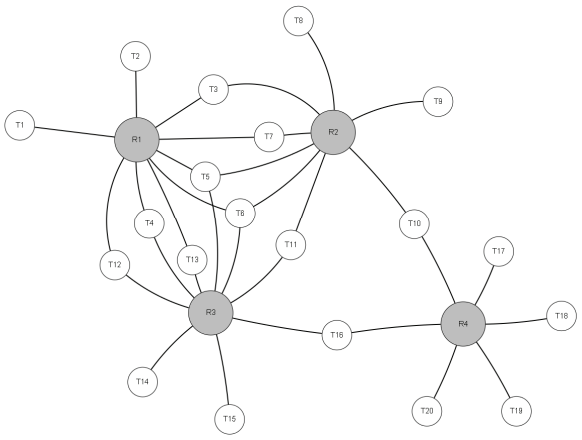

Figure 10.14: Basic communication graph of the example ISO 18000-7 network (readers are shaded and tags are not shaded).

The four single reader and associated tags topological entities are shown in the following four figures. The single reader and associated tags topological entity generated around reader R1, TOP1, are shown in Figure 10.15. The single reader and associated tags topological entity generated around reader R2, TOP2, are shown in Figure 10.16. The single reader and associated tags topological entity generated around reader R3, TOP3, are shown in Figure 10.17. The single reader and associated tags topological entity generated around reader R4, TOP4, are shown in Figure 10.18. The basic communication graph of the example network, shown in Figure 10.14, when covered by the four single reader and associated tags topological entities (TOP1, TOP2, TOP3, and TOP4) is shown in Figure 10.19.

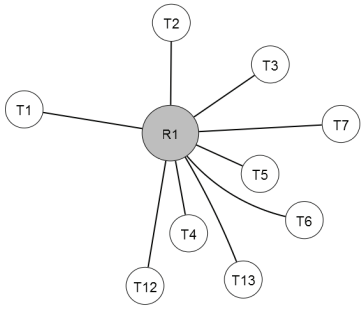

Figure 10.15: Entities contained in topology, TOP1, built around reader R1.

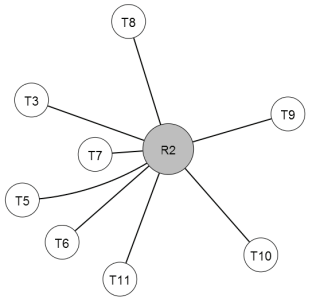

Figure 10.16: Entities contained in topology, TOP2, built around reader R2.

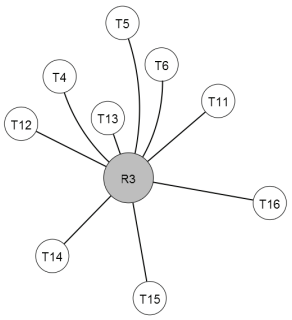

Figure 10.17: Entities contained in topology, TOP3, built around reader R3.

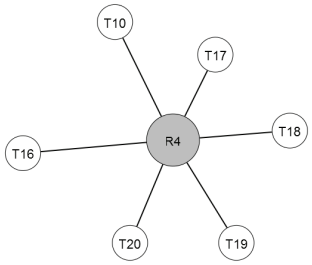

Figure 10.18: Entities contained in topology, TOP4, built around reader R4.

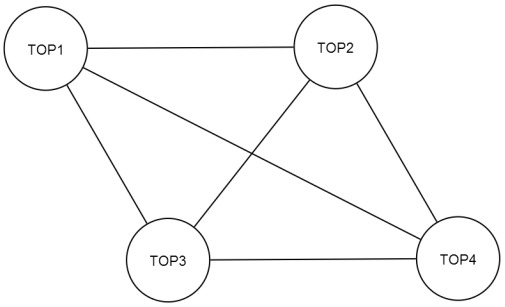

Figure 10.19: Basic communication graph of the four single reader and associated tags topological entities.

The readers communicate with each other, thus linking all the single reader and associated tags topologies together forming a fully connected mesh topology. The highest level of abstraction combines all four single reader and associated tags topologies into a single topological entity.

10.2.11 Step 9: Identification of the Multi-Reader Topological Entity

The multi-reader topological entity is identified based on the tasks (Section 10.2.8) and interactions of the single reader and associated tags topological entities (Section 10.2.10). This is step 9 in the algorithm. The interactions between the single reader and associated tags topological entities is shown in Figure 10.19. In this example, the readers form a fully connected mesh network and this fully connected mesh topology can be represented by a larger topological entity. This level of abstraction allows the example network to be represented as a single topological entity. Once the multi-reader topological entity is identified it is replaced with the topological entity reducing the order (number of Markov processes) of the basic communication graph. With the multi-reader fundamental topological entity identified existing algorithms can be used to search the basic communication graph to identify these topological entities. These algorithms form the basis for CAD tools for integrated circuit design [46]. In this example, the 4 single reader and associated tags topological entities shown in Figure 10.19 can be replaced with a single multi-reader topological entity as shown in Figure 10.20. Thus, using this topology only one entity must be evaluated to obtain the energy consumption of the network, as shown in Figure 10.20, rather than evaluating the twenty-four entities originally required, as shown in Figure 10.13.

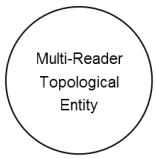

Figure 10.20: The entire ISO 18000-7 example network is covered using a single multi-reader topological entity.

10.2.12 Steps 6 through 9 for the Multi-Reader Topological Entity

Development of the model describing the energy consumption of the multi-reader topological entity requires that steps 6 through 9 of the algorithm be repeated for the multi-reader topological entity. The multi-reader topological entity must still perform the same tasks as the previous single reader and associated tags topological entity. The multi-reader topological entity must still communicate with other readers, both within the topological entity and in other topological entities, and still communicates with the outside world. The readers within this topological entity still interrogate the tags in the topology, and readers still overhear replies that are addressed to readers other than themselves. Thus, the Markov process for this topology is identical to that for the single reader and associated tags topological entity and is shown below in Figure 10.21. The probability matrix, P, and the reward matrix, R, have the same form as the previous topology.

$$P = \begin{bmatrix} P_{SS} & P_{SI} & P_{SR} & P_{SW} & P_{SO} \\ P_{IS} & P_{II} & 0 & 0 & 0 \\ P_{RS} & 0 & P_{RR} & 0 & 0 \\ P_{WS} & 0 & 0 & P_{WW} & 0 \\ 0 & 0 & P_{OR} & 0 & P_{OO} \end{bmatrix} \tag{10-102}$$

$$R = \begin{bmatrix} R_{SS} & R_{SI} & R_{SR} & R_{SW} & R_{SO} \\ R_{IS} & R_{II} & 0 & 0 & 0 \\ R_{RS} & 0 & R_{RR} & 0 & 0 \\ R_{WS} & 0 & 0 & R_{WW} & 0 \\ 0 & 0 & R_{OR} & 0 & R_{OO} \end{bmatrix} \tag{10-103}$$

There are, N_R, readers within each multi-reader topological entity. The transition probabilities for the multi-reader topological entity are found using similar methods as those for the single reader and associated tags topological entity where. However, all contributions from all, N_R, readers must be taken into account. Each of the, N_R, readers has a single reader and associated tags topological entity built around it. The Markov process for the multi-reader topological entity is identical, in structure, to that for the single reader and associated tags topological entity. Therefore, the transition probabilities for each of the, N_R, single reader and

associated tags topological entities that make up the multi-reader topology are simply summed and divided by, N_R, to obtain the multi-reader transition probabilities. The rewards matrix, for the multi-reader topological entity must account for the energy consumption of the transitions for each of the, N_R, single reader and associated tags topological entities it contains. Hence, the rewards matrix for the multi-reader topological entity is simply the sum of the reward matrices for each of the, N_R, single reader and associated tags topological entities.

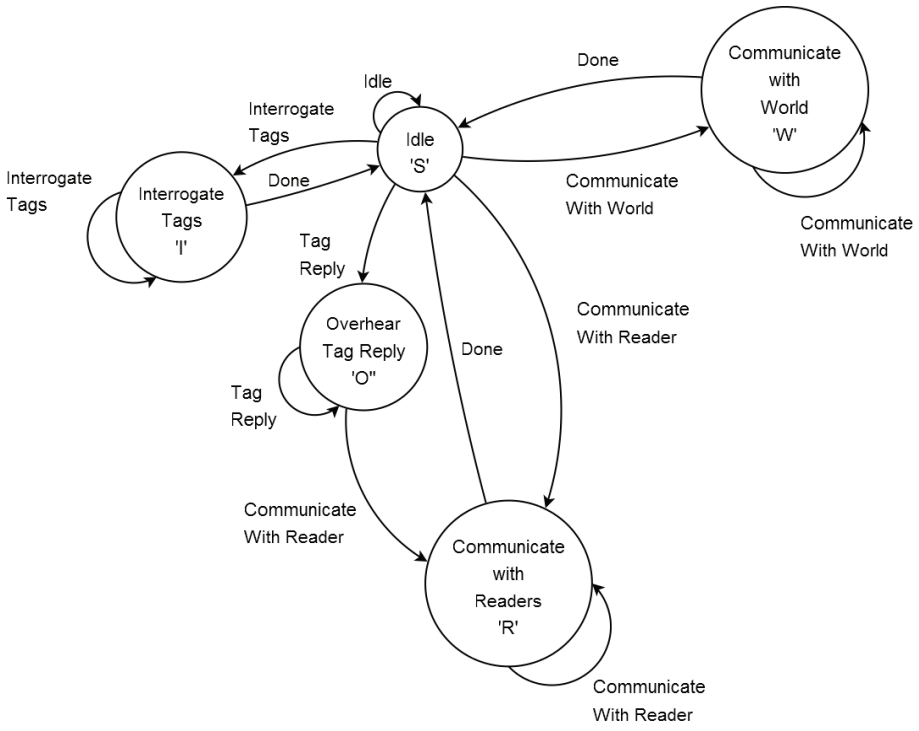

Figure 10.21: Markov process for highest-level topology containing multiple readers.

10.2.13 Step 10: Covering the Network with a Single Entity

In this example, the multi-reader topological entity consists of all four readers and by extension all four single reader and associated tags topological entities. Thus, using the Multi-Reader

topological entity the entire network can be covered using a single entity and the algorithm is finished.

The energy consumption of the entire network can now be found by evaluating only the single Multi-Reader topological entity. Using the results from the four single reader and associated topologies calculated in the previous section, the energy consumption for the multi-reader topology representing the entire network is, 161862122.37 mJ, which is very close to that calculated for the network composed of four single reader and associated tags topologies.

10.3 SUMMARY OF THE ENERGY CONSUMPTION

The energy consumed by the network for a period of 1 day was calculated using three different methods. First, the energy consumed by the network was calculated by first calculating the energy consumed by each base level entities and then summing the values to obtain the energy consumption for the entire network (Section 10.2.4 for the tag and Section 10.2.5 for the reader). Next, the four single reader and associated tags topological were used to cover the network. Again the energy consumption for the period of 1 day was computed for each of the four single reader and associated tags topological entities and then summed to arrive at the energy consumption of the entire network (Section 10.2.9). Finally, the multi-reader topological entity was used to cover the entire network with a single entity and the energy consumption of the multi-reader topological entity was calculated for the period of 1 day (Sections 10.2.12 and 10.2.13). The energy consumed for each case is listed in Table 10.19. The percent difference, D, is computed as follows, where, $E_{Base\text{-}Level}$, is the energy consumed calculated using the base level entities, and E_{Test}, is the energy consumed using one of the other two sets of topological entities (four single reader and associated tags topological entities or one multi-reader topological entity).

$$D = \frac{\left|E_{\text{Base-Level}} - E_{\text{Test}}\right|}{E_{\text{Base-Level}}} \qquad (10\text{-}104)$$

Table 10.19: Energy consumed calculated using the three different sets of entities and percent difference from the energy consumption of the base level entities.

Entity	Energy Consumed (mJ)	Percent Difference (%)
Base Level Entities	161644185.89 mJ	0 %
Four Single Reader and Associated Tags Topological Entities	161644185.89 mJ	3.9054 %
One Multi-Reader Topological Entity	161862122.37 mJ	3.7758 %

The results show that the percent difference is no more than 3.89 % in the worst case (using the four single reader and associated tags topological entities) and the method of using the topological entities is considered accurate.

10.4 SUMMARY OF STEPS IN ALGORITHM TO IDENTIFY TOPOLOGICAL ENTITIES

The steps in the algorithm to identify the topological entities in the ISO 18000-7 example network are summarized in this section. First, Step 1, identification of the base level entities, was performed in Section 10.2.1. Step 2, identification of the tasks of the base level entities was performed in Section 10.2.2. Step 3, development of the Markov processes with expressions for the probability and rewards matrices for each type of base level entity were performed in Section 10.2.4 for the tag and in Section 10.2.5 for the Reader. Step 4, identification of the interactions between the base level entities, occurs in Section 10.2.6. Step 5, identification of the first-level topologies occurs in Section 10.2.7. Step 6, identification of the tasks each topological entity performs was accomplished in Section 10.2.8. Step 7, development of Markov processes and expressions for the probability and rewards matrices was performed in Section 10.2.9. Step 8, identification of interaction between topological entities, was presented in Section 10.2.10. Step 9, identification of the next higher topological entities, was presented in Section 10.2.11. Steps 6

through 9 were repeated for the Multi-Reader topological entity in Section 10.2.12. Step 10, is satisfied because only one Multi-Reader topological entity was required to cover the entire network, at the end on Section 10.2.13.

10.5 EVALUATION TIME FOR LARGER NETWORKS

To determine how the execution time of the method developed in this work scales with the size of the network being evaluated, the example ISO 18000-7 was doubled and doubled again to generate three different size networks. The single size network has the same size as that in the example presented in the Section 10.2. The double network consists of two single size networks with connections between tags 1 and 2 and tags 38 and 39 (in the new numbering, in the single network these are tags 18 and 19) and the basic communication graph of the double size network is shown in Figure 10.22.

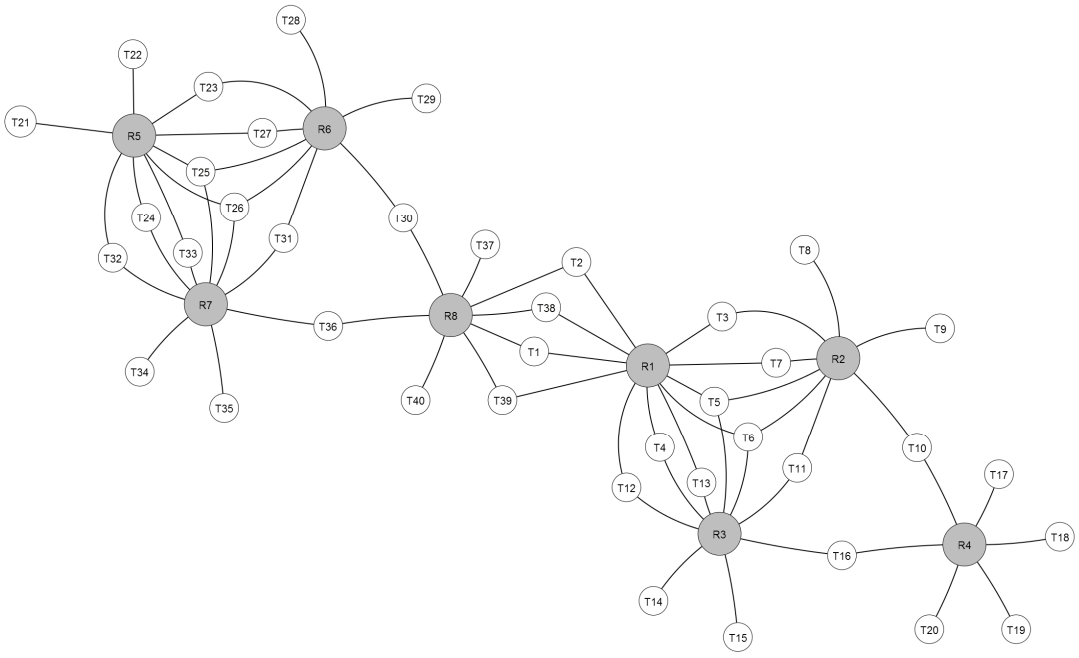

Figure 10.22: Basic communication graph of the double size example network.

There are a total of eight single reader and associated tags topological entities contained in the double network. Each single reader and associated tag topological entity is a star topology with the reader acting as the central entity with all tags within communication of the reader as members of the topology. With the single reader and associated tags fundamental topological entity identified, existing algorithms can be used to identify the eight single reader and associated tags topological entities in the double size example network shown in Figure 10.22. These algorithms form the basis for CAD tools for integrated circuit design [46]. The eight individual single reader and associated tags topological entities are illustrated in the following eight figures.

The first single reader and associated tags topological entity, TOP1, is centered around reader R1, and is illustrated in Figure 10.23. The second single reader and associated tags topological entity, TOP2, is centered around reader R2, and is illustrated in Figure 10.24. The second single reader and associated tags topological entity, TOP3, is centered around reader R3, and is illustrated in Figure 10.25. The second single reader and associated tags topological entity, TOP4, is centered around reader R4, and is illustrated in Figure 10.26. The second single reader and associated tags topological entity, TOP5, is centered around reader R5, and is illustrated in Figure 10.27. The second single reader and associated tags topological entity, TOP6, is centered around reader R6, and is illustrated in Figure 10.28. The second single reader and associated tags topological entity, TOP7, is centered around reader R7, and is illustrated in Figure 10.29. The second single reader and associated tags topological entity, TOP8, is centered around reader R8, and is illustrated in Figure 10.30. The network shown in Figure 10.22 can be covered using the eight single reader and associated tags topological entities shown in the previous eight figures. The network covered using the eight single reader and associated tags topological entities (TOP1, TOP2, TOP3, TOP4, TOP5, TOP6, TOP7, and TOP8) is shown in Figure 10.31.

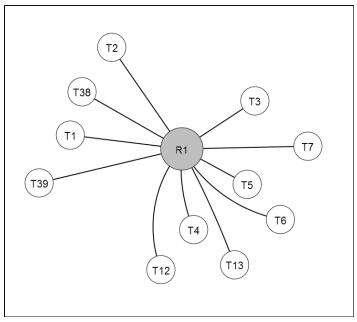

Figure 10.23: Single reader and associated tags topology, TOP1, centered around reader R1 in the double size network.

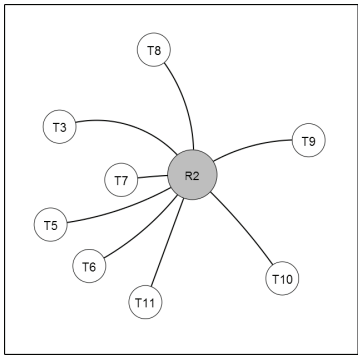

Figure 10.24: Single reader and associated tags topology, TOP2, centered around reader R2 in the double size network.

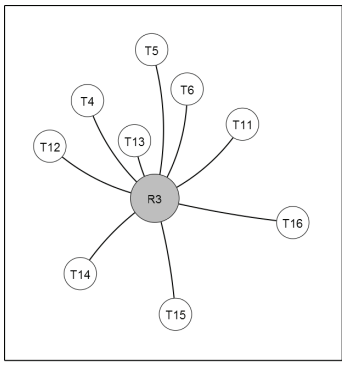

Figure 10.25: Single reader and associated tags topology, TOP3, centered around reader R3 in the double size network.

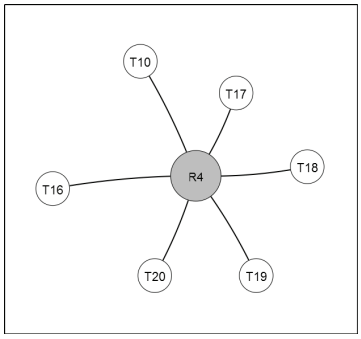

Figure 10.26: Single reader and associated tags topology, TOP4, centered around reader R4 in the double size network.

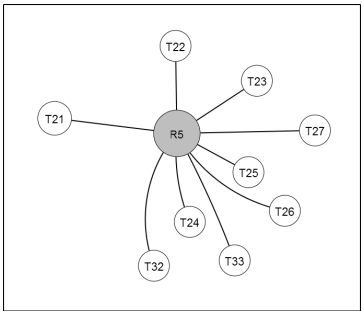

Figure 10.27: Single reader and associated tags topology, TOP5, centered around reader R5 in the double size network.

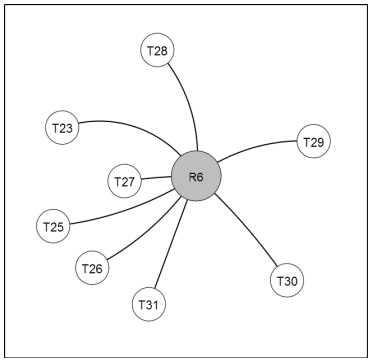

Figure 10.28: Single reader and associated tags topology, TOP6, centered around reader R6 in the double size network.

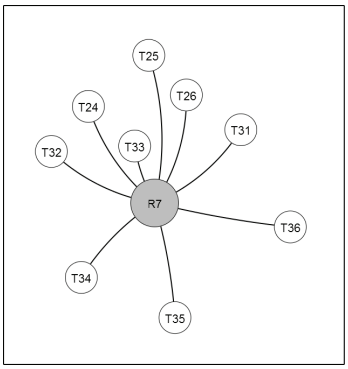

Figure 10.29: Single reader and associated tags topology, TOP7, centered around reader R7 in the double size network.

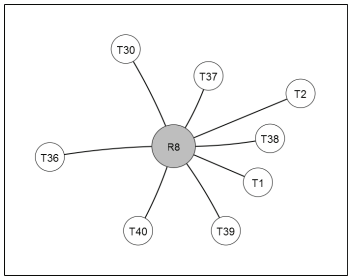

Figure 10.30: Single reader and associated tags topology, TOP8, centered around reader R8 in the double size network.

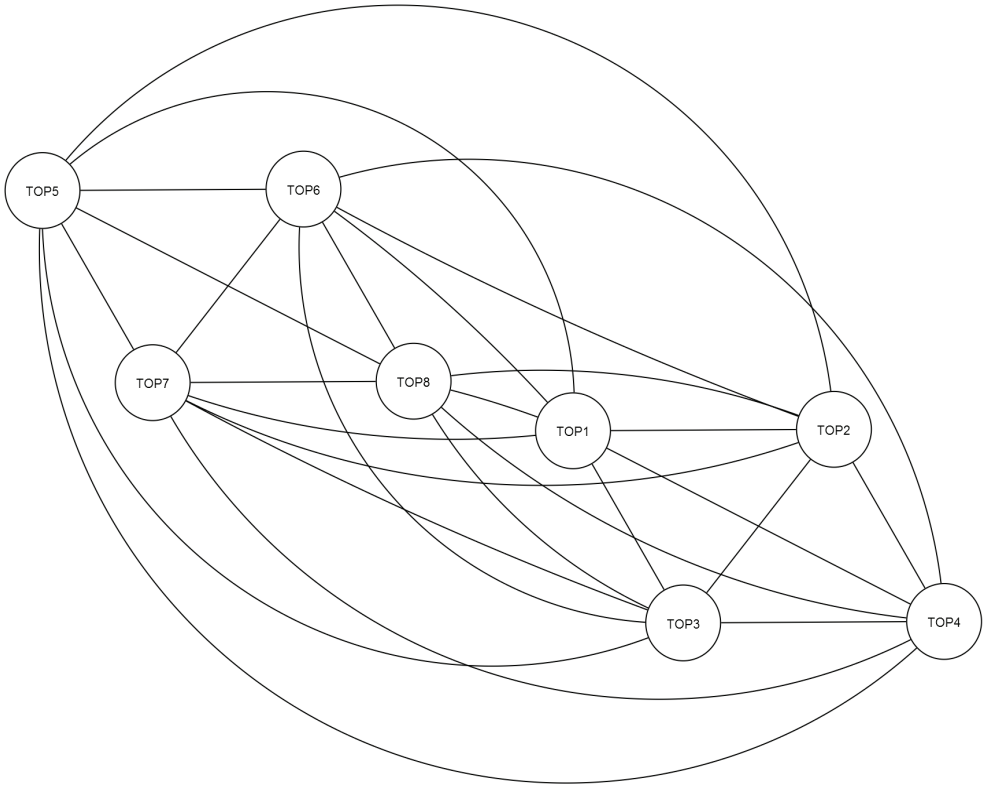

Figure 10.31: Double size network covered with eight single reader and associated tags topological entities.

With the single reader and associated tags fundamental topological entity identified existing algorithms can be used to search the basic communication graph to identify these topological entities. These algorithms form the basis for CAD tools for integrated circuit design [46]. The order (number of Markov processes) of the double size network is reduced from the 48 base level entities as shown in Figure 10.22 to 8 single reader and associated tags topological entities as shown in Figure 10.31. The covering of the double size entity by the eight single reader and associated tags topological entities results in a fully connected mesh. This network can be reduced to a single multi-reader topological entity that contains the eight single reader and associated topological entities (TOP1, TOP2, TOP3, TOP4, TOP5, TOP6, TOP7, and TOP8) is shown in Figure 10.32.

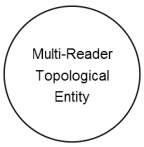

Figure 10.32: The double size network can be covered by a single multi-reader topological entity.

The tasks and the Markov process for the multi-reader topological entity is identical to the Markov process developed for the multi-reader topological entity in Section 10.2.12 is shown in Figure 10.33. In this example, the 8 single reader and associated tags topological entities shown in Figure 10.31 can be replaced with a single multi-reader topological entity as shown in Figure 10.32. This reduces the order of the network from 48, as shown in Figure 10.22, to 1, as shown in Figure 10.32.

The probability and reward matrices for the multi-reader topological entity are shown below and are calculated from the eight single reader and associated tags topological entities as described in Section 10.2.10. The probability matrix is computed by summing the probability matrices for each of the eight single reader and associated tags topological entities together and then dividing by eight (the number of single reader and associated tags topological entities).

$$P = \begin{bmatrix} P_{SS} & P_{SI} & P_{SR} & P_{SW} & P_{SO} \\ P_{IS} & P_{II} & 0 & 0 & 0 \\ P_{RS} & 0 & P_{RR} & 0 & 0 \\ P_{WS} & 0 & 0 & P_{WW} & 0 \\ 0 & 0 & P_{OR} & 0 & P_{OO} \end{bmatrix} \tag{10-105}$$

$$R = \begin{bmatrix} R_{SS} & R_{SI} & R_{SR} & R_{SW} & R_{SO} \\ R_{IS} & R_{II} & 0 & 0 & 0 \\ R_{RS} & 0 & R_{RR} & 0 & 0 \\ R_{WS} & 0 & 0 & R_{WW} & 0 \\ 0 & 0 & R_{OR} & 0 & R_{OO} \end{bmatrix} \tag{10-106}$$

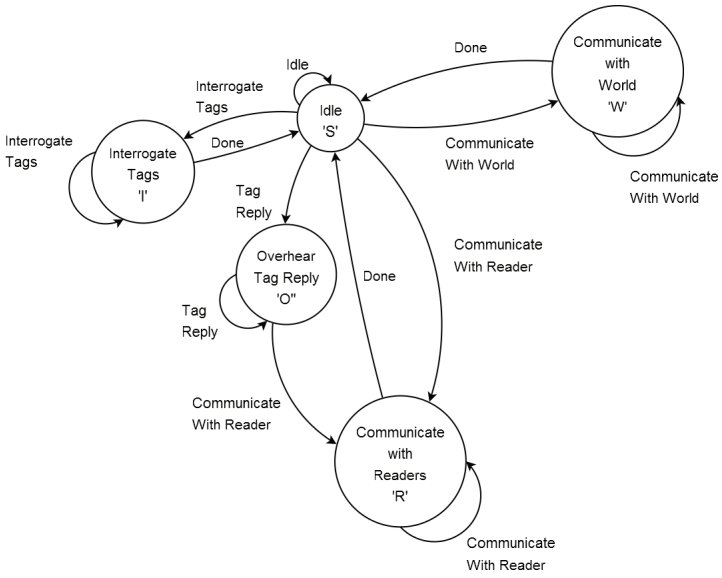

Figure 10.33: Markov process for the multi-reader topological entity.

The energy consumption of the network using two different groups of entities. First, the energy consumption of the network is calculated using the base entities and then summing the energy consumed by each base entity together to obtain the energy consumed by the entire network. The models developed for the base entities (single tag and single reader) are used for each of the base entities in the double example network. Then, the energy consumption of the network is computed using the single multi-reader topological entity, using the model developed in Section 10.2.10. The parameters for the double size network are identical to those for the single size network examined in Section 10.2. The parameters for the base entities in the double size network correspond to those for the single size network. The parameters for tags 1 through 20 in the double size network are identical to those for tags 1 through 20 (Section 10.2.9) in the single size network respectively. The parameters for tags 21 through 40 in the double size network are identical to those for tags 1 through 20 (Section 10.2.9) in the single size network, respectively. The double size network is created by duplicating the single size network and connecting the two together. The duplicate network has the same parameters as the original network. The parameters for readers 1 through 4 in the double size network are identical to those for readers 1 through 4 (Section 10.2.9) in the single size network, respectively. Similarly, the parameters for readers 5 through 8 in the double size network are identical to the parameters for

readers 1 through 4 (Section 10.2.9) in the single size network, respectively. The energy consumption for the double size network and execution time for each of the two methods is presented and analyzed at the end of this section.

The quadruple network combines two double networks in a similar fashion with tags 21 and 22 connected to tags 58 and 59 in the other network and the basic communication graph of the quadruple size network is shown in Figure 10.34.

The quadruple size network contains sixteen single reader and associated tags topological entities. These sixteen single reader and associated tags topological entities can be combined into a single multi-reader topological entity. With the single reader and associated tags fundamental topological entity identified, existing algorithms can be used to identify the sixteen single readers and associated tags topological entities in the quadruple size network shown in Figure 10.34. These algorithms form the basis for CAD tools for integrated circuit design [46]. The sixteen single reader and associated tags topological entities in the quadruple size network are shown in the following sixteen figures.

The first single reader and associated tags topological entity, TOP1, is centered around reader R1, and is illustrated in Figure 10.35. The second single reader and associated tags topological entity, TOP2, is centered around reader R2, and is illustrated in Figure 10.36. The second single reader and associated tags topological entity, TOP3, is centered around reader R3, and is illustrated in Figure 10.37. The second single reader and associated tags topological entity, TOP4, is centered around reader R4, and is illustrated in Figure 10.38. The second single reader and associated tags topological entity, TOP5, is centered around reader R5, and is illustrated in Figure 10.39. The second single reader and associated tags topological entity, TOP6, is centered around reader R6, and is illustrated in Figure 10.40. The second single reader and associated tags topological entity, TOP7, is centered around reader R7, and is illustrated in Figure 10.41. The second single reader and associated tags topological entity, TOP8, is centered around reader R8, and is illustrated in Figure 10.42. The second single reader and associated tags topological entity, TOP9, is centered around reader R9, and is illustrated in Figure 10.43. The second single reader and associated tags topological entity, TOP10, is centered around reader R10, and is illustrated in Figure 10.44. The second single reader and associated tags topological entity, TOP11, is centered around reader R11, and is illustrated in Figure 10.45. The second single reader and associated tags topological entity, TOP12, is centered around reader

R12, and is illustrated in Figure 10.46. The second single reader and associated tags topological entity, TOP13, is centered around reader R13, and is illustrated in Figure 10.47. The second single reader and associated tags topological entity, TOP14, is centered around reader R14, and is illustrated in Figure 10.48. The second single reader and associated tags topological entity, TOP15, is centered around reader R15, and is illustrated in Figure 10.49. The second single reader and associated tags topological entity, TOP16, is centered around reader R16, and is illustrated in Figure 10.50.

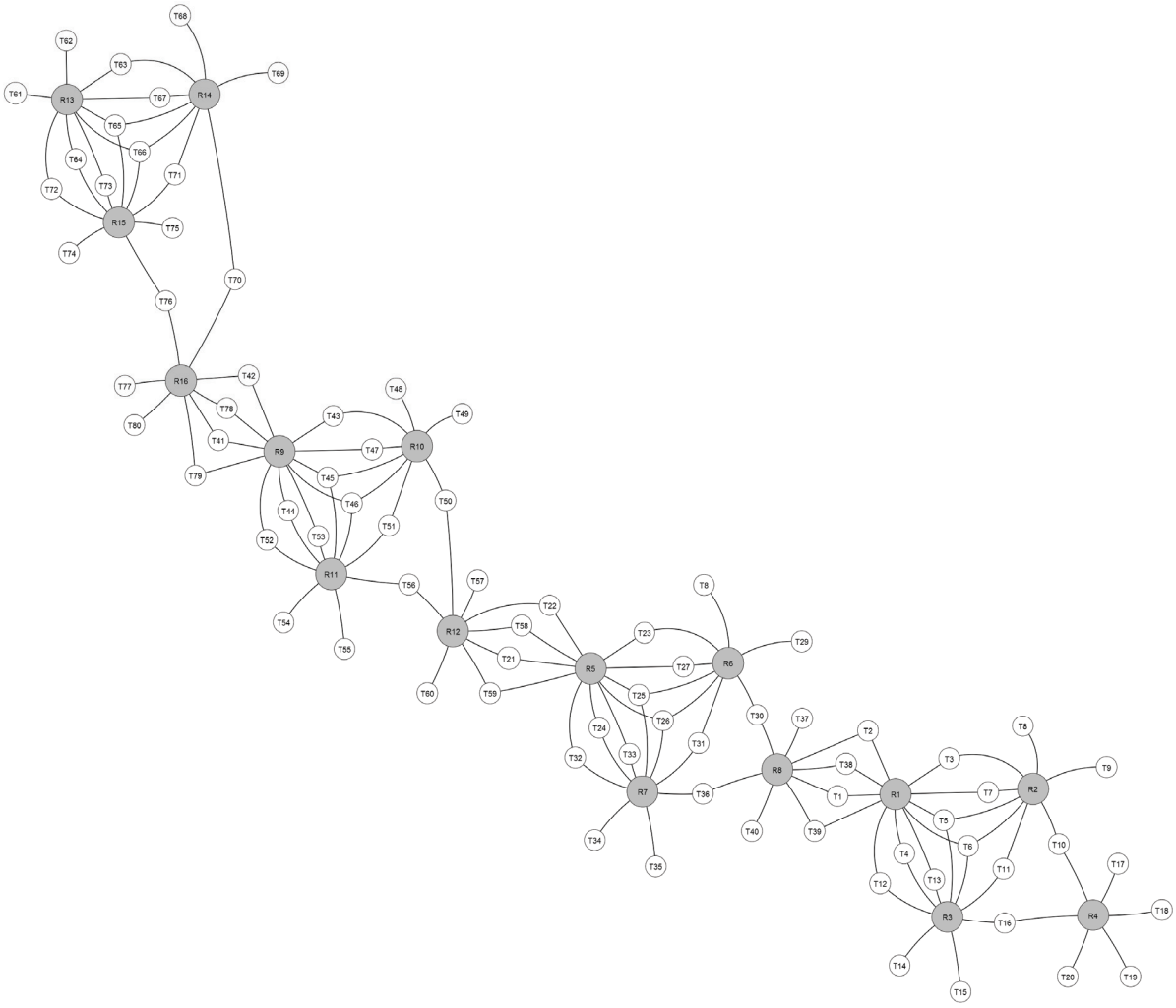

Figure 10.34: Basic communication graph for the quadruple size example network (not to scale).

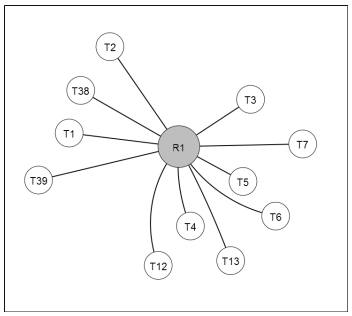

Figure 10.35: Single reader and associated tags topology, TOP1, centered around reader R1 in the quadruple size network.

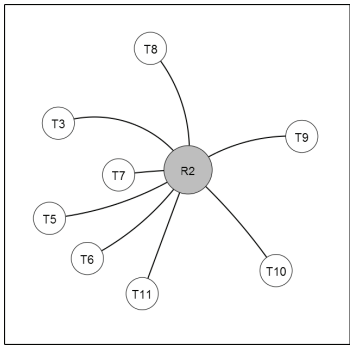

Figure 10.36: Single reader and associated tags topology, TOP2, centered around reader R2 in the quadruple size network.

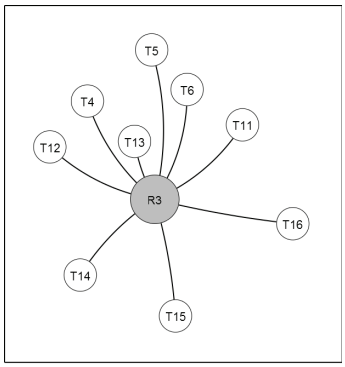

Figure 10.37: Single reader and associated tags topology, TOP3, centered around reader R3 in the quadruple size network.

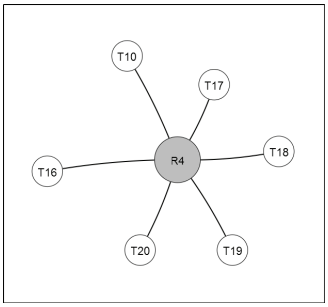

Figure 10.38: Single reader and associated tags topology, TOP4, centered around reader R4 in the quadruple size network.

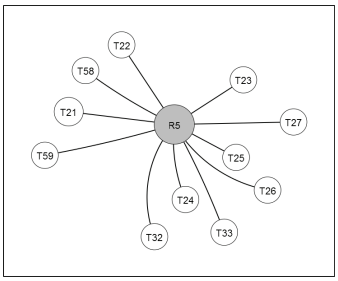

Figure 10.39: Single reader and associated tags topology, TOP5, centered around reader R5 in the quadruple size network.

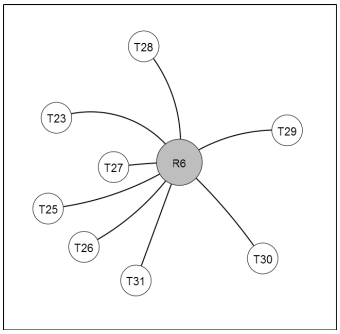

Figure 10.40: Single reader and associated tags topology, TOP6, centered around reader R6 in the quadruple size network.

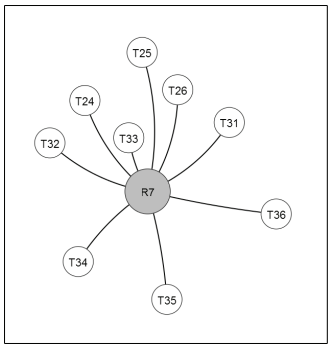

Figure 10.41: Single reader and associated tags topology, TOP7, centered around reader R7 in the quadruple size network.

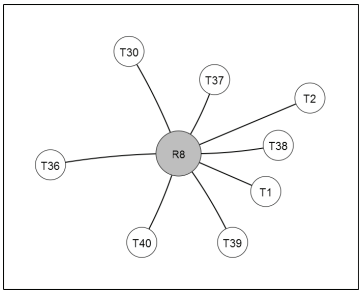

Figure 10.42: Single reader and associated tags topology, TOP8, centered around reader R8 in the quadruple size network.

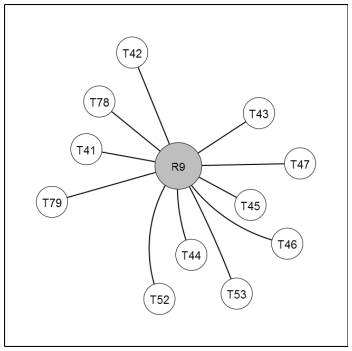

Figure 10.43: Single reader and associated tags topology, TOP9, centered around reader R9 in the quadruple size network.

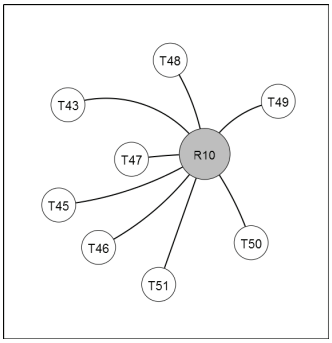

Figure 10.44: Single reader and associated tags topology, TOP10, centered around reader R10 in the quadruple size network.

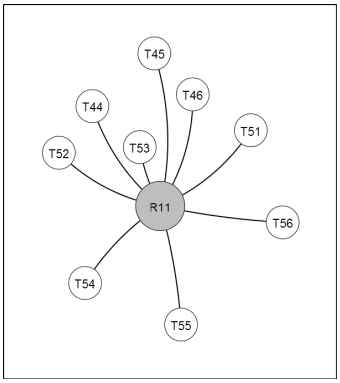

Figure 10.45: Single reader and associated tags topology, TOP11, centered around reader R11 in the quadruple size network.

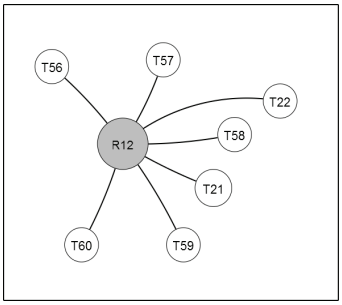

Figure 10.46: Single reader and associated tags topology, TOP12, centered around reader R12 in the quadruple size network.

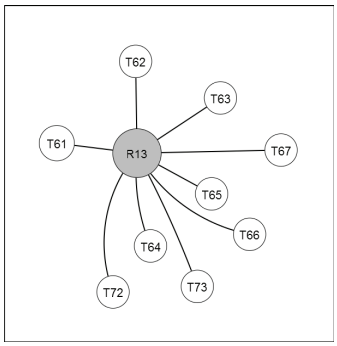

Figure 10.47: Single reader and associated tags topology, TOP13, centered around reader R13 in the quadruple size network.

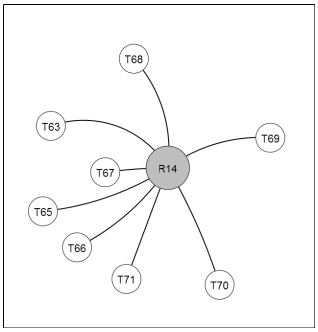

Figure 10.48: Single reader and associated tags topology, TOP14, centered around reader R14 in the quadruple size network.

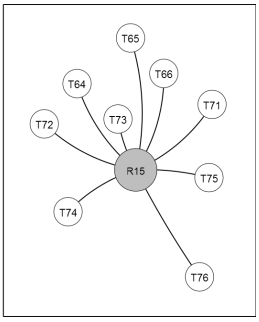

Figure 10.49: Single reader and associated tags topology, TOP15, centered around reader R15 in the quadruple size network.

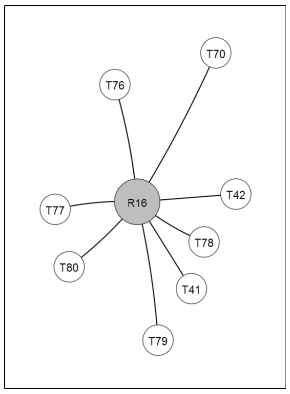

Figure 10.50: Single reader and associated tags topology, TOP16, centered around reader R16 in the quadruple size network.

The quadruple size network shown in Figure 10.34 can be covered using the sixteen single reader and associated tags topological entities shown in the previous sixteen figures. The sixteen single reader and associated tags topological entities covering the quadruple size network are shown in Figure 10.51. All sixteen readers are connected forming a fully connected mesh of the 16 readers and associated tags topological entities.

The covering of the quadruple size entity by the sixteen single reader and associated tags topological entities results in a fully connected mesh. Once a single reader and associated tags topological entity is identified it is replaced with the topological entity reducing the order of the basic communication graph. With the single reader and associated tags fundamental topological entity identified existing algorithms can be used to search the basic communication graph to identify these topological entities. These algorithms form the basis for CAD tools for integrated circuit design [46]. The order of the quadruple size graph is reduced from the 96 base level entities as shown in Figure 10.34 to 16 single reader and associated tags topological entities as shown in Figure 10.51.

All of the connections are between the single reader and associated tags topological entities in Figure 10.51 are not shown because showing all connections would prevent the figure from being readable. This network can be reduced to a single multi-reader topological entity that contains the eight single reader and associated topological entities (TOP1, TOP2, TOP3, TOP4, TOP5, TOP6, TOP7, TOP8, TOP9, TOP10, TOP11, TOP12, TOP13, TOP14, TOP15, and

TOP16) is shown in Figure 10.52. Again, once the multi-reader fundamental topological entity is identified the basic communication graph can be searched for the multi-reader topological entities. In this example, the 16 single reader and associated tags topological entities shown in Figure 10.51 can be replaced with a single multi-reader topological entity as shown in Figure 10.52.

The tasks and the Markov process for the multi-reader topological entity is identical to the Markov process developed for the multi-reader topological entity in Section 10.2.10 is shown in Figure 10.53. The probability and reward matrices for the multi-reader topological entity are shown below and are calculated from the sixteen single reader and associated tags topological entities as described in Section 10.2.10. The probability matrix is computed by summing the probability matrices for each of the sixteen single reader and associated tags topological entities together and then dividing by sixteen (the number of single reader and associated tags topological entities in the quadruple size network).

$$P = \begin{bmatrix} P_{SS} & P_{SI} & P_{SR} & P_{SW} & P_{SO} \\ P_{IS} & P_{II} & 0 & 0 & 0 \\ P_{RS} & 0 & P_{RR} & 0 & 0 \\ P_{WS} & 0 & 0 & P_{WW} & 0 \\ 0 & 0 & P_{OR} & 0 & P_{OO} \end{bmatrix} \tag{10-107}$$

$$R = \begin{bmatrix} R_{SS} & R_{SI} & R_{SR} & R_{SW} & R_{SO} \\ R_{IS} & R_{II} & 0 & 0 & 0 \\ R_{RS} & 0 & R_{RR} & 0 & 0 \\ R_{WS} & 0 & 0 & R_{WW} & 0 \\ 0 & 0 & R_{OR} & 0 & R_{OO} \end{bmatrix} \tag{10-108}$$

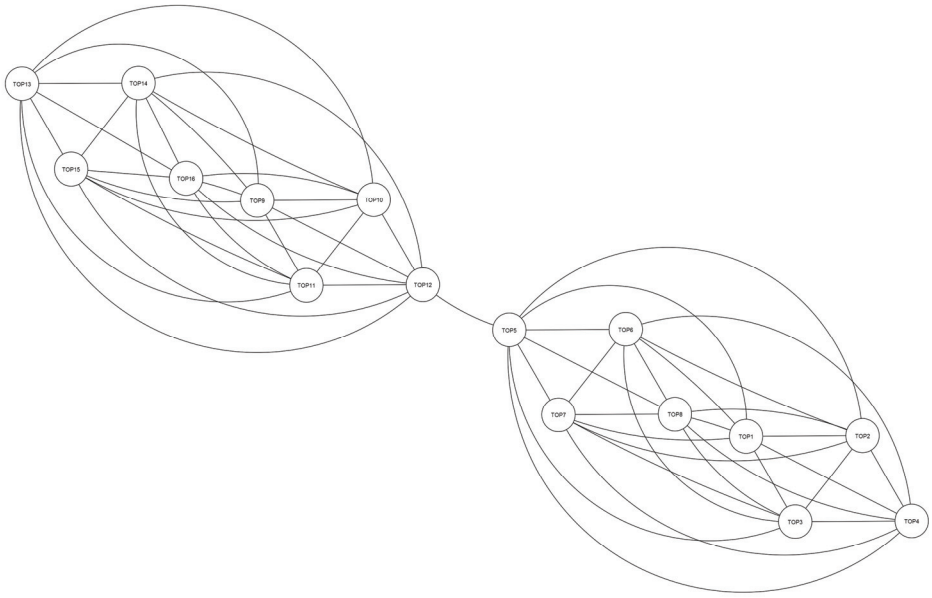

Figure 10.51: Covering of the quadruple size network with the sixteen single reader and associated tags topological entities.

Figure 10.52: The double size network can be covered by a single multi-reader topological entity.

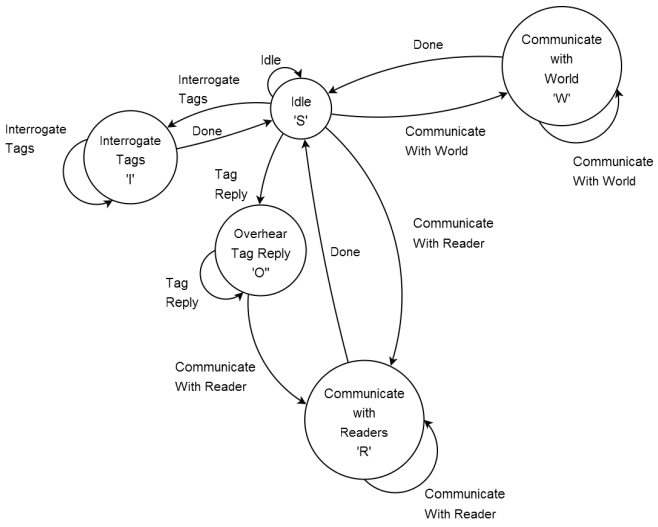

Figure 10.53: Markov process for the multi-reader topological entity.

The energy consumption of the network using two different groups of entities. First, the energy consumption of the network is calculated using the base entities and then summing the energy consumed by each base entity together to obtain the energy consumed by the entire network. The models developed for the base entities (single tag and single reader) are used for each of the base entities in the quadruple example network. Then, the energy consumption of the network is computed using the single multi-reader topological entity, using the model developed in Section 10.2.10. The parameters for the quadruple size network are identical to those for the single size network examined in Section 10.2. The parameters for the base entities in the quadruple size network correspond to those for the single size network. The parameters for tags 1 through 20 in the quadruple size network are identical to those for tags 1 through 20 (Section 10.2.9) in the single size network respectively. The parameters for tags 21 through 40 in the double size network are identical to those for tags 1 through 20 (Section 10.2.9) in the single size network, respectively. Likewise, the parameters for tags 41 through 60 in the quadruple network are identical to those for tags 1 though 20 (Section 10.2.9) in the single size network, respectively. Again, the parameters for tags 61 through 80 in the quadruple network are identical to those for tags 1 though 20 (Section 10.2.9) in the single size network, respectively. The quadruple size network is created by duplicating the double size network and connecting the two together. The duplicate network has the same parameters as the original network. The

parameters for readers 1 through 4 in the quadruple size network are identical to those for readers 1 through 4 (Section 10.2.9) in the single size network, respectively. Similarly, the parameters for readers 5 through 8 in the quadruple size network are identical to the parameters for readers 1 through 4 (Section 10.2.9) in the single size network, respectively. Likewise, the parameters for readers 9 through 12 in the quadruple size network are identical to the parameters for readers 1 through 4 (Section 10.2.9) in the single size network, respectively. Again, the parameters for readers 13 through 16 in the quadruple size network are identical to the parameters for readers 1 through 4 (Section 10.2.9) in the single size network, respectively. The energy consumption for the quadruple size network and execution time for each of the two methods is presented and analyzed at the end of this section.

The time required to calculate the energy consumption of the model using the base level entities and using the multiple reader topological entities was found. Each model was executed 100 times in Matlab, and the execution time for all 100 evaluations was obtained. The average time for a single evaluation is found by simply dividing the overall time by 100. Running each model 100 times minimizes any effects on execution time resulting from start-up times and context switches within the computing environment. The time to evaluate the single size example network 100 times and the average time for one evaluation to determine the energy consumed over one day are listed in Table 10.20. The time to evaluate the double size example network 100 times and the average time to evaluate the energy consumption over 1 day is given in Table 10.21. The time required to evaluate the quadruple size example network 100 times and the average time for evaluation of the energy consumed by the network over 1 day is given in Table 10.22.

Table 10.20: Time to evaluate the single size example networks.

Entity	Time for 100 Evaluations (seconds)	Average Time for 1 Evaluation (seconds)
Base Level Entities	1.881434 seconds	0.018814 seconds
Multi-Reader Topology	0.470980 seconds	0.004710 seconds

Table 10.21: Time to evaluate the double size example networks.

Entity	Time for 100 Evaluations (seconds)	Average Time for 1 Evaluation (seconds)
Base Level Entities	3.670574 seconds	0.036706 seconds
Multi-Reader Topology	0.923373 seconds	0.009234 seconds

Table 10.22: Time to evaluate the quadruple size example networks.

Entity	Time for 100 Evaluations (seconds)	Average Time for 1 Evaluation (seconds)
Base Level Entities	7.286213 seconds	0.072862 seconds
Multi-Reader Topology	1.807628 seconds	0.018076 seconds

The time required to evaluate the three different size networks scales less than linear in time. Evaluating the single size network using the multi-reader topological entity requires 4.7 ms (milliseconds), and the time required to evaluate the quadruple size network using the multi-reader topological entity requires 18.0 ms (milliseconds). If the execution time of the method scaled linearly with the size of the network, then the quadruple network should require 18.8 ms (milliseconds) for evaluation, but it only requires 18.0 ms (milliseconds). Hence, the execution time of the evaluation method scales less than linearly to the size of the network.

The percent difference, D, between the energy consumed by the network over one day calculated using the base level entity model, $E_{\text{Base-Level}}$, and the multi-reader topological entity model, $E_{\text{Multi-Reader}}$, is given by the following equation,

$$D = \frac{\left| E_{\text{Base-Level}} - E_{\text{Multi-Reader}} \right|}{E_{\text{Base-Level}}} \tag{10-109}$$

The energies consumed by the single, double, and quadruple size example networks as well as the percentage difference, D, between the base level and multi-reader topological entity models for each of the three cases are shown in Table 10.23.

Table 10.23: Energy consumption and percent differences between models for single, double, and quadruple size example networks.

Size	Entity	Energy Consumed (mJ)	Percent Difference (%)
Single	Base Level Entities	168213595.82 mJ	3.7758 %
	Multi-Reader Topological Entity	161862122.37 mJ	
Double	Base Level Entities	336647721.33 mJ	1.4771 %
	Multi-Reader Topological Entity	341620256.48 mJ	
Quadruple	Base Level Entities	673448327.46 mJ	3.9495 %
	Multi-Reader Topological Entity	700046334.95 mJ	

The percent difference between the model using the base level entities and the model using the multi-reader topological entity are within 4 % of each other in all three cases. Hence, the method developed in this work is considered accurate.

11.0 ZIGBEE NETWORK EXAMPLE

ZigBee is a network structure focusing on the connecting low-power wireless devices together to form a network. ZigBee was created by a group of companies looking to develop a standard for wireless low-power devices [63]. The ZigBee standard partitions the network into a number of layers like the Open Systems Interconnection (OSI) standard [54]. ZigBee networks use IEEE 802.15.4 for the lowest two layers (PHY and MAC), and the ZigBee standard defines the upper layers of the network [54, 64].

A ZigBee network is composed of three types of devices. The first, the ZigBee coordinator, is the ZigBee implementation of the IEEE 802.15.4 personal area network (PAN) coordinator [54]. There is only one PAN coordinator in an IEEE 802.15.4 network and the PAN coordinator is responsible for controlling the network [64]. The second type of device found in a ZigBee network is a ZigBee router which is an IEEE 802.15.4 Full-Function Device (FFD) that is not acting as the ZigBee coordinator [54, 64]. The ZigBee coordinator is simply a ZigBee router that has been selected to oversee the entire network. The final type of device in a ZigBee network is the ZigBee end-device. The ZigBee end-device is analogous to the IEEE 802.15.4 Reduced-Function Device (RFD) and usually contains only a sensor, minimal control logic, and a transceiver [54, 64]. A RFD can only communicate with a FFD, while a FFD can communicate with other FFDs or with RFDs [64].

Two categories of topologies are defined by IEEE 802.15.4 and ZigBee, star topology and the peer-to-peer topology [54, 64]. In the star topology, the PAN coordinator acts as the central entity and devices can communicate directly with the PAN coordinator [64]. An example of a ZigBee network arranged in a star topology with each device having a direct link to the PAN coordinator is shown in Figure 11.1.

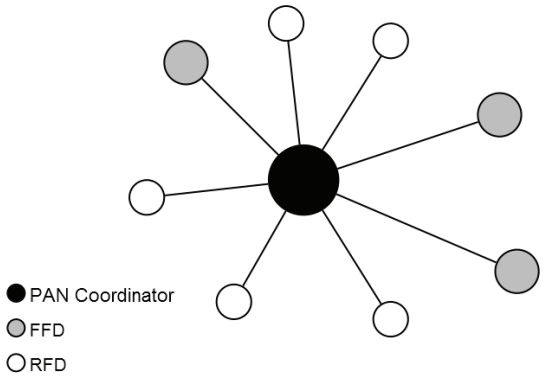

Figure 11.1: Example of a ZigBee network arranged in a star topology.

In a peer-to-peer network there is still a PAN coordinator, and devices do not need to have a direct connection to the PAN coordinator [64]. In a peer-to-peer topology, it is possible for a device to communicate with the PAN coordinator using multi-hop communication through a number of intermediate ZigBee routers (or FFDs in IEEE 802.15.4 terminology). An IEEE 802.15.4 peer-to-peer topology may contain a mesh, cluster, tree, single link topologies. The entire peer-to-peer network may contain multiple topologies. An example of a ZigBee network arranged using the peer-to-peer topology is shown in Figure 11.2. The RFDs (ZigBee end-devices) must communicate with the PAN controller through FFDs (ZigBee routers).

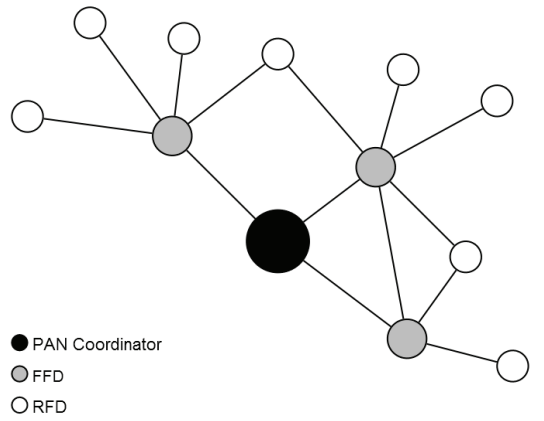

Figure 11.2: Example of a ZigBee network arranged using the Peer-to-Peer topology.

In this example case, the energy consumption of the following ZigBee network shown in Figure 11.4 will be investigated. This example ZigBee network is arranged as a binary tree of star topologies. The ZigBee coordinator is the root and communicates with two ZigBee routers. Each ZigBee router connects to five ZigBee end-devices forming two star topologies. The ZigBee end-devices are temperature sensors that periodically transmit their temperature reading to the ZigBee router. The ZigBee routers may request the ZigBee end-devices to report their temperature by sending a broadcast command to the ZigBee end-devices connected to them. The top-level network shown in Figure 11.3 contains a single entity, the ZigBee tree topological entity, representing the entities shown in Figure 11.4.

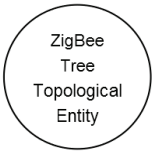

Figure 11.3: Top-level of the example ZigBee network, represented by a single ZigBee tree topological entity.

The intermediate-level network, shown in Figure 11.4, resembles a binary tree with the ZigBee coordinator (ZC) as the root and star topologies as the two children. This is an example of using the top-down methodology to construct a network. First, the ZigBee tree topological entity is deployed. Next the ZigBee tree topological entity is broken down into three entities the ZigBee coordinator and two ZigBee star topological entities. Finally, the intermediate-level depiction shown in Figure 11.4 will be reduced to the base entities as shown in Figure 11.5. The example ZigBee network showing the deployment of base entities is shown in Figure 11.5 with each base entity assigned an identifier (E denotes an end-device, C a coordinator, and R a router).

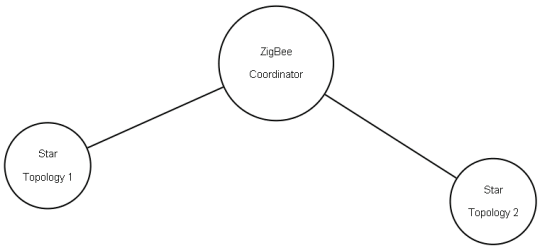

Figure 11.4: Intermediate-level depiction of the ZigBee network used in this example.

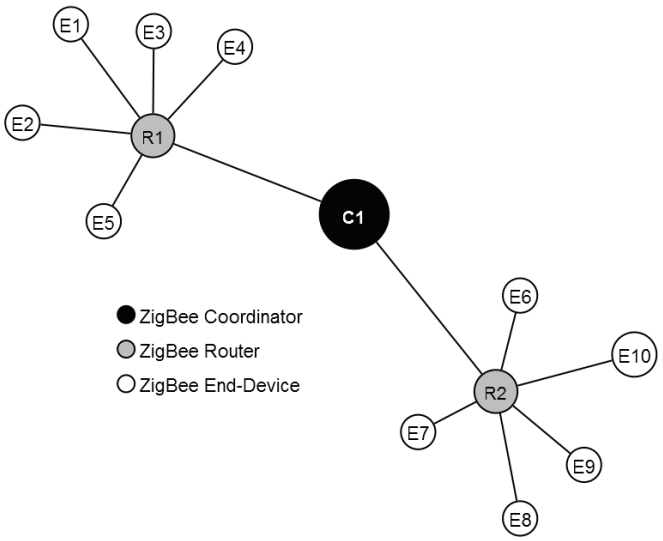

Figure 11.5: ZigBee network example case.

The ZigBee end-devices must perform two tasks, receive the take a reading command from a ZigBee router, and take a temperature reading and then send that reading to the ZigBee router periodically. The ZigBee router must perform three tasks. First, the ZigBee router must receive, store, and average the temperature readings from the ZigBee end-devices. Second, the ZigBee router must receive messages from the ZigBee coordinator and forward those requests to the end-devices. In this example, the ZigBee coordinator only sends the message for the ZigBee routers to request the ZigBee end-devices to take a temperature reading and report the results. The third task that the ZigBee router performs is to transmit the average temperature to the

ZigBee coordinator. The ZigBee coordinator must perform three tasks. The first task that the ZigBee coordinator performs is to receive the average temperatures from the ZigBee routers. Second, the ZigBee coordinator sometimes requires the temperature readings between periodic reporting intervals and sends a command to the ZigBee routers requesting the temperature. Third, the ZigBee controller transmits the temperature readings to the outside world for further use.

11.1 ANALYSIS OF ENERGY CONSUMPTION USING BASE LEVEL ENTITIES

The energy consumption of the ZigBee network described in the previous section will be evaluated by focusing on the energy consumption of each base level entity (ZigBee end-devices, ZigBee routers, and ZigBee coordinators). This example assumes that there are no collisions because the routers can assign the end-devices separate timeslots to transmit information, and that all messages arrive at the destination without error. Markov processes with probability and reward matrices will be developed to describe each of the three types of base entities and will then be evaluated to determine the energy consumed by the network over the period of one day. The components used in the ZigBee devices are listed in the following table, Table 11.1.

Table 11.1: Components used in the base entities in this example.

Component	Function
MC13192	2.4 GHz Low Power IEEE 802.15.4 Transciever
ATmega128(L)	Processor
LM70	Temperature Sensor

The ATmega128 process is assumed to operate at 4 MHz for the temperature sensors (ZigBee end-devices) and at 8 MHz for the ZigBee routers and ZigBee coordinator. The

temperature sensor, LM70, is found only in the ZigBee end-devices (temperature sensors). The transceiver is used in all three entities.

11.1.1 Step 1: Identification of Base Level Entities

The base level entities in a ZigBee network are identified in the ZigBee standard. There are three base level entities in a ZigBee network, the ZigBee coordinator, the ZigBee router, and the temperature sensor (ZigBee end-device). These three base level entities are described in detail in Section 11.0.

11.1.2 Step 2: Identification of Tasks of the Base Level Entities

The tasks of the three base level entities in the ZigBee example network are presented in this section. This is step 2 in the algorithm. These tasks will be used to help identify the two ZigBee star topological entities and the ZigBee tree topological entity during later steps in the algorithm.

The ZigBee end-devices must perform two tasks, receive the take a reading command from a ZigBee router, and to take a temperature reading and then send that reading to the ZigBee router periodically.

The ZigBee router must perform three tasks. First, the ZigBee router must receive, store, and average the temperature readings from the ZigBee end-devices. Second, the ZigBee router must receive messages from the ZigBee coordinator and forward those requests to the end-devices. In this example, the ZigBee coordinator only sends the message for the ZigBee routers to request the ZigBee end-devices to take a temperature reading and report the results. The third task that the ZigBee router performs is to transmit the average temperature to the ZigBee coordinator.

The ZigBee coordinator must also perform three tasks. The first task that the ZigBee coordinator performs is to receive the average temperatures from the ZigBee routers. Second, the ZigBee coordinator sometimes requires the temperature readings between periodic reporting intervals and sends a command to the ZigBee routers requesting the temperature. Third, the ZigBee controller transmits the temperature readings to the outside world for further use.

11.1.3 Step 3: Development of the Markov Process, Probability and Rewards Matrices for the Base Level Entities

The Markov processes and expressions for the probability and rewards matrices are presented for each of the three base level entities of the ZigBee example in this section. This is step 3 in the algorithm. The Markov processes and expressions for the probability and rewards matrices for the temperature sensor entity are presented in Section 11.1.4. The Markov processes and expressions for the probability and rewards matrices for the ZigBee router entity are presented in Section 11.1.5. The Markov processes and expressions for the probability and rewards matrices for the ZigBee coordinator entity are presented in Section 11.1.6. The energy consumption for each base level entity is calculated and presented in these sections. The energy consumption will be used to determine the energy consumed by the entire network using the base level entities and will be compared to the energy consumption found using the topological entities developed in later sections.

11.1.4 Step 3 for the Temperature Sensor

The ZigBee end-devices in this network consist entirely of temperature sensors. These temperature sensors periodically take a temperature reading and report that reading to the ZigBee router they communicate with. These readings are taken without any explicit command from the ZigBee routers. However, the network may need a temperature reading between the periodic intervals. In this case the ZigBee routers issue a command instructing the temperature sensors to take a temperature reading. Upon receiving this command, the temperature sensors take and transmit the temperature reading to the ZigBee router to which they are attached. The Markov process describing the temperature sensor ZigBee end-device is shown Figure 11.6.

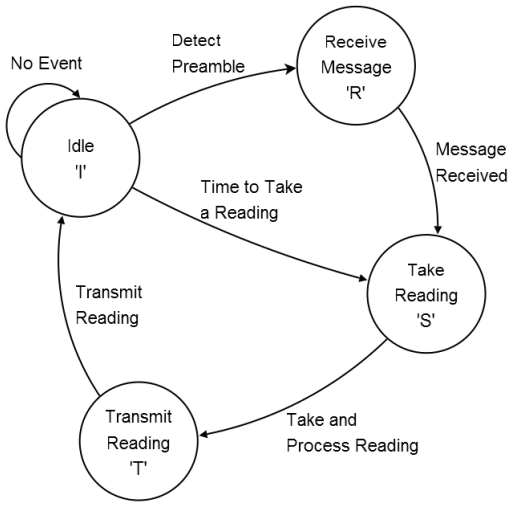

Figure 11.6: Markov process for the temperature sensor ZigBee end-device.

The probability and reward matrices are as follows:

$$P = \begin{bmatrix} P_{II} & P_{IR} & P_{IS} & 0 \\ 0 & 0 & P_{RS} & 0 \\ 0 & 0 & 0 & P_{ST} \\ P_{TI} & 0 & 0 & 0 \end{bmatrix} \tag{11-1}$$

$$R = \begin{bmatrix} R_{II} & R_{IR} & R_{IS} & 0 \\ 0 & 0 & R_{RS} & 0 \\ 0 & 0 & 0 & R_{ST} \\ R_{TI} & 0 & 0 & 0 \end{bmatrix} \tag{11-2}$$

The probabilities for those transitions originating in state I are found first. The time required to receive the whole ZigBee preamble is denoted as, $T_{Preamble}$. The preamble is defined as a 4 byte sequence in the ZigBee and IEEE 802.15.4 standard [54, 64]. The time required to check the device's counters to determine if it is time to take a periodic temperature reading also requires, $T_{Preamble}$, seconds. Hence, the period of one day can be broken up into, T, intervals of, $T_{Preamble}$, seconds.

$$T = \frac{(\text{Seconds in One Day})}{T_{\text{Preamble}}} \tag{11-3}$$

Each temperature sensor receives, M_{Router}, messages from the router requesting a temperature reading per day. Hence, the probability that the temperature sensor transitions into state R, to receive a message from the ZigBee router, P_{IR}, is given by the following equation.

$$P_{IR} = \frac{M_{\text{Router}}}{T} \tag{11-4}$$

The temperature sensor also automatically takes a temperature reading every, $T_{Reading}$, seconds. The number of readings that the temperature sensor takes automatically per day, T_{Auto}, is given by the following equation.

$$T_{\text{Auto}} = \frac{(\text{Seconds in One Day})}{T_{\text{Reading}}} \tag{11-5}$$

The probability that the temperature sensor must take a temperature reading automatically, P_{IS}, is given below.

$$P_{IS} = \frac{T_{\text{Auto}}}{T} \tag{11-6}$$

It is assumed that, $(T_{Auto} + M_{Router}) < T$, is always true. The probability that the temperature sensor remains idle, P_{II}, is simply,

$$P_{II} = 1 - (P_{IS} + P_{IR}) \tag{11-7}$$

The remaining probabilities are 1, thus,

$$P_{RS} = P_{ST} = P_{TI} = 1 \tag{11-8}$$

The reward value for each the transition is now found. In the idle state, state I, the temperature sensor is dormant. In state I, only the processor and receiver are active, while the transmitter and temperature sensor are dormant. The temperature sensor receives the message sent from the ZigBee router in state R. When in state R, only the processor and receiver must be active, no other components are required to be active. While the temperature sensor is taking a temperature reading, the processor and temperature sensor must be active, while the receiver and transmitter can be turned off. Finally, in the transmitting reply state, state T, only the transmitter and processor must be active, and the receiver must be turned on when leaving state T, so that the temperature sensor is ready to receive a message in state I.

The power consumed when the receiver and processor are active is denoted as, PW_{PR}. The power consumption of the other components in sleep mode is also included unless otherwise mentioned. The reward for the temperature sensor to remain in state I, R_{II}, is given by the following equation.

$$R_{II} = T_{\text{Preamble}} * PW_{PR} \tag{11-9}$$

When a ZigBee preamble is detected the temperature sensor does not need to activate or power down any additional components, hence the reward for transitioning from state I to state R, R_{IR}, is equal to, R_{II}.

$$R_{IR} = R_{II} = T_{\text{Preamble}} * PW_{PR} \tag{11-10}$$

When it is time for the temperature sensor to take an automatic temperature reading, the receiver can be powered down while the temperature sensor must be activated. The receiver can be powered down almost instantaneously, while the temperature sensor requires, $T_{TSWakeUp}$, to wake up. The power consumption of the temperature sensor, with only the processor and temperature sensor active, is denoted by, PW_{PS}. Thus, the reward for the I to S transition, R_{IS}, is,

$$R_{IS} = T_{\text{Preamble}} * PW_{PR} + T_{\text{TSWakeUp}} * PW_{PS} \tag{11-11}$$

The only message that the ZigBee router sends to the temperature sensors is the request for a reading command. Hence, all messages from the ZigBee router are the same length denoted by, L_{RMsg}. The ZigBee message consists of take a reading command requiring, $L_{ReadCmd}$, bytes. In this example, $L_{ReadCmd}$, is 1 byte. All messages received by the temperature sensor are within the same PAN so intra-PAN message formats and short addresses will be used. The overhead of the ZigBee data frame minus the preamble, $L_{Overhead}$, is 15 bytes [64]. Thus, the length of a ZigBee message, minus the preamble (as that is detected in the I to R transition), L_{RMsg}, is 16 bytes. The time to receive one byte of a ZigBee message is denoted as, $T_{Byte\text{-}RX}$. The energy consumed to receive the router's message and to wake-up the temperature sensor, R_{RS}, is given by the following equation.

$$R_{RS} = L_{RMsg} * T_{\text{Byte}-\text{RX}} * PW_{PR} + T_{\text{TSWakeUp}} * PW_{PS} \tag{11-12}$$

The time required to take and process a temperature reading is denoted as, T_{Temp}. The time required to process the reading includes the time to obtain the reading from the temperature sensor, $T_{R\text{-}Temp}$, and the time to process the reading, T_{Proc}. The time to process the reading, T_{Proc}, can be obtained by dividing the number of instructions, N_i, required to process the reading by the number of instructions per second the processor can perform, C_i. Hence, T_{Proc}, is,

$$T_{\text{Proc}} = \frac{N_i}{C_i} \tag{11-13}$$

T_{Temp}, is then simply,

$$T_{\text{Temp}} = T_{\text{Proc}} + T_{\text{R}-\text{Temp}} \tag{11-14}$$

The transmitter must also be wakened, and this requires time, $T_{Wake\text{-}TX}$. The power consumption with the processor and transmitter active and the other components dormant is denoted by, PW_{PT}. Hence, the energy consumed to take and process the temperature reading, R_{ST}, is,

$$R_{ST} = T_{\text{Wake-TX}} * PW_{PT} + T_{\text{Temp}} * PW_{PS} \quad (11\text{-}15)$$

The length of the preamble, L_{Preamble}, is 4 bytes. The length of the data message, excluding the preamble, containing the temperature reading is denoted as, L_{TMsg}. The message contains, L_{TempOver}, bytes of overhead. The overhead, L_{TempOver}, for a data message is 15 bytes (excluding the 4 byte preamble sequence) [64]. The data portion of the message consists of the temperature reading, requiring, L_{TData}, bytes. In this example, L_{TData}, is assumed to be 4 bytes. Hence, L_{TMsg}, is,

$$L_{\text{TMsg}} = L_{\text{TempOver}} + L_{\text{TData}} \quad (11\text{-}16)$$

The time to transmit one byte of the message in ZigBee is denoted as, $T_{\text{Byte-TX}}$, and the time for the receiver to wake-up is denoted as, $T_{\text{Wake-RX}}$. It is assumed that the transmitter can be powered down almost instantaneously. Hence, the energy consumed by the temperature sensor to transmit a temperature reading, R_{TI}, is,

$$R_{TI} = T_{\text{Wake-RX}} * PW_{PR} + \left(L_{\text{TMsg}} + L_{\text{Preamble}}\right) * T_{\text{Byte-TX}} * PW_{PT} \quad (11\text{-}17)$$

The LM70 temperature sensor operates at 2.65 V, drawing 12 μA (mico-Amps) when shutdown and 260 μA (micro-Amps) when active [65]. The LM70 requires 210 ms (milli-seconds) to take a reading and can wake-up time is included in the time to take a reading [65]. The ATmega128L operates at 3 V when the clock frequency is 4 MHz and draws 5.5 mA (milli-Amps) of current [61]. The Freescale transceiver operates at 2.7 V and draws 37 mA (milli-Amps) when receiving; 30 mA (milli-Amps) when transmitting; and 500 μA (micro-Amps) when idle [66]. The transceiver requires 144 μs (micro-seconds) to wake-up the transmitter or receiver [66]. The parameters used to evaluate the energy consumption of the temperature sensor ZigBee end-devices are listed in Table 11.2. The energy consumption of each of the temperature sensors over one day in the example network using the parameters in Table 11.2 are listed in Table 11.3.

Table 11.2: Parameters for evaluation for the energy consumption of the temperature sensor.

Parameter	Value
Milliseconds in One Day	86,400,000 milliseconds / day
ZigBee Operational Frequency	2.4 GHz
$T_{Byte-TX}$	0.032 ms
$T_{Preamble}$	0.128 ms
M_{Router} – Temperature sensors connected to R1	48
M_{Router} – Temperature sensors connected to R2	48
$T_{Reading}$	180 seconds
PW_{PR}	116.4318 mW
PW_{PS}	18.5390 mW
$T_{TSWakeUp}$	0 ms
L_{RMsg}	16 Bytes
$T_{Byte-RX}$	0.032 ms
N_i	1000 instructions
C_i	4 MHz
T_{Proc}	0.25 ms
T_{R-Temp}	210 ms
T_{Temp}	210.25 ms
$T_{Wake-TX}$	0.144 ms
PW_{PT}	97.5318 mW
$L_{TempOver}$	15 Bytes
L_{TData}	4 Bytes
L_{TMsg}	19 Bytes
$T_{Wake-RX}$	0.144 ms
T	2,700,000,000
$L_{Preamble}$	4 Bytes

Table 11.3: Energy consumption of the temperature sensors in the example network over 1 day.

Temperature Sensor	Energy Consumed (mJ)	Temperature Sensor	Energy Consumed (mJ)
E1	10048845.49 mJ	E6	10048845.49 mJ
E2	10048845.49 mJ	E7	10048845.49 mJ
E3	10048845.49 mJ	E8	10048845.49 mJ
E4	10048845.49 mJ	E9	10048845.49 mJ
E5	10048845.49 mJ	E10	10048845.49 mJ

11.1.5 Step 3 for the ZigBee Router

The two ZigBee routers in the example network must collect temperature readings from the temperature sensors with which they communicate. After, all the temperature readings have been received, the ZigBee router computes the average temperature and sends it to the ZigBee coordinator. The ZigBee router must also listen for the message from the ZigBee coordinator to take a temperature reading between reporting intervals. Upon receiving this message from the ZigBee coordinator, the ZigBee router must broadcast the request to the temperature sensors that it is responsible for causing the temperature sensors to take and transmit their temperature readings to the ZigBee router. The Markov process describing the ZigBee router is shown in Figure 11.7. The probability and reward matrices have the following form.

$$P = \begin{bmatrix} P_{II} & P_{IR} & 0 & 0 & 0 & 0 \\ 0 & 0 & P_{RB} & P_{RP} & 0 & 0 \\ P_{BI} & 0 & 0 & 0 & 0 & 0 \\ P_{PI} & 0 & 0 & 0 & P_{PA} & 0 \\ 0 & 0 & 0 & 0 & 0 & P_{AT} \\ P_{TI} & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{11-18}$$

$$R = \begin{bmatrix} R_{II} & R_{IR} & 0 & 0 & 0 & 0 \\ 0 & 0 & R_{RB} & R_{RP} & 0 & 0 \\ R_{BI} & 0 & 0 & 0 & 0 & 0 \\ R_{PI} & 0 & 0 & 0 & R_{PA} & 0 \\ 0 & 0 & 0 & 0 & 0 & R_{AT} \\ R_{TI} & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{11-19}$$

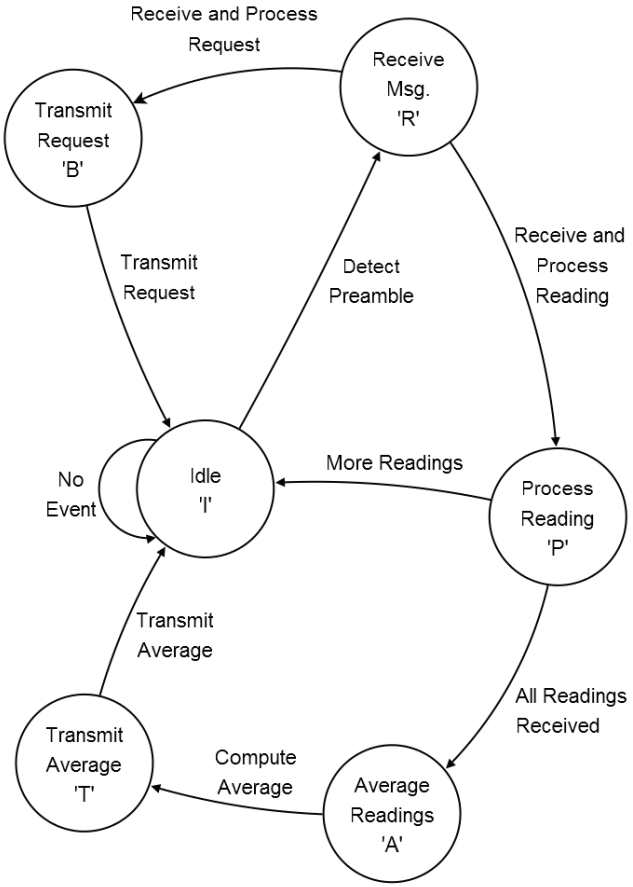

Figure 11.7: Markov process for a ZigBee router.

The probability for those transitions originating in state I are found first. The time required to receive the whole ZigBee preamble is denoted as, $T_{Preamble}$. The preamble is defined as a 4 byte sequence in the ZigBee and IEEE 802.15.4 standard [54, 64]. The time required to check the device's counters to determine if it is time to take a periodic temperature reading also

requires, $T_{Preamble}$, seconds. Hence, the period of one day can be broken up into, T, intervals of, $T_{Preamble}$, seconds.

$$T = \frac{(\text{Seconds in One Day})}{T_{\text{Preamble}}} \tag{11-20}$$

Each router communicates with, N_T, temperature sensor end-devices. Each temperature sensor transmits temperatures readings automatically at a frequency of, $F_{Reading}$, resulting in, $M_{Reading}$, temperature readings per day from one temperature sensor.

$$M_{\text{Reading}} = (\text{Seconds in One Day}) * F_{\text{Reading}} \tag{11-21}$$

Each router receives, M_{Coord}, messages from the coordinator requesting a temperature reading between reporting intervals per day. Each of the, M_{Coord}, messages causes additional readings to be received by the router. The number of readings received by the router due to, M_{Coord}, messages, $M_{Read\text{-}C}$, is,

$$M_{\text{Read-C}} = \text{M}_{\text{Coord}} * N_T \tag{11-22}$$

The total number of messages received by the router is denoted as, M_{Total}.

$$M_{\text{Total}} = N_T * M_{\text{Reading}} + M_{\text{Read-C}} + M_{\text{Coord}} \tag{11-23}$$

Hence, the probability that the router transitions into state R, to receive a message from the ZigBee coordinator, P_{IR}, is given by the following equation.

$$P_{IR} = \frac{M_{\text{Total}}}{T} \tag{11-24}$$

It is assumed that, $M_{Total} < T$; otherwise if, $M_{Total} > T$, then such a network could not exist because there are more messages than time permits. The probability that a preamble is not detected, P_{II}, is simply,

$$P_{II} = 1 - P_{IR} \tag{11-25}$$

After the preamble is detected the message may be either a temperature reading, or a request from the ZigBee coordinator to take a reading. The probability that message is a temperature reading from one of the temperature sensors, P_{RP}, is,

$$P_{RP} = \frac{N_T * M_{\text{Reading}} + M_{\text{Read-C}}}{M_{\text{Total}}} \tag{11-26}$$

The probability that the message is a request for the temperature from the ZigBee coordinator is,

$$P_{RB} = 1 - P_{RP} \tag{11-27}$$

The probability that the last reading has been received, P_{PA}, is simply,

$$P_{PA} = \frac{1}{N_T} \tag{11-28}$$

The probability that there are more readings left, P_{PI}, is simply,

$$P_{PI} = 1 - P_{PA} \tag{11-29}$$

The remaining transition probabilities are all 1.

$$P_{BI} = P_{AT} = P_{TI} = 1 \tag{11-30}$$

The reward value for each transition denotes the energy consumed for the task represented by the transition. The router, in state I, is waiting for a preamble indicating a message is present. To monitor for the preamble, the processor and receiver must be active, and all other components (transmitter) are dormant, but the power consumed in the dormant state is included unless otherwise mentioned. While receiving a message or while processing a reading only the receiver and processor must be active. When the average of the temperature readings is being calculated only the processor must be active. When transmitting the average temperature or broadcasting the temperature reading request command, only the processor and transmitter must be active, the receiver can be dormant.

The power consumed when the receiver and processor are active is denoted as, PW_{PR}. The reward for the router to remain in state I, R_{II}, is given by the following equation.

$$R_{II} = T_{\text{Preamble}} * PW_{PR} \tag{11-31}$$

When a ZigBee preamble is detected, the router does not need to activate or power down any additional components, hence the reward for transitioning from state I to state R, R_{IR}, is equal to, R_{II}.

$$R_{IR} = R_{II} = T_{\text{Preamble}} * PW_{PR} \tag{11-32}$$

The router can receive two messages, the request for a reading from the ZigBee coordinator or the temperature reading from a single temperature sensor. The only message that the ZigBee coordinator sends to the router is to request a reading. Hence, all messages from the ZigBee coordinator are the same length denoted by, L_{CMsg}. The ZigBee message consists of the take a reading command, requiring, $L_{ReadCmd}$, bytes. In this example, $L_{ReadCmd}$, is 1 Byte. All messages in this example are within the same PAN so intra-PAN message formats and short addresses will be used. The overhead of the ZigBee data frame, $L_{Overhead}$, is 15 bytes (preamble is excluded) [64]. Thus, the length of a ZigBee message, minus the preamble (as that is detected in the I to R transition), L_{CMsg}, is 16 bytes. The length of the preamble, $L_{Preamble}$, is 4 bytes. The time to receive one byte of a ZigBee message is denoted as, $T_{Byte\text{-}RX}$.

The time required to process a request for a temperature reading is denoted as, $T_{P\text{-}Req}$. The time to process the reading, $T_{P\text{-}Req}$, can be obtained by dividing the number of instructions, $N_{i\text{-}Req}$, required to process the reading by the number of instructions per second the processor can perform, $C_{i\text{-}Req}$. Hence, $T_{P\text{-}Req}$, is,

$$T_{\text{P-Req}} = \frac{N_{i\text{-Req}}}{C_{i\text{-Req}}} \tag{11-33}$$

The power consumed when only the processor is active is denoted as, PW_P. Hence, the energy consumed to receive the request for a temperature reading from the ZigBee coordinator, R_{RB}, is,

$$R_{RB} = L_{CMsg} * T_{\text{Byte-RX}} * PW_{PR} + T_{\text{P-Req}} * PW_P \tag{11-34}$$

After receiving the request for a reading from the ZigBee coordinator, the router must broadcast that request to the temperature sensors for which it is responsible. The transmitter must also be wakened and this requires that time, $T_{Wake\text{-}TX}$. The power consumption with the processor and transmitter active and the other components dormant is denoted by, PW_{PT}. The time for the receiver to wake-up is denoted as, $T_{Wake\text{-}RX}$. It is assumed that the transmitter and receiver can be powered down almost instantaneously. The energy consumed to broadcast the reading request, R_{BI}, is,

$$R_{BI} = T_{\text{Wake-RX}} * PW_{PR} + \left(T_{\text{Wake-TX}} + \left(L_{\text{CMsg}} + L_{\text{Preamble}}\right) * T_{\text{Byte-TX}}\right) * PW_{PT} \tag{11-35}$$

The length of the data message containing the temperature reading is denoted as, L_{TMsg}. The message contains, $L_{TempOver}$, bytes of overhead. The overhead, $L_{TempOver}$, for a data message is 15 bytes (excluding the 4 byte preamble sequence) [64]. The data portion of the message consists of the temperature reading, requiring, L_{TData}, bytes. In this example, L_{TData}, is assumed to be 4 bytes. Hence, L_{TMsg}, is,

$$L_{\text{TMsg}} = L_{\text{TempOver}} + L_{\text{TData}} \tag{11-36}$$

The energy consumed to receive a temperature reading from a temperature sensor, R_{RP}, is,

$$R_{RP} = L_{TMsg} * T_{\text{Byte-RX}} * PW_{PR} \tag{11-37}$$

The time required to process a temperature reading and determine if all temperature readings have been received or if more are expected is denoted as, T_{Proc}. The time to process the reading, T_{Proc}, can be obtained by dividing the number of instructions, N_i, required to process the reading by the number of instructions per second the processor can perform, C_i, (this is simply the clock frequency). Hence, T_{Proc}, is,

$$T_{\text{Proc}} = \frac{N_i}{C_i} \tag{11-38}$$

Therefore, the energy consumed to determine if there are more temperature readings to receive, R_{PI}, is,

$$R_{PI} = T_{\text{Proc}} * PW_{PR} \tag{11-39}$$

The same processing must be done on the message to determine if that received reading is the last reading. Thus, the energy consumed to determine that a reading is the last reading, R_{PA}, is equal to, R_{PI}.

$$R_{PA} = R_{PI} \tag{11-40}$$

The time required to compute the average of the temperature readings, T_{Avg}, can be found using (11-38) with the appropriate value for, N_i, and C_i, for calculating the average temperature. Therefore, the energy consumed to determine if there are more temperature readings to receive, R_{AT}, is,

$$R_{AT} = T_{\text{Avg}} * PW_P \tag{11-41}$$

The length of the message containing the average temperature is the same as that containing a temperature reading, L_{TMsg}, because the size of the temperature data field is kept the same so as not to lose any precision. The time to transmit one byte of the message in ZigBee is denoted as, $T_{Byte\text{-}TX}$, and the time for the receiver to wake-up is denoted as, $T_{Wake\text{-}RX}$. It is assumed that the transmitter can be powered down almost instantaneously.

$$R_{TI} = T_{\text{Wake-RX}} * PW_{PR} + \left(\left(L_{\text{TMsg}} + L_{\text{Preamble}}\right) * T_{\text{Byte-TX}} + T_{\text{Wake-TX}}\right) * PW_{PT} \tag{11-42}$$

The Atmega128 processor with a clock speed of 8 MHz operates at 5 V and draws 19 mA (milli-Amps) of current [61]. The parameters used to evaluate the energy consumption of the temperature sensor ZigBee end-devices are listed in Table 11.4. The energy consumption of each of the routers over one day in the example network using the parameters in Table 11.4 are listed in Table 11.5.

Table 11.4: Parameters for evaluation for the energy consumption of the router.

Parameter	Value
Milliseconds in One Day	86,400,000 milliseconds / day
ZigBee Operational Frequency	2.4 GHz
$T_{Byte-TX}$	0.032 ms
$T_{Preamble}$	0.128 ms
$T_{Reading}$	180 seconds
PW_{PR}	194.9 mW
L_{RMsg}	16 Bytes
$T_{Byte-RX}$	0.032 ms
N_i	1500 instructions
C_i	8 MHz
T_{Proc}	0.25 ms
$T_{Wake-TX}$	0.144 ms
PW_{PT}	176 mW
$L_{TempOver}$	15 Bytes
L_{TData}	4 Bytes
L_{TMsg}	19 Bytes
$T_{Wake-RX}$	0.144 ms
M_{Coord}	48
$F_{Reading}$	0.0056 Readings per second
N_T	5
M_{Read-C}	240
M_{Total}	720
T_{AVG}	0.1875 ms
PW_P	95 mW
$M_{Reading}$	480 readings per day
T	2,700,000,000
$L_{Preamble}$	4 Bytes
N_{i-Req}	1000
C_{i-Req}	8 MHz
T_{P-Req}	0.1250 ms

Table 11.5: Energy consumption of the routers in the example network over 1 day.

Router	Energy Consumption (mJ)
R1	16839340.02 mJ
R2	16839340.02 mJ

11.1.6 Step 3 for the ZigBee Coordinator

The ZigBee coordinator is responsible for controlling the PAN. The coordinator receives temperature readings from the attached routers, and forwards those readings to the outside world. The coordinator also receives requests for temperature readings from the outside world causing the coordinator to request the temperature readings from the routers it is responsible for. The Markov process for the ZigBee coordinator is shown in Figure 11.8.

The coordinator in the idle state, I, is waiting for an incoming message. The coordinator upon detecting the preamble will transition into state R to receive the message. The message may be a temperature reading or a request for the temperature reading from the outside world. If the message is a temperature reading, the coordinator will transition into state P, and if the message is a request for the temperature the coordinator will transition into state D. The coordinator processes the received temperature request message or temperature reading message in states D and P respectively. The coordinator will forward the temperature reading to the outside world in state W and then return to state I. Similarly, if the message is a request for the temperature, the coordinator will broadcast the temperature request to the ZigBee routers. This will cause the routers to request and obtain the temperature readings from the temperature sensors and then transmit the average temperature to the coordinator. The probability and reward matrices have the following form.

$$P = \begin{bmatrix} P_{II} & P_{IR} & 0 & 0 & 0 & 0 \\ 0 & 0 & P_{RP} & 0 & 0 & P_{RD} \\ 0 & 0 & 0 & P_{PW} & 0 & 0 \\ P_{WI} & 0 & 0 & 0 & 0 & 0 \\ P_{BI} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & P_{DB} & 0 \end{bmatrix} \tag{11-43}$$

$$R = \begin{bmatrix} R_{II} & R_{IR} & 0 & 0 & 0 & 0 \\ 0 & 0 & R_{RP} & 0 & 0 & R_{RD} \\ 0 & 0 & 0 & R_{PW} & 0 & 0 \\ R_{WI} & 0 & 0 & 0 & 0 & 0 \\ R_{BI} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & R_{DB} & 0 \end{bmatrix} \tag{11-44}$$

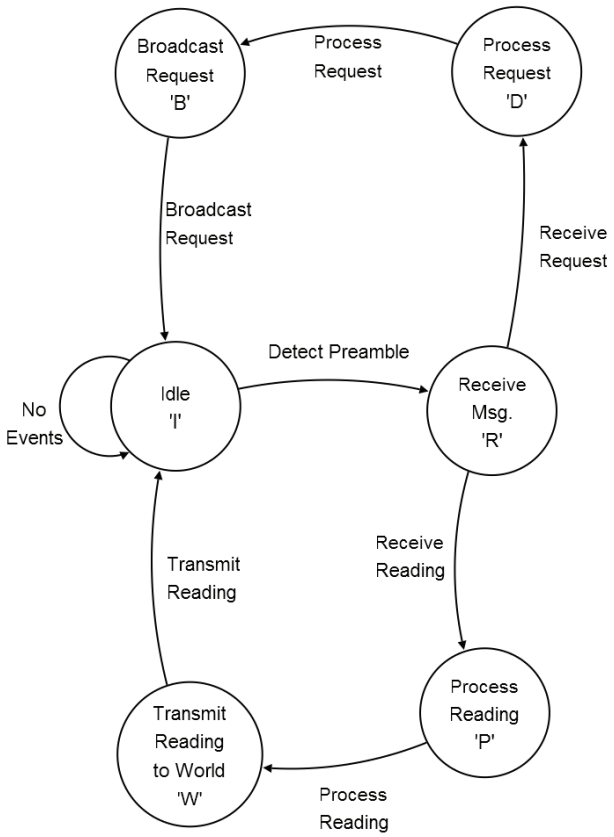

Figure 11.8: Markov process for the ZigBee coordinator.

The probability for those transitions originating in state I are found first. The time required to receive the whole ZigBee preamble is denoted as, $T_{Preamble}$. The preamble is defined as a 4 byte sequence in the ZigBee and IEEE 802.15.4 standards [54, 64]. The time required to check the device's counters to determine if it is time to take a periodic temperature reading also requires, $T_{Preamble}$, seconds. Hence, the period of one day can be broken up into, T, intervals of, $T_{Preamble}$, seconds.

$$T = \frac{(\text{Seconds in One Day})}{T_{\text{Preamble}}} \tag{11-45}$$

Each coordinator communicates with, N_R, ZigBee routers. Each router transmits the average temperature in response to automatic readings by the temperature sensors at a frequency of, $F_{Reading}$, resulting in, $M_{Reading}$, average temperature readings per day from one router.

$$M_{\text{Reading}} = (\text{Seconds in One Day}) * F_{\text{Reading}} \tag{11-46}$$

Each coordinator receives, M_{OUT}, messages from the outside world requesting a temperature reading between reporting intervals per day. Each of the, M_{OUT}, messages causes one additional average temperature reading to be received by the coordinator from each router that the coordinator is attached to. Thus, the number of readings received by the coordinator due to the, M_{OUT}, messages, $M_{Read\text{-}O}$, is,

$$M_{\text{Read-O}} = \text{M}_{\text{OUT}} * N_R \tag{11-47}$$

The total number of messages received by the router is denoted as, M_{Total}, is,

$$M_{\text{Total}} = \sum_{x=1}^{N_R} M_{\text{Reading}_x} + M_{\text{Read}-\text{O}} + M_{\text{OUT}} \tag{11-48}$$

The number of periodic reports from each router connected to the coordinator is, $M_{Readingx}$, and there are, N_R, routers connected to the coordinator. Hence, the probability that the coordinator transitions into state R, to receive a message from the ZigBee router or the outside world, P_{IR}, is given by the following equation.

$$P_{IR} = \frac{M_{\text{Total}}}{T} \tag{11-49}$$

The probability that a preamble is not detected and the coordinator stays in the idle state, P_{II}, is simply,

$$P_{II} = 1 - P_{IR} \tag{11-50}$$

The probability that the message is from the a temperature reading from the router is,

$$P_{RP} = \frac{\sum_{x=1}^{N_R} M_{\text{Reading}_x} + M_{\text{Read-O}}}{M_{\text{Total}}} \tag{11-51}$$

The probability that the message is a request for the temperature from the ZigBee coordinator is,

$$P_{RD} = 1 - P_{RP} \tag{11-52}$$

The remaining probabilities are all 1, therefore,

$$P_{DB} = P_{BI} = P_{PW} = P_{WI} = 1 \tag{11-53}$$

The reward values represent the energy consumption of coordinator for the task represented by each transition in the Markov process for the coordinator shown in Figure 11.8. The coordinator in the idle state is simply waiting for a preamble indicating the start of an incoming message. Only the processor and receiver must be active in this state. While receiving a message in state R, only the processor and receiver must be active. When processing either message, the receiver can be powered down. Thus, only the processor is active in states D and P. When transmitting the average temperature to the outside world, state W, or broadcasting the request, state B, only the transmitter and processor are active.

The power consumed when the receiver and processor are active is denoted as, PW_{PR}. The reward for the coordinator to remain in state I, R_{II}, is given by the following equation.

$$R_{II} = T_{\text{Preamble}} * PW_{PR} \tag{11-54}$$

When a ZigBee preamble is detected the coordinator does not need to activate or power down any additional components. Hence, the reward for transitioning from state I to state R, R_{IR}, is equal to, R_{II}.

$$R_{IR} = R_{II} = T_{\text{Preamble}} * PW_{PR} \tag{11-55}$$

The coordinator can receive two messages, the request for a reading from the outside, or the temperature reading from a router. A request for a temperature reading originates outside the PAN managed by the coordinator. Hence, the inter-PAN message format and long addresses are used. The command to request the temperature, L_{ReadCmd}, is 1 byte. The message overhead is, $L_{\text{Overhead-O}}$, is 29 bytes (preamble is excluded). The length of the entire message, L_{OUT}, is 30 bytes.

$$L_{\text{OUT}} = L_{\text{Overhead}-\text{O}} + L_{\text{ReadCmd}} \tag{11-56}$$

The time to receive or transmit one byte of a ZigBee message is denoted as, $T_{\text{Byte-RX}}$. Hence, the energy consumption to receive a request for the temperature from the outside world, R_{RD}, is,

$$R_{RD} = L_{OUT} * T_{\text{Byte}-\text{RX}} * PW_{PR} \tag{11-57}$$

The temperature reading is 4 bytes. Hence the length of the average temperature reading message received from a router, $L_{\text{Avg-Temp}}$, is 19 bytes (preamble is excluded). Therefore, the energy consumed to receive an average temperature reading from the router, R_{RP}, is,

$$R_{RP} = L_{\text{Avg}-\text{Temp}} * T_{\text{Byte}-\text{RX}} * PW_{PR} \tag{11-58}$$

The coordinator must process both the reading and the request for a temperature reading messages. The time required process a temperature reading is denoted as, $T_{\text{P-Read}}$, and the time required to process the request for a temperature reading is denoted as, $T_{\text{P-Req}}$. The time to

process the reading, $T_{P\text{-}Read}$, can be obtained by dividing the number of instructions, $N_{i\text{-}Read}$, required to process the reading by the number of instructions per second the processor can perform, $C_{i\text{-}Read}$. Hence, $T_{P\text{-}Read}$, is,

$$T_{\text{P-Read}} = \frac{N_{\text{i-Read}}}{C_{\text{i-Read}}} \tag{11-59}$$

Similarly, the time to process the reading, $T_{P\text{-}Req}$, can be obtained by dividing the number of instructions, $N_{i\text{-}Req}$, required to process the reading by the number of instructions per second the processor can execute, $C_{i\text{-}Req}$. Hence, $T_{P\text{-}Req}$, is,

$$T_{\text{P-Req}} = \frac{N_{\text{i-Req}}}{C_{\text{i-Req}}} \tag{11-60}$$

Since, only the processor needs to be active, the power consumption of the coordinator with only the processor active is denoted by, PW_P. Therefore, the energy consumed to process a temperature reading, R_{PW}, is,

$$R_{PW} = T_{\text{P-Read}} * PW_P \tag{11-61}$$

Similarly, the energy consumed by processing the request for a temperature reading message, R_{DB}, is,

$$R_{DB} = T_{\text{P-Req}} * PW_P \tag{11-62}$$

After receiving the request for a reading from the outside world, the coordinator must broadcast that request to the routers to which it is connected. The only message that the ZigBee coordinator sends to the routers is the request a temperature reading command. Hence, all messages sent by the ZigBee coordinator are the same length, denoted by, L_{RMsg}. The ZigBee message consists of take a reading command requiring, $L_{ReadCmd}$, bytes. In this example, $L_{ReadCmd}$, is 1 byte. This message is sent within the same PAN so the intra-PAN message format

and short addresses will be used. The overhead of the ZigBee data frame, $L_{Overhead}$, is 15 bytes (excluding the preamble) [64]. Thus, the length of a ZigBee message, minus the preamble, L_{RMsg}, is 16 bytes. The length of the preamble, $L_{Preamble}$, is 4 bytes. The transmitter must also be woken up, requiring time, $T_{Wake\text{-}TX}$. The power consumption with the processor and transmitter active and the other components dormant is denoted by, PW_{PT}. The time for the receiver to wake-up is denoted as, $T_{Wake\text{-}RX}$. It is assumed that the transmitter and receiver can be powered down almost instantaneously. The energy consumed to broadcast the reading request, R_{BI}, is,

$$R_{BI} = T_{\text{Wake-RX}} * PW_{PR} + \left(\left(L_{\text{RMsg}} + L_{\text{Preamble}}\right) * T_{\text{Byte-TX}} + T_{\text{Wake-TX}}\right) * PW_{PT} \tag{11-63}$$

Just like the request for a temperature reading message, the average temperature reading that is sent to the outside world is sent to another PAN so the inter-PAN format and long addresses are used. The average temperature reading, $L_{Temp\text{-}Avg}$, is 4 bytes. The message overhead is, $L_{Overhead\text{-}O}$, is 29 bytes (preamble is excluded). The length of the entire message, $L_{Temp\text{-}TX}$, is 34 bytes.

$$L_{\text{Temp-TX}} = L_{\text{Overhead-O}} + L_{\text{Temp-Avg}} + L_{\text{Preamble}} \tag{11-64}$$

Hence, the energy consumed to transmit the average temperature to the outside world and to activate the receiver, R_{WI}, is,

$$R_{WI} = T_{\text{Wake-RX}} * PW_{PR} + \left(L_{\text{Temp-TX}} * T_{\text{Byte-TX}} + T_{\text{Wake-TX}}\right) * PW_{PT} \tag{11-65}$$

The parameters used to evaluate the energy consumption of the temperature sensor ZigBee end-devices are listed in Table 11.6. The energy consumption of each of the coordinator over one day in the example network using the parameters in Table 11.6 are listed in Table 11.7.

Table 11.6: Parameters for evaluation for the energy consumption of the coordinator.

Parameter	Value
Milliseconds in One Day	86,400,000 milliseconds / day
ZigBee Operational Frequency	2.4 GHz
$T_{Byte-TX}$	0.032 ms
$T_{Preamble-Detect}$	0.128 ms
$T_{Reading}$	180 seconds
PW_{PR}	194.9 mW
L_{RMsg}	16 Bytes
$T_{Byte-RX}$	0.032 ms
N_{i-Read}	1000 instructions
C_{i-Read}	8 MHz
T_{P-Read}	0.1250 ms
$T_{Wake-TX}$	0.144 ms
PW_{PT}	176 mW
$L_{TempOver}$	15 Bytes
L_{TData}	4 Bytes
$L_{Avg-Temp}$	19 Bytes
$T_{Wake-RX}$	0.144 ms
M_{OUT}	48
$F_{Reading}$	0.0056 Readings per second
N_T	5
M_{Read-O}	240
M_{Total}	720
T_{AVG}	0.1875 ms
PW_P	95 mW
$\sum_{x=1}^{N_R} M_{Reading_x}$	960
N_R	2
T	2,700,000,000
L_{OUT}	30 Bytes
$L_{Overhead-O}$	29 Bytes
$L_{ReadCmd}$	1 Byte
N_{i-Req}	5000 instructions
C_{i-Req}	8 MHz
T_{P-Req}	0.6250 ms
$L_{Temp-Avg}$	4 Bytes
$L_{Temp-TX}$	37 Bytes
$L_{Preamble}$	4 Bytes

Table 11.7: Energy consumption of the coordinator in the example network over 1 day.

Router	Energy Consumption (mJ)
C1	16839316.60 mJ

The total energy consumed by the network over one day is obtained by summing the energy consumption of all 10 temperature sensors, the 2 routers, and the coordinator. The energy consumption of the entire network over 1 day is, 151006451.54 mJ.

11.1.7 Step 4: Identification of Interactions between Base Level Entities

The interactions between the base level entities is determined in this section and will be used in the following section to identify the ZigBee star topological entities in the network. This is step 4 in the algorithm. Each temperature sensor communicates with exactly one ZigBee router. The ZigBee routers communicate with the ZigBee coordinator. Hence, all messages for the temperature sensors from the ZigBee coordinator must go through one of the ZigBee routers. Likewise, any message for the temperature sensor from the ZigBee coordinator must travel through a ZigBee router. Further, the temperature sensors must report their temperature reading to the router periodically or upon request.

11.1.8 Step 5: Identification of the ZigBee Star Topological Entity

In step 5 of the algorithm, the topological entities are identified based on the tasks and interactions of the base level entities. The two ZigBee star topological entities are identified based on the tasks defined in Section 11.1.2 and the interactions identified in Section 11.1.7. The ZigBee routers and attached temperature sensors can be thought of as a single unit because they work together to perform the tasks in the network. Thus, the two stars constructed around the two routers in the example network can be represented by a single star topological entity. There are a total of twelve base-level entities in the two star networks. By representing each star

network with a single star topological entity, the number of entities that must be evaluated for those two star networks is reduced from twelve to two. The first of the star topologies, TOP_{S1}, is constructed around router, R1, and is shown in Figure 11.9. The second of the star topologies, TOP_{S2}, is constructed around router, R2, and is shown in Figure 11.10.

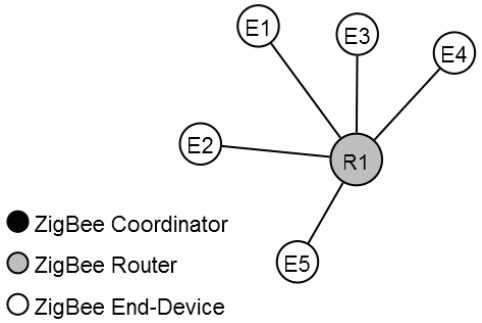

Figure 11.9: ZigBee base entities contained in topological entity TOP_{S1}.

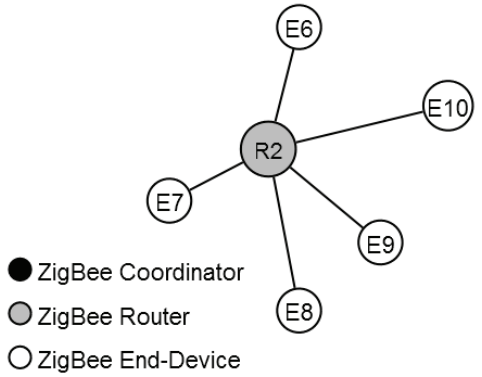

Figure 11.10: ZigBee base entities contained in topological entity TOP_{S2}.

With the ZigBee star fundamental topological entity defined, existing algorithms can be used to search the basic communication graph for the ZigBee star topological entities. These algorithms form the basis for CAD tools for integrated circuit design [46]. The two ZigBee star topological entities (TOP_{S1} and TOP_{S2}) identified in this example are shown in Figure 11.9 and

Figure 11.10, above. Employing the two ZigBee star topological entities the network can be covered using three entities, the ZigBee coordinator and the two ZigBee star topological entities (TOP_{S1} and TOP_{S2}) and this is shown in Figure 11.11. Thus, the order (number of Markov processes) of the ZigBee example network is reduced from 13 as shown in Figure 11.5 to 3 as shown in Figure 11.11.

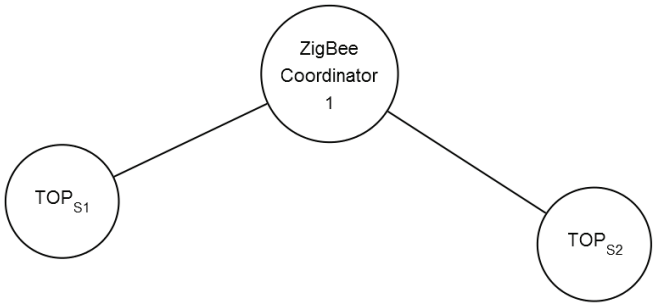

Figure 11.11: Covering of the ZigBee example network using the two ZigBee star topological entities (TOP_{S1} and TOP_{S2}).

11.1.9 Step 6: Identification of the Tasks of the ZigBee Star Topological Entity

The tasks of the ZigBee star topological entity are identified in this section. This is step 6 of the algorithm. These tasks will be used to help identify the ZigBee tree topological entity. The ZigBee star topological entity must take periodic temperature readings and provide the average temperature reading to the ZigBee coordinator. The ZigBee star topology must also take temperature readings in response to requests from the ZigBee coordinator for such readings. Again, the ZigBee star topological entities must average the readings and send the average reading to the ZigBee coordinator.

11.1.10 Step 7: Development of the Markov Process for the ZigBee Star Topological Entity

In step 7 of the algorithm the Markov process for the topological entity is constructed and expressions for the probability and rewards matrices are found. The Markov process for the ZigBee star topological entity and expressions for the probability and rewards matrices are developed in this section. The tasks identified in Section 11.1.9 form the basis for development of the Markov process in this section. The energy consumed by each star topological entity as well as the ZigBee coordinator is determined too. The energy consumption is used to determine the accuracy of the ZigBee star topological entities to the results using all the base level entities. The operation of each star topological entity is described by the Markov process shown in Figure 11.12.

The probability and reward matrices have the following form.

$$P = \begin{bmatrix} P_{II} & P_{IR} & P_{IC} & P_{IT} \\ P_{RI} & 0 & 0 & 0 \\ P_{CI} & 0 & 0 & 0 \\ P_{TI} & 0 & 0 & 0 \end{bmatrix} \tag{11-66}$$

$$R = \begin{bmatrix} R_{II} & R_{IR} & R_{IC} & R_{IT} \\ R_{RI} & 0 & 0 & 0 \\ R_{CI} & 0 & 0 & 0 \\ R_{TI} & 0 & 0 & 0 \end{bmatrix} \tag{11-67}$$

As stated in the previous section, the star topology performs three basic tasks; receive a temperature reading; receive a message from the ZigBee coordinator; and calculate the average temperature and transmit the average temperature reading to the ZigBee coordinator.

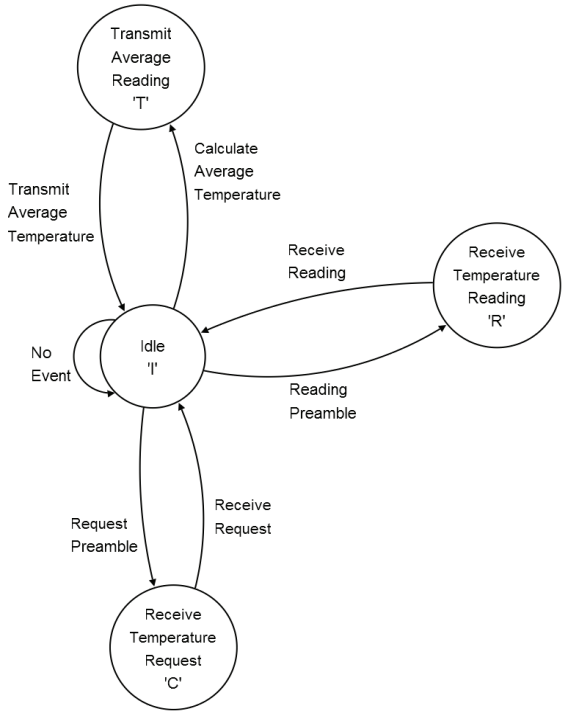

Figure 11.12: Markov process describing the star topological entity.

There are, N_T, temperature sensors contained in each of the star topologies. Each of the, N_T, temperature sensors periodically takes a temperature reading and sends that reading to the router with a frequency of, $F_{Reading}$. The total number of readings sent by one temperature sensor per day, $M_{Reading}$, is,

$$M_{\text{Reading}} = (\text{Seconds in One Day}) * F_{\text{Reading}} \tag{11-68}$$

Thus, the total number of periodic temperature readings sent to the router per day, $M_{Total\text{-}Periodic}$, is the sum of the number of periodic readings for each temperature sensor in the topology,

$$M_{\text{Total}-\text{Periodic}} = N_T * M_{\text{Reading}} \tag{11-69}$$

The router receives, M_{Req}, requests for a temperature reading from the ZigBee coordinator per day. Each request for a temperature reading causes, all, N_T, temperature sensors to take and send a temperature reading to the router. Hence, the number of temperature readings received by the router due to requests, $M_{Req\text{-}Reads}$, is,

$$M_{\text{Req}-\text{Read}} = M_{\text{Req}} * N_T \tag{11-70}$$

The router collects the temperature readings from all attached temperature sensors and then calculates the average temperature. The average temperature is transmitted to the ZigBee coordinator. By sending only the average temperature rather than all temperature readings, the number of messages is reduced lowering energy consumed for transmitting and reducing collisions to increase throughput. The number of times per day that the router calculates and sends the average temperature to the coordinator, M_{Calc}, is,

$$M_{\text{Calc}} = M_{\text{Req}} + M_{\text{Reading}} \tag{11-71}$$

The time required to receive the whole ZigBee preamble is denoted as, $T_{Preamble}$. The preamble is defined as a 4 byte sequence in the ZigBee and IEEE 802.15.4 standards [54, 64]. The time required to check the device's counters to determine if it is time to take a periodic temperature reading also requires, $T_{Preamble}$, seconds. Hence, the period of one day can be broken up into, T, intervals of, $T_{Preamble}$, seconds.

$$T = \frac{(\text{Seconds in One Day})}{T_{\text{Preamble}}} \tag{11-72}$$

The probability that the topology receives a temperature reading from one of the temperature sensor within the topology, P_{IR}, is,

$$P_{IR} = \frac{M_{\text{Req}-\text{Read}} + M_{\text{Total}-\text{Periodic}}}{T} \tag{11-73}$$

The probability that the topology receives a request for a temperature reading from the coordinator, P_{IC}, is,

$$P_{IC} = \frac{M_{\text{Req}}}{T} \tag{11-74}$$

The probability that the topology calculates the average temperature and sends the average temperature to the coordinator, P_{IT}, is,

$$P_{IT} = \frac{M_{\text{Calc}}}{T} \tag{11-75}$$

The probability that the topology remains idle, in state I, P_{II}, is simply,

$$P_{II} = 1 - \left(P_{IR} + P_{IT} + P_{IC}\right) \tag{11-76}$$

It is assumed that,

$$M_{\text{Req-Read}} + M_{\text{Total-Periodic}} + M_{Calc} + M_{\text{Req}} < T \tag{11-77}$$

Otherwise the network would not be realizable because more messages would be transmitted than there is time in one day. The remaining transition probabilities are 1, thus,

$$P_{TI} = P_{RI} = P_{CI} = 1 \tag{11-78}$$

In state I, the topology is waiting to perform one of the three tasks; receive a reading; receive a request for a temperature reading; or to calculate and transmit the average temperature to the outside world. In state R, the router in the topology is receiving and processing one temperature reading from a temperature sensor within the topology. In state C, the router receives a request for a temperature reading from the ZigBee coordinator and then forwards that request to the temperature sensors contained in the topology.

The power consumed when the receiver and processor are active is denoted as, $PW_{PR\text{-}R}$, for the router, and $PW_{PR\text{-}T,}$ for the temperature sensor. The energy consumed when no event occurs, R_{II}, is,

$$R_{II} = T_{\text{Preamble}} * \left(PW_{\text{PR}-\text{R}} + N_T * PW_{\text{PR}-\text{T}}\right) \tag{11-79}$$

The energy consumed for the I to R transition, R_{IR}, is,

$$R_{IR} = T_{\text{Preamble}} * \left(PW_{\text{PR}-\text{R}} + N_T * PW_{PR-T}\right) \tag{11-80}$$

The energy consumed to receive the preamble for the I to R and I to C transitions, R_{IR}, and R_{IC}, respectively is equal to that of R_{II}.

$$R_{IC} = R_{II} = T_{\text{Preamble}} * \left(PW_{\text{PR}-\text{R}} + N_T * PW_{\text{PR}-\text{T}}\right) \tag{11-81}$$

When a request for a temperature reading is received by the router from the ZigBee coordinator, the router must receive and process the message and finally forward the request for a temperature reading to the topology. Each temperature sensor must also receive and process the request. The only message that the ZigBee coordinator sends to the topology is to request a reading. Hence, all messages from the ZigBee coordinator are the same length denoted by, L_{CMsg}. The ZigBee message consists of take a reading command requiring, $L_{ReadCmd}$, bytes. In this example, $L_{ReadCmd}$, is 1 byte. The overhead of the ZigBee data frame, $L_{Overhead}$, is 15 bytes (excluding the 4 byte preamble) [64]. Thus, the length of a ZigBee message, minus the preamble, L_{CMsg}, is 16 bytes. The time to receive one byte of a ZigBee message is denoted as, $T_{Byte\text{-}RX}$.

The router must process both the reading and the request for a temperature reading messages. The time required to process a temperature reading is denoted as, $T_{P\text{-}R\text{-}Rd}$, and the time required to process the request for a temperature reading is denoted as, $T_{P\text{-}R\text{-}RQ}$. The time to process the reading, $T_{P\text{-}R\text{-}Rd}$, can be obtained by dividing the number of instructions, $N_{i\text{-}R\text{-}Rd}$, required to process the reading by the number of instructions per second the processor can perform, $C_{i\text{-}R\text{-}Rd}$. Hence, $T_{P\text{-}R\text{-}Rd}$, is,

$$T_{\text{P-R-Rd}} = \frac{N_{\text{i-R-Rd}}}{C_{\text{i-R-Rd}}} \tag{11-82}$$

Similarly, the time to process the reading, $T_{\text{P-R-RQ}}$, can be obtained by dividing the number of instructions, $N_{\text{i-R-RQ}}$, required to process the reading by the number of instructions per second the processor can perform, $C_{\text{i-R-RQ}}$. Hence, $T_{\text{P-R-RQ}}$, is,

$$T_{\text{P-R-RQ}} = \frac{N_{\text{i-R-RQ}}}{C_{\text{i-R-RQ}}} \tag{11-83}$$

Likewise, the temperature sensor must process the request and the temperature reading. The temperature sensor must process both the reading and the request for a temperature reading messages. The time required process a temperature reading is denoted as, $T_{\text{P-T-Rd}}$, and the time required to process the request for a temperature reading is denoted as, $T_{\text{P-T-RQ}}$. The time to process the reading, $T_{\text{P-T-Rd}}$, can be obtained by dividing the number of instructions, $N_{\text{i-T-Rd}}$, required to process the reading by the number of instructions per second the processor can perform, $C_{\text{i-T-Rd}}$. Hence, $T_{\text{P-T-Rd}}$, is,

$$T_{\text{P-T-Rd}} = \frac{N_{\text{i-T-Rd}}}{C_{\text{i-T-Rd}}} \tag{11-84}$$

Similarly, the time to process the reading, $T_{\text{P-T-RQ}}$, can be obtained by dividing the number of instructions, $N_{\text{i-T-RQ}}$, required to process the reading by the number of instructions per second the processor can perform, $C_{\text{i-T-RQ}}$. Hence, $T_{\text{P-T-RQ}}$, is,

$$T_{\text{P-T-RQ}} = \frac{N_{\text{i-T-RQ}}}{C_{\text{i-T-RQ}}} \tag{11-85}$$

The transmitter in the router and temperature sensor require, $T_{\text{TXWake-R}}$, and $T_{\text{TXWake-T}}$, seconds respectively to wake-up before they can be used to transmit a message. Similarly, the receiver in the router and temperature sensor require, $T_{\text{RXWake-R}}$, and $T_{\text{RXWake-T}}$, seconds respectively to wake-up before they can be used to receive a message. The time to receive one

byte of information on the ZigBee network is denoted as, $T_{Byte\text{-}RX}$, and $T_{Byte\text{-}TX}$, seconds are required to transmit one byte of information on the ZigBee network. The length of the preamble, $L_{Preamble}$, is 4 bytes.

When a router receives a request for a temperature reading, it must transmit the request to the temperature sensors in the star. To disseminate the request for a temperature reading to the star topology, the router must first process the request. Next, the router wakes up the transmitter and transmits the message. Simultaneously, the temperature sensors receive the request. Finally, the router wakes up the receiver. Hence, the energy consumed to receive the request for a temperature reading from the ZigBee coordinator, R_{CI}, is,

$$\begin{aligned} R_{CI} &= \left(L_{CMsg} * T_{\text{Byte-RX}}\right) * \left(PW_{PR-R} + N_T * PW_{PR-T}\right) \\ &+ T_{\text{RXWake-R}} * PW_{PR-R} + T_{\text{RXWake-T}} * N_T * PW_{PR-T} \\ &+ \left(PW_{P-R}\right) * T_{\text{P-R-RQ}} + \left(PW_{PT-R}\right) * T_{TXWake-R} + \left(L_{CMsg} + L_{\text{Preamble}}\right) * \left(T_{\text{Byte-TX}}\right) * \left(PW_{PT-R}\right) \end{aligned} \tag{11-86}$$

The length of the data message containing the temperature reading is denoted as, L_{TMsg}. The message contains, $L_{TempOver}$, bytes of overhead. The overhead, $L_{TempOver}$, for a data message is 15 bytes (excluding the 4 byte preamble sequence) [64]. The data portion of the message consists of the temperature reading, requiring, L_{TData}, bytes. In this example, L_{TData}, is assumed to be 4 bytes. Hence, L_{TMsg}, is,

$$L_{\text{TMsg}} = L_{\text{TempOver}} + L_{\text{TData}} \tag{11-87}$$

The time required for the temperature sensor to wake-up is denoted as, T_{TSWake}. The time required to take and process a temperature reading is denoted as, T_{Temp}. The time required to process the reading includes the time to obtain the reading from the temperature sensor, $T_{R\text{-}Temp}$, and the time to process the reading, T_{Proc}. The time to process the reading, T_{Proc}, can be obtained by dividing the number of instructions, $N_{i\text{-}TS}$, required to process the reading by the number of instructions per second the processor can perform, $C_{i\text{-}TS}$. Hence, T_{Proc}, is,

$$T_{\text{Proc}} = \frac{N_{i-TS}}{C_{i-TS}} \tag{11-88}$$

T_{Temp} is then simply,

$$T_{\text{Temp}} = T_{\text{Proc}} + T_{\text{R-Temp}} \tag{11-89}$$

When the router in the topology receives a reading from a temperature sensor, a number of sub-tasks must take place in the temperature sensor before the reading is sent. The temperature sensor starts by deciding to take a reading, either because it is time for a periodic reading or because a reading request was received. First, the temperature sensor must be wakened and then read. The reading must then be processed and then finally sent to the router. The temperature sensor consumes, $PW_{PT\text{-}T}$, Watts when the processor and transmitter are active, and $PW_{PS\text{-}T}$, Watts when the temperature sensor and processor are active. The energy consumed to receive a temperature reading from a temperature sensor, R_{RI}, is,

$$\begin{aligned} R_{RI} = & \left(\left(L_{TMsg} + L_{\text{Preamble}}\right) * T_{\text{Byte-TX}} + T_{TXWake-T}\right) * \left(PW_{PT-T}\right) \\ & + \left(T_{\text{P-R-Rd}} + L_{TMsg} * T_{\text{Byte-RX}}\right) * PW_{PR-R} \\ & + T_{RXWake-T} * \left(PW_{PR-T}\right) + \left(T_{\text{TSWake}} + T_{\text{Temp}}\right) * \left(PW_{PS-T}\right) \end{aligned} \tag{11-90}$$

The router requires, T_{AVG}, seconds to calculate the average temperature. The value of, T_{AVG}, can be found by obtaining the number of instructions required to calculate the average, $N_{i\text{-}AVG}$, and the number of instructions that the processor can evaluate in one second, $C_{i\text{-}AVG}$. Hence, T_{AVG}, is,

$$T_{\text{AVG}} = \frac{N_{i-AVG}}{C_{i-AVG}} \tag{11-91}$$

The energy consumed while the router is calculating the average temperature, R_{IT}, is simply,

$$R_{IT} = (T_{AVG}) * (PW_{PR-R}) \tag{11-92}$$

The router must calculate the average temperature once readings from all temperature sensors are collected. This task can be broken into three sub-tasks. The first sub-task is for the router to determine that the last reading has been received and to then calculate the average reading. During this time the router activates the transmitter which requires time, $T_{TXWake\text{-}R}$, to wake-up before data can be transmitted. Last, the average temperature reading is transmitted to the ZigBee coordinator and the router wakes up the receiver. The average temperature reading has the same length as the temperature readings received from the temperature sensors. The energy consumed to transmit the message to the coordinator, R_{TI}, is,

$$R_{TI} = T_{TXWake-R} * (PW_{PT-R}) + (L_{TMsg} + L_{\text{Preamble}}) * (T_{\text{Byte-TX}}) * (PW_{PT-R}) + T_{RXWake-R} * (PW_{PR-R}) \tag{11-93}$$

The coordinator is identical to the coordinator analyzed in section 11.1.6, and the parameters listed in Table 11.6 for the coordinator will be used to evaluate the coordinator in this example. The parameters for the temperature sensors are identical to those used to analyze the temperature sensor in 11.1.4, and the appropriate parameters listed in Table 11.3 will be used for the temperature sensor parameters in the star topological entity. The parameters for the routers are identical to those used to analyze the routers in 11.1.5, and the appropriate parameters listed in Table 11.4 will be used for the router parameters in the star topological entity. The energy consumption of the entire network is the sum of the energy consumed by the two star topological entities and the energy consumed by the coordinator. The energy consumed by the two star topological entities, the coordinator, and the entire network over 1 day are listed in Table 11.8.

The percent difference, D, between the energy consumption calculated from the model where the Markov process for each base level entity was evaluated, $E_{Base\text{-}Level}$, and the model replacing the two routers and associated temperature sensors with two star topologies, $E_{Star\text{-}Topology}$, is calculated using,

$$D = \frac{\left| E_{\text{Base-Level}} - E_{\text{Star-Topology}} \right|}{E_{\text{Base-Level}}} \tag{11-94}$$

Table 11.8: Energy consumed for each entity and the entire network using the Star Topology.

Entity	Energy Consumed (mJ)
Star Topological Entity S1	67336006.64 mJ
Star Topological Entity S2	67336006.64 mJ
C1	16839316.60 mJ
Entire Network	151511329.88 mJ

The difference in the reported energy consumption of the two models, D, is 0.3300 %. This result indicates that the model employing the two star topologies in place of the two routers and the temperature sensors associated with those routers is accurate.

11.1.11 Step 8 for the ZigBee Star Topological Entity

These interactions will be used to help identify the ZigBee tree fundamental topological entity in the following section. In this section, the interactions of the star topological entity are identified. This is step 8 of the algorithm. The ZigBee star topological entity interacts with the ZigBee coordinator. All data is transmitted to the ZigBee coordinator and the ZigBee coordinator then transmits the data to the outside world. Any messages from the outside world are first received by the ZigBee coordinator and forwarded onto the ZigBee star topological entities.

11.1.12 Step 9: Identification of the ZigBee Tree Topological Entity

The Zigbee tree topological entity is identified based on the tasks and interactions of the ZigBee star topological entities and the ZigBee coordinator. This is step 9 in the algorithm. The example network illustrated in Figure 11.5, can be represented as a single topological entity based around the ZigBee coordinator and the two star topological entities attached to it. This topological entity consists of three entities, the ZigBee coordinator and the two star topological entities, and is illustrated in Figure 11.13.

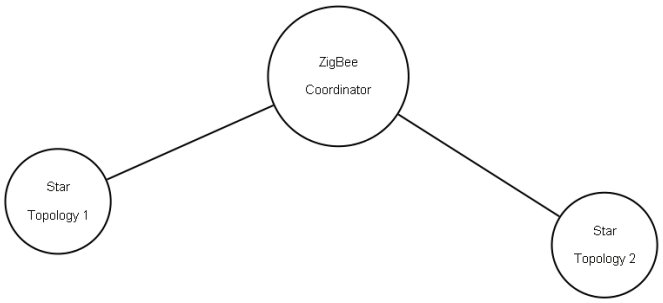

Figure 11.13: Tree topological entity.

With the ZigBee tree fundamental topological entity defined, existing algorithms can be used to search the basic communication graph for the ZigBee tree topological entities. These algorithms form the basis for CAD tools for integrated circuit design [46]. The ZigBee example network can be covered with a single ZigBee tree topological entity, reducing the order (number of Markov proceof the network from 3 as shown in Figure 11.13 to 1 as shown in Figure 11.14.

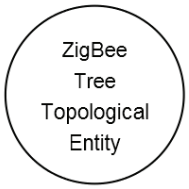

Figure 11.14: Covering of the ZigBee example network using a single ZigBee tree topological entity.

11.1.13 Step 6 for the ZigBee Tree Topological Entity

Steps 6 through 9 of the algorithm must be repeated to develop the model of the energy consumption for the Zigbee tree topological entity. Step 6 of the algorithm for the ZigBee tree topological entity is presented in this section. The ZigBee tree topology must perform two tasks.

First, the ZigBee tree topology must collect temperature readings, compute the average temperature and then transmit that average temperature to the outside world. The second task that is performed by the ZigBee tree topology is to receive and disseminate throughout the ZigBee tree topological entity the request for a temperature reading from the outside world.

11.1.14 Step 7 for the ZigBee Tree Topological Entity

The Markov process is developed for the ZigBee tree topological entity and expressions for the probability and rewards matrices are found in this section. This is step 7 of the algorithm for the ZigBee tree topological entity. The tasks identified in Section 11.1.13 form the basis for constructing the Markov process in this section. The ZigBee coordinator receives messages from the two star topologies containing the average temperature reading. The coordinator forwards these readings to the outside world, and receives requests for a temperature reading from the outside world. The requests for a temperature reading received by the coordinator must be converted from the inter-PAN message format to the intra-PAN format and transmitted to the two star topologies. The Markov process describing the ZigBee coordinator tree topological entity is shown in Figure 11.15. The probability and reward matrices have the following form.

$$P = \begin{bmatrix} P_{II} & P_{IR} & P_{IT} \\ P_{RI} & 0 & 0 \\ P_{TI} & 0 & 0 \end{bmatrix} \tag{11-95}$$

$$R = \begin{bmatrix} R_{II} & R_{IR} & R_{IT} \\ R_{RI} & 0 & 0 \\ R_{TI} & 0 & 0 \end{bmatrix} \tag{11-96}$$

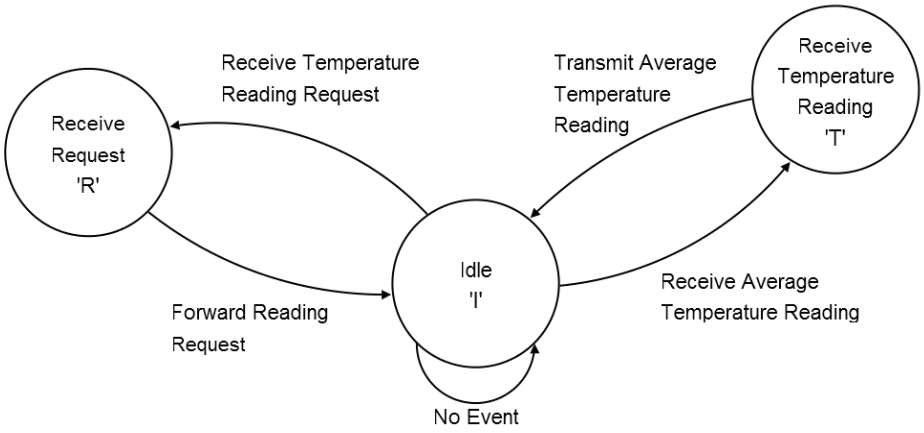

Figure 11.15: Markov process describing the ZigBee coordinator tree topological entity.

The preamble requires, $T_{Preamble}$, seconds to be received or transmitted. Hence, the period of one day can be broken up into, T, periods each equal to time, $T_{Preamble}$.

$$T = \frac{(\text{Seconds in One Day})}{T_{\text{Preamble}}} \tag{11-97}$$

The ZigBee tree topological entity contains, N_S, ZigBee star topological entities. Each star topology automatically takes temperature readings and sends the average temperature calculated from those readings to the coordinator. Each star takes temperature readings with a frequency of, $F_{Reading\text{-}x}$, where, x, denotes one of the, N_S, star topologies. The total number of automatic temperature readings that the coordinator receives per day, $M_{Reading\text{-}x}$, and again, x, denotes one of the, N_S, star topologies, is,

$$M_{\text{Reading}} = \sum_{x=1}^{N_S} (\text{Seconds in One Day}) * F_{\text{Reading}-x} \tag{11-98}$$

The topology also receives requests for a temperature reading from outside of the PAN. A total of, M_{OUT}, requests for a temperature reading are received by the topology per day from outside of the PAN. These, M_{OUT}, messages generate one average temperature reading from

each ZigBee star topological entity within the topology. Hence, the number of average temperature readings that are received, M_{ATR}, is,

$$M_{\text{ATR}} = M_{\text{Reading}} + M_{\text{OUT}} * N_{\text{S}} \tag{11-99}$$

The probability that a temperature reading is received, P_{IT}, is,

$$P_{IT} = \frac{M_{\text{ATR}}}{T} \tag{11-100}$$

Likewise, the probability that a request for a temperature reading is received, P_{IR}, is,

$$P_{IR} = \frac{M_{\text{OUT}}}{T} \tag{11-101}$$

The following condition is assumed to hold true. Otherwise there would not be sufficient time in one day for all the operations.

$$M_{\text{ATR}} + M_{\text{OUT}} < T \tag{11-102}$$

The probability that the topology remains idle, P_{II}, is simply,

$$P_{II} = 1 - \left(P_{IR} + P_{IT}\right) \tag{11-103}$$

The remaining probabilities are all 1. Hence,

$$P_{RI} = P_{TI} = 1 \tag{11-104}$$

The power consumption of the coordinator when only the processor and receiver are active is denoted by, $PW_{PR\text{-}C}$. The power consumed when the receiver and processor are active is denoted as, $PW_{PR\text{-}R}$, for the router, and $PW_{PR\text{-}T,}$ for the temperature sensor. The power

consumption of an entire ZigBee star topology when only the processor and receiver are active is denoted by, $PW_{PR\text{-}S}$, where, x, denotes one of the, N_S, star topology.

$$PW_{\mathrm{PR-S-x}} = N_{\mathrm{T-x}} * PW_{PR-T} + PW_{PR-R} \tag{11-105}$$

The energy consumption of the topology while it is idle, R_{II}, is,

$$R_{II} = T_{\mathrm{Preamble}} * \left(PW_{PR-C} + \sum_{x=1}^{N_S} PW_{\mathrm{PR-S-x}} \right) \tag{11-106}$$

The coordinator can receive two messages, the request for a reading from the outside world, or the temperature reading from a temperature sensor. The request for a temperature reading originates outside the PAN managed by the coordinator. Hence, the inter-PAN message format and long addresses are used. The command to request the temperature, L_{ReadCmd}, is 1 byte. The message overhead is, $L_{\mathrm{Overhead\text{-}O}}$, is 29 bytes (preamble excluded). The length of the entire message, L_{OUT}, is 30 bytes.

$$L_{\mathrm{OUT}} = L_{\mathrm{Overhead-O}} + L_{\mathrm{ReadCmd}} \tag{11-107}$$

The time to receive and transmit one byte of a ZigBee message is denoted as, $T_{\mathrm{Byte\text{-}RX}}$, and $T_{\mathrm{Byte\text{-}TX}}$, respectively. The energy consumed for the topology to receive a request for a temperature reading, R_{IR}, is,

$$R_{IR} = \left(\left(L_{\mathrm{OUT}} + L_{\mathrm{Preamble}} \right) * T_{\mathrm{Byte-RX}} \right) * PW_{PR-C} \tag{11-108}$$

The length of the preamble, L_{Preamble}, is 4 bytes. The length of the data message containing a temperature reading is denoted as, L_{TMsg}. The message contains, L_{TempOver}, bytes of overhead. The overhead, L_{TempOver}, for a data message is 15 bytes (excluding the 4 byte preamble sequence) [64]. The data portion of the message consists of the temperature reading, requiring, L_{TData}, bytes. In this example, L_{TData}, is assumed to be 4 bytes. Hence, L_{TMsg}, is,

$$L_{\text{TMsg}} = L_{\text{TempOver}} + L_{\text{TData}} \tag{11-109}$$

The coordinator receives average temperature readings from the attached star topologies. The star topology requires four steps to calculate the average temperature. First, all, N_T, tags in the star topology must take a temperature reading and transmit that reading to the router in the topology. This step consumes, R_{Sample}, Joules. Simultaneously, the router must receive and collect all the temperature readings, consuming, $R_{\text{Temp-RX}}$, Joules. When the router receives all the temperature readings, it must calculate the average, consuming, R_{Calc}, Joules. Finally, the router transmits the average temperature reading to the coordinator, consuming, $R_{\text{Temp-TX}}$, Joules.

The energy consumed, R_{Sample}, by a star topology to take and transmit all, N_T, temperature readings is,

$$R_{\text{Sample}} = \begin{pmatrix} \left(T_{\text{Temp}} + T_{\text{TS-Wake}}\right) * PW_{PS-T} + \left(L_{\text{TMsg}} + L_{\text{Preamble}}\right) * T_{\text{Byte-TX}} * PW_{\text{PT-T}} \\ + \left(T_{\text{Wake-TX-T}}\right) * PW_{\text{PT-T}} + T_{\text{Wake-RX-T}} * PW_{\text{PR-T}} \end{pmatrix} * N_T \tag{11-110}$$

where, T_{Temp}, is the time required to take and process a temperature reading; found using (11-14); $T_{\text{TS-Wake}}$, is the time required for the temperature sensor to wake-up; $PW_{\text{PS-T}}$, is the power consumed by the temperature sensor with the processor and temperature sensor active; $PW_{\text{PT-T}}$, is the power consumed by the temperature sensor with the transmitter and processor active; $T_{\text{Wake-TX-T}}$, is the time required for the transmitter on the temperature sensor to wake-up; and $T_{\text{Wake-RX-T}}$, is the time required for the receiver on the temperature sensor to wake-up. The router requires time, $T_{\text{P-R-Rd}}$, to process the reading, and $T_{\text{P-R-Rd}}$, can be found using (11-82). The energy consumed by the router to receive all, N_T, temperature readings, for a particular star topology, $R_{\text{Temp-RX}}$, is,

$$R_{\text{Temp-RX}} = \left(\left(L_{\text{TMsg}} + L_{\text{Preamble}}\right) * T_{\text{Byte-RX}} + T_{\text{P-R-Rd}}\right) * PW_{PR-R} * N_T \tag{11-111}$$

The router requires time, T_{AVG}, to compute the average temperature, and T_{AVG}, can be found using (11-91). The energy consumed by the router to calculate the average temperature, R_{Calc}, is

$$R_{\text{Calc}} = T_{\text{AVG}} * PW_{\text{P-R}} \tag{11-112}$$

where, PW_{P-R}, represents the power consumed by the router when only the processor active. Finally, the router must transmit the average temperature reading to the coordinator. The energy consumed by the router to transmit the message to the coordinator, $R_{Temp-TX}$, is,

$$R_{\text{Temp-TX}} = T_{\text{Wake-RX-R}} * PW_{PR-R} + \left(\left(L_{\text{TMsg}} + L_{\text{Preamble}}\right) * T_{\text{Byte-TX}} + T_{\text{Wake-TX-R}}\right) * PW_{PT-R} \tag{11-113}$$

The energy consumed by the star topology to transmit an average temperature reading, $E_{TX-Temp}$, is simply,

$$E_{\text{TX-Temp}} = R_{\text{Temp-TX}} + R_{\text{Calc}} + R_{\text{Temp-RX}} + R_{\text{Sample}} \tag{11-114}$$

The energy consumed by the coordinator receiving the average temperature reading from a star topology, R_{IT}, is,

$$R_{IT} = E_{\text{TX-Temp}} + \left(L_{\text{TMsg}} + L_{\text{Preamble}}\right) * T_{\text{Byte-RX}} * \left(PW_{PR-C}\right) \tag{11-115}$$

The energy consumed by the coordinator to transmit the average temperature to the outside world, R_{TI}, is requires two steps. First, the coordinator must process the average temperature message converting that message into the inter-PAN message format because the message will be transmitted outside of the coordinator's PAN. Second, the coordinator must wake up the transmitter and then transmit the message. Just like the request for a temperature reading message the average temperature reading that is sent to the outside world is sent to another PAN so the inter-PAN format and long addresses are used. The average temperature reading, $L_{Temp-Avg}$, is 4 bytes. The message overhead is, $L_{Overhead-O}$, is 29 bytes (preamble excluded). Hence, the length of the entire message, $L_{Temp-TX}$, is 33 bytes.

$$L_{\text{Temp-TX}} = L_{\text{Overhead-O}} + L_{\text{Temp-Avg}} \tag{11-116}$$

The coordinator requires, $T_{P\text{-}Read}$, seconds to process the average temperature reading, and $T_{P\text{-}Read}$, can be calculated using (11-59). The transmitter requires, $T_{Wake\text{-}TX\text{-}C}$, seconds to wake-up, and the coordinator consumes, $PW_{P\text{-}C}$, Watts when only the processor is active, and $PW_{PT\text{-}C}$, Watts when the processor and transmitter are active. Further, the receiver is turned off during the transmission and requires, $T_{Wake\text{-}RX\text{-}C}$, seconds to wake-up. Thus, the energy consumed by the coordinator to transmit the average temperature to the outside world, R_{TI}, is,

$$R_{TI} = \left(T_{\text{Wake-TX-C}} + \left(L_{\text{Temp-TX}} + L_{\text{Preamble}}\right) * T_{\text{Byte-TX}}\right) * PW_{PT-C} + T_{\text{P-Read}} * PW_{\text{P-C}} + T_{\text{Wake-RX-C}} * PW_{\text{PR-C}} \quad (11\text{-}117)$$

When the coordinator receives a request for a temperature reading from the outside world, it must forward the request to the, N_S, star topologies it is responsible for. Three steps are required to disseminate the request to the temperature sensors. First, the coordinator must process the message and convert the message to intra-PAN message format from the inter-PAN format because the message was sent from another PAN. Second, the coordinator must wake-up the transmitter and broadcast the request to the, N_S, star topologies to which it is connected. The coordinator, after transmitting the request, wakes up the receiver. Third, each of the, N_S, star topologies must process and retransmit the request. The coordinator requires, $T_{P\text{-}Req\text{-}C}$, seconds to process the request, and this is calculated using (11-60). Similarly, the router in the star topology requires, $T_{P\text{-}Req\text{-}R}$, seconds to process the request and can be found using (11-33).

The ZigBee message consists of take a reading command requiring, $L_{ReadCmd}$, bytes. In this example, $L_{ReadCmd}$, is 1 byte. The overhead of the ZigBee data frame, $L_{Overhead}$, is 15 bytes (excluding the 4 byte preamble sequence) [64]. Thus, the length of a ZigBee message, minus the preamble, L_{CMsg}, is 16 bytes. Thus, the energy consumed to disseminate the request for a temperature reading to the temperature sensors, R_{RI}, is,

$$
\begin{aligned}
R_{RI} = & \left(T_{\text{Wake-TX-C}} + \left(L_{\text{CMsg}} + L_{\text{Preamble}}\right) * T_{\text{Byte-TX}}\right) * PW_{PT-C} \\
& + T_{\text{P-Read}} * PW_{\text{P-C}} + T_{\text{Wake-RX-C}} * PW_{\text{PR-C}} \\
& + \left(L_{\text{CMsg}} + L_{\text{Preamble}}\right) * T_{\text{Byte-RX}} * \left(N_S * PW_{PR-R} + \sum_{x=1}^{N_S} N_{T_x} * PW_{PR-T} \right) \\
& + \left(T_{\text{Wake-TX-R}} + \left(L_{\text{CMsg}} + L_{\text{Preamble}}\right) * T_{\text{Byte-TX}}\right) * N_S * PW_{PT-R} \\
& + T_{\text{P-Req-R}} * N_S * PW_{\text{P-R}} + N_S * T_{\text{Wake-RX-R}} * PW_{\text{PR-R}}
\end{aligned}
\tag{11-118}
$$

11.1.15 Steps 8, 9, and 10 for the ZigBee Tree Topological Entity

Steps 8, 9, and 10 of the algorithm must be repeated for the ZigBee tree topological entity. The tree topological entity combines the coordinator and two star topologies in the previous example together into a single entity. There are no interactions between topological entities at this level because the network can be covered with a single ZigBee tree topological entity. Because a single ZigBee tree topological entity can cover the network, no larger topological entities can be identified in step 9. Step 10 is satisfied because the entire network can be covered by a single topological entity and the algorithm finishes.

The tree topological entity requires parameters from the base level entities such as power consumed in different states and the time required to perform various tasks, and these parameters are identical to those found for the base level entities. This entity uses the parameters for the temperature sensor, router, and coordinator found in Table 11.2, Table 11.4, and Table 11.6 respectively, for the appropriate parameters for this entity. The energy consumption of the entire network is simply of the energy consumed by the tree topological entity because with the tree topological entity the network is reduced to a single topological entity. The energy consumed by the entire network over 1 day is listed in Table 11.9.

Table 11.9: Energy consumption of the example using the Tree Topological Entity.

Entity	Energy Consumed (mJ)
Tree Topological Entity T1	150875856.85 mJ
Entire Network	150875856.85 mJ

The percent difference, D, between the energy consumption calculated from the model where the Markov process for each base level entity was evaluated, $E_{Base\text{-}Level}$, and the model using the tree topological entity, $E_{Tree\text{-}Topology}$, is calculated using,

$$D = \frac{\left| E_{\text{Base-Level}} - E_{\text{Tree-Topology}} \right|}{E_{\text{Base-Level}}} \tag{11-119}$$

The difference in the reported energy consumption of the two models, D, is 0.0865 %. This result indicates that the model employing the tree topological entity is accurate. The algorithm is satisfied and finishes because the entire network can be represented by a single ZigBee tree topological entity.

11.2 SUMMARY OF THE ENERGY CONSUMPTION

The energy consumed by the network for a period of 1 day was calculated using three different methods. First, the energy consumed by the network was calculated by first calculating the energy consumed by each base level entities and then summing the values to obtain the energy consumption for the entire network (Section 11.1.4 for the temperature sensor, Section 11.1.5 for the ZigBee router, and Section 10.2.5 for the ZigBee coordinator). Next, the two ZigBee star topological entities replaced the two routers and ten temperature sensors in the example network as shown in Figure 11.11. Again, the energy consumption for the period of 1 day was computed for these three entities (ZigBee coordinator and two ZigBee star topological entities) and then summed to arrive at the energy consumption of the entire network (Section 11.1.10). Finally, the ZigBee tree topological entity was used to cover the entire network with a single entity and the energy consumption of the ZigBee tree topological entity was calculated for the period of 1 day (Sections 11.1.14 and 11.1.15). The energy consumed for each case is listed in Table 11.10. The percent difference, D, is computed as follows, where, $E_{Base\text{-}Level}$, is the energy consumed calculated using the base level entities, and E_{Test}, is the energy consumed using one of the other two sets of topological entities (ZigBee coordinator and two ZigBee star topological entities or one ZigBee tree topological entity).

$$D = \frac{\left|E_{\text{Base-Level}} - E_{\text{Test}}\right|}{E_{\text{Base-Level}}} \tag{11-120}$$

Table 11.10: Energy consumed calculated using the three different sets of entities and percent difference from the energy consumption of the base level entities.

Entity	Energy Consumed (mJ)	Percent Difference (%)
Base Level Entities	151006451.538448 mJ	0 %
ZigBee Coordinator and Two ZigBee Star Topological Entities	151511329.875985 mJ	0.3343 %
One ZigBee Tree Topological Entity	150875856.845581 mJ	0.086483 %

The results show that the percent difference is no more than 0.34 % in the worst case (using the ZigBee coordinator and two ZigBee star topological entities) and the method of using the topological entities is considered accurate.

11.3 SUMMARY OF STEPS IN ALGORITHM TO IDENTIFY TOPOLOGICAL ENTITIES

The steps of the algorithm to identify the topological entities were illustrated in the previous section analyzing the energy consumption of the ZigBee example network. Step 1, identification of the base level entities was presented in Section 11.1.1. Step 2, the identification of the tasks performed by each base level entity type was presented in Section 11.1.2. In Sections 11.1.3, 11.1.4 (temperature sensor), 11.1.5 (router), 11.1.6 (coordinator), the Markov processes and the expressions for the probability and reward matrices are developed for the three types of base level entities. The Markov processes and the expressions for the probability and reward matrices for the temperature sensor were presented in Section 11.1.4. The Markov processes and the expressions for the probability and reward matrices for the ZigBee router were presented in

Section 11.1.5. The Markov processes and the expressions for the probability and reward matrices for the ZigBee coordinator were presented in Section 11.1.6. Step 4, the identification of interactions between the base level entities was presented in Section 11.1.7. Step 5, the identification of topological entities, specifically the identification of the two ZigBee star topological entities, was presented in Section 11.1.8. Section 11.1.9 presents the tasks that the ZigBee star topological entities perform and this was Step 6 of the algorithm. The Markov processes and the expressions for the probability and reward matrices are developed for the ZigBee star topological entity in Section 11.1.10 and this was step 7 of the algorithm. In Section 11.1.11 step 8 of the algorithm, identification of the interactions between the ZigBee star topologies and the ZigBee coordinator was presented. Step 9, the identification of topological entities, in this case the identification of the ZigBee tree topological entity, was presented in Section 11.1.12. Step 6 for the ZigBee tree topological entity, identifying the tasks of the ZigBee tree topological entity was presented in Section 11.1.13. Step 7 for the ZigBee tree topological entity, development of the Markov processes and the expressions for the probability and reward matrices, was presented in Section 11.1.14. Steps 8, 9, and 10, for the ZigBee tree topological entity were presented in Section 11.1.15.

11.4 EVALUATION TIME FOR LARGER NETWORKS

To determine how the method in this work scales with the size of the network, the example ZigBee network was doubled and then doubled again (quadruple the size of the original network). The single size network is the network analyzed in the previous section. When the network is double a new coordinator and two star topologies are added to the network as shown in Figure 11.16.

The double size network contains two ZigBee tree topological entities, TOP1 and TOP2. The two ZigBee tree topological entities, TOP1, and TOP2, are shown in Figure 11.17 and Figure 11.18, respectively.

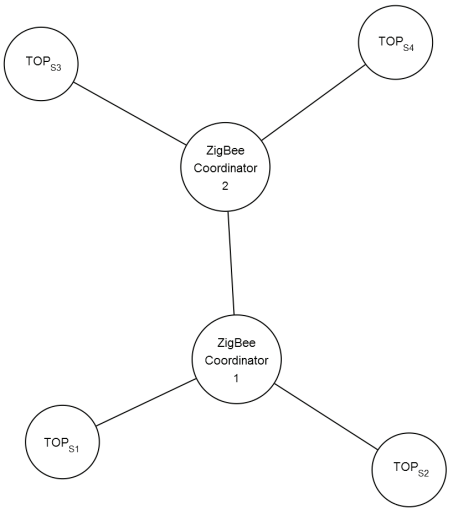

Figure 11.16: Double size example network.

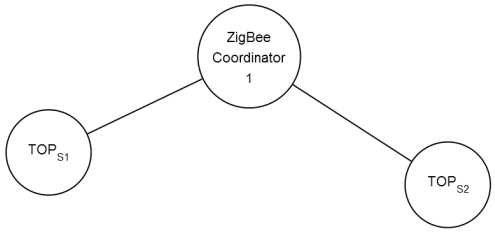

Figure 11.17: ZigBee tree topological entity, TOP1, in the double size network.

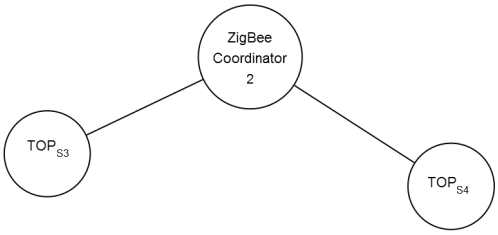

Figure 11.18: ZigBee tree topological entity, TOP2, in the double size network.

Both TOP1 and TOP2 contain two ZigBee star topological entities, and one ZigBee coordinator. Each star topological entity contains five temperature sensors and one ZigBee router as shown in Figure 11.9 and Figure 11.10. Each ZigBee star topological entity connects to one ZigBee coordinator as shown in Figure 11.17 and Figure 11.18. The four individual ZigBee star topological entities TOP_{S1}, TOP_{S2}, TOP_{S3}, and TOP_{S4}, are shown in the following four figures.

The base entities contained in TOP_{S1} are shown in Figure 11.19. The base entities contained in TOP_{S2} are shown in Figure 11.20. The base entities contained in TOP_{S3} are shown in Figure 11.21. The base entities contained in TOP_{S4} are shown in Figure 11.22.

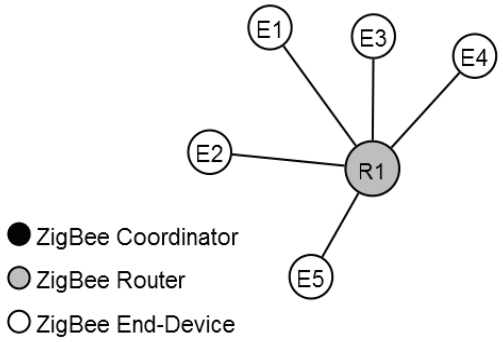

Figure 11.19: ZigBee star topological entity 1, TOP_{S1}.

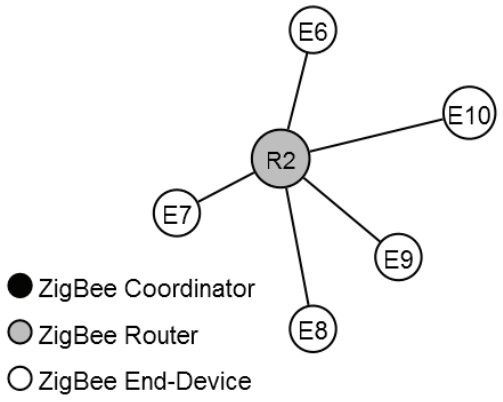

Figure 11.20: ZigBee star topological entity 2, TOP_{S2}.

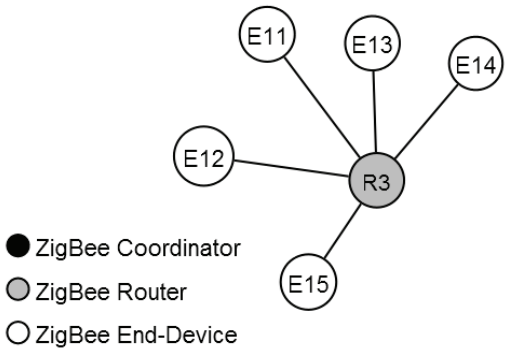

Figure 11.21: ZigBee star topological entity 3, TOP_{S3}.

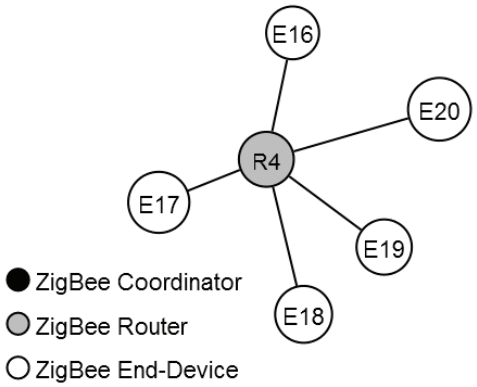

Figure 11.22: ZigBee star topological entity 4, TOP_{S4}.

ZigBee routers R1 and R2 are connected to ZigBee coordinator 1, while ZigBee routers R3 and R4 are connected to ZigBee coordinator 2. The double size network can be covered by two ZigBee tree topological entities as shown in Figure 11.23. With the ZigBee tree fundamental topological entity defined, existing algorithms can be used to search and find the two ZigBee tree topological entities (TOP1 and TOP2) contained within the basic communication graph of the double size example network as shown in Figure 11.23. These algorithms form the basis for CAD tools for integrated circuit design [46].

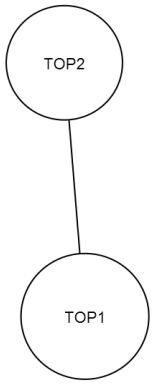

Figure 11.23: Covering of the double size network with two ZigBee tree topological entities.

The interaction between the two ZigBee tree topological entities resembles the fully connected mesh topology on two entities. Both of the ZigBee tree topological entities can be combined into a single topological entity, the ZigBee multi-tree topological entity. The ZigBee multi-tree topological entity must perform the same tasks as the ZigBee tree topological entity and the Markov process for the ZigBee multi-tree topological entity and ZigBee tree topological entity are identical. The Markov process for the ZigBee multi-tree topological entity is shown in Figure 11.24. This reduces the order (number of Markov processes) of the network from 6, as shown in Figure 11.16, to 2, as shown in Figure 11.23. The probability and reward matrices have the following form.

$$P = \begin{bmatrix} P_{II} & P_{IR} & P_{IT} \\ P_{RI} & 0 & 0 \\ P_{TI} & 0 & 0 \end{bmatrix} \tag{11-121}$$

$$R = \begin{bmatrix} R_{II} & R_{IR} & R_{IT} \\ R_{RI} & 0 & 0 \\ R_{TI} & 0 & 0 \end{bmatrix} \tag{11-122}$$

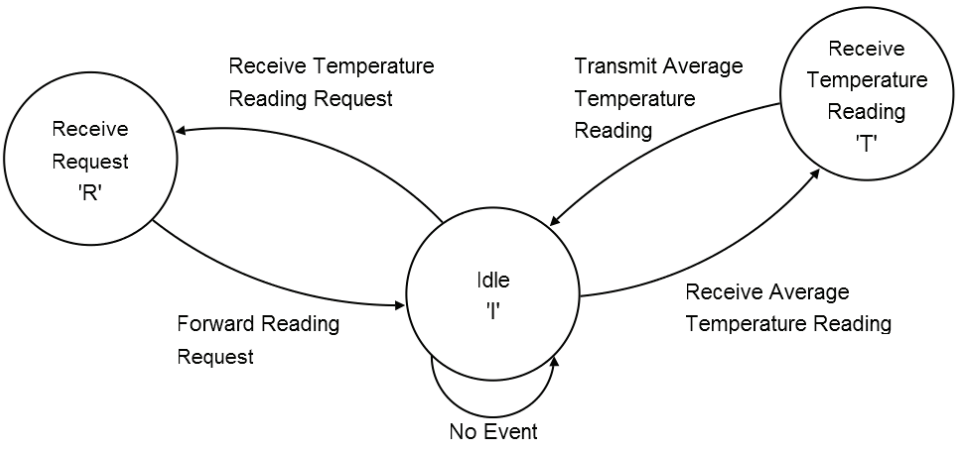

Figure 11.24: Markov process describing the ZigBee coordinator tree topological entity.

Each of the ZigBee tree topological entities that are contained in the ZigBee multi-tree topological entity contribute to the operation of the ZigBee multi-tree topological entity. Thus, the probability matrix for the ZigBee multi-tree topological entity is obtained by adding the probability matrices of each of the ZigBee tree topological entities contained in the ZigBee multi-tree topological entity together and dividing by the number of ZigBee tree topological entities contained in the ZigBee multi-tree topological entity. Because each task represents multiple ZigBee tree topological entities performing that task the rewards matrix for the ZigBee multi-tree topological entity is simply the sum of the rewards matrices of each of the ZigBee tree topological entities contained in the ZigBee multi-tree topological entity. The network can be covered by a single ZigBee multi-tree topological entity as shown in Figure 11.25.

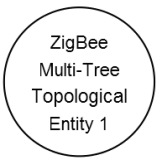

Figure 11.25: Covering of the network with a single ZigBee multi-tree topological entity.

With the ZigBee multi-tree fundamental topological entity defined, existing algorithms can be used to search and find the single ZigBee multi-tree topological entities contained within the basic communication graph of the double size example network as shown in Figure 11.23. These algorithms form the basis for CAD tools for integrated circuit design [46]. The order (number of Markov processes) of the network is reduced from 2 ZigBee tree topological entities to a single ZigBee multi-tree topological entities as shown in Figure 11.25.

The parameters of all temperature sensors are identical to those used for the used for the temperature sensors in Section 11.1.4. The parameters for the odd number routers (R1 and R3) are identical to those used for router R1 in Section 11.1.5. The parameters for the even number routers (R2 and R4) are identical to those used for router R2 in Section 11.1.5. The parameters for the four ZigBee coordinators are identical to the parameters used for the ZigBee coordinator in Section 11.1.6. The parameters for the ZigBee star topological entity TOP_{S1} are identical to those used for the ZigBee star topological entity TOP_{S1} in Section 11.1.10. Likewise, the parameters for the ZigBee star topological entity TOP_{S2} are identical to those used for the star topological entity TOP_{S2} in Section 11.1.10. The parameters used for the two ZigBee tree topological entities (TOP1 and TOP2) are identical to those used for the ZigBee tree topological entity in Section 11.1.15.

When the network is quadrupled, three additional coordinators and six additional star topologies are added as shown in Figure 11.26.

Again, each star topological entity contains five temperature sensors and one ZigBee router as shown in Figure 11.31 through Figure 11.38. The quadruple size network contains four ZigBee tree topological entities (TOP1, TOP2, TOP3, and TOP4) each containing one ZigBee coordinator and two ZigBee star topological entities.

The ZigBee tree topological entity, TOP1, having ZigBee coordinator 1 as the root is shown in Figure 11.27. The ZigBee tree topological entity, TOP2, having ZigBee coordinator 2 as the root of the tree is shown in Figure 11.28. The ZigBee tree topological entity, TOP3, having ZigBee coordinator 3 as the root of the tree is shown in Figure 11.29. The ZigBee tree topological entity, TOP4, having ZigBee coordinator 4 as the root of the tree is shown in Figure 11.30.

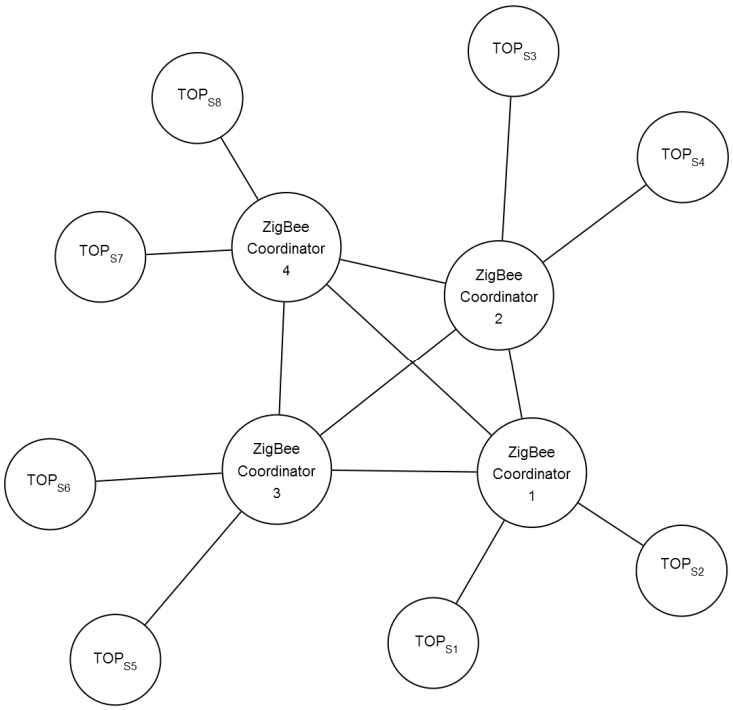

Figure 11.26: Quadruple size example network.

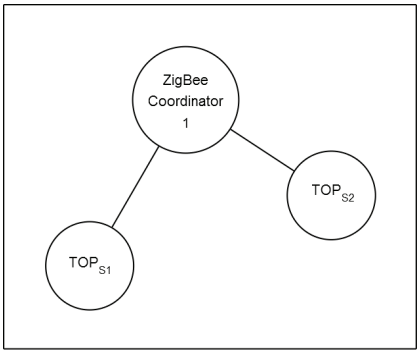

Figure 11.27: ZigBee tree topological entity, TOP1, rooted at ZigBee coordinator 1.

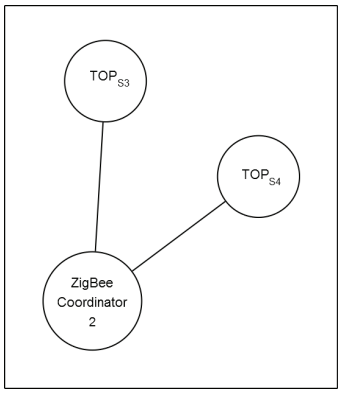

Figure 11.28: ZigBee tree topological entity, TOP2, rooted at ZigBee coordinator 2.

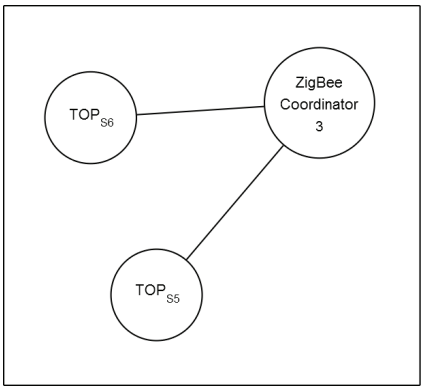

Figure 11.29: ZigBee tree topological entity, TOP3, rooted at ZigBee coordinator 3.

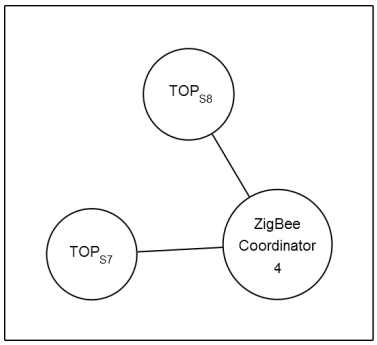

Figure 11.30: ZigBee tree topological entity, TOP4, rooted at ZigBee coordinator 4.

The eight ZigBee star topological entities (TOP_{S1}, TOP_{S2}, TOP_{S3}, TOP_{S4}, TOP_{S5}, TOP_{S6}, TOP_{S7}, and TOP_{S8}) shown in the previous four figures are reduced to their base level entities in the following eight figures. The four individual ZigBee star topological entities TOP_{S1}, TOP_{S2}, TOP_{S3}, TOP_{S4}, TOP_{S5}, TOP_{S6}, TOP_{S7}, and TOP_{S8}, are shown in the following eight figures.

The base entities contained in TOP_{S1} are shown in Figure 11.31. The base entities contained in TOP_{S2} are shown in Figure 11.32. The base entities contained in TOP_{S3} are shown in Figure 11.33. The base entities contained in TOP_{S4} are shown in Figure 11.34. The base entities contained in TOP_{S5} are shown in Figure 11.35. The base entities contained in TOP_{S6} are shown in Figure 11.36. The base entities contained in TOP_{S7} are shown in Figure 11.37. The base entities contained in TOP_{S8} are shown in Figure 11.38.

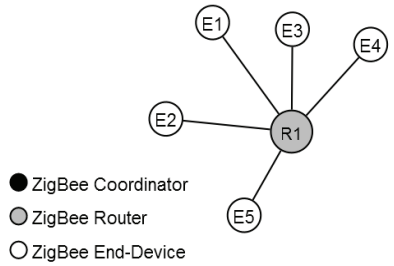

Figure 11.31: ZigBee star topological entity 1, TOP_{S1}.

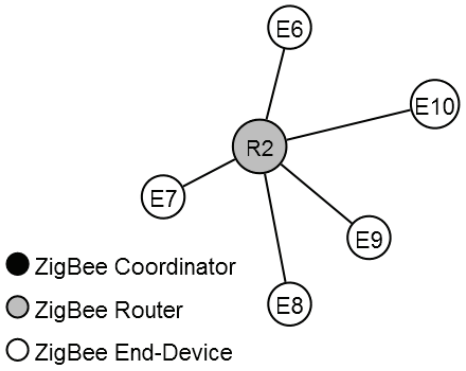

Figure 11.32: ZigBee star topological entity 2, TOP_{S2}.

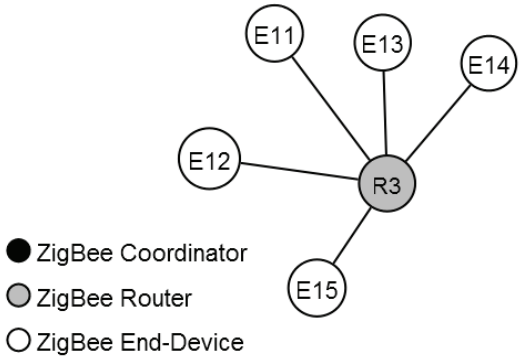

Figure 11.33: ZigBee star topological entity 3, TOP_{S3}.

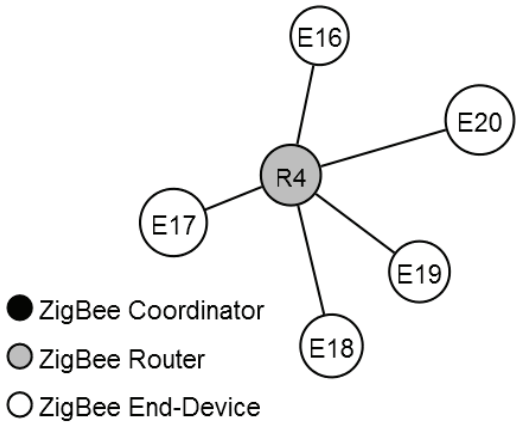

Figure 11.34: ZigBee star topological entity 4, TOP_{S4}.

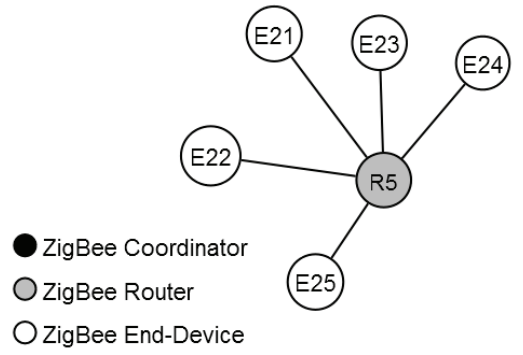

Figure 11.35: ZigBee star topological entity 5, TOP_{S5}.

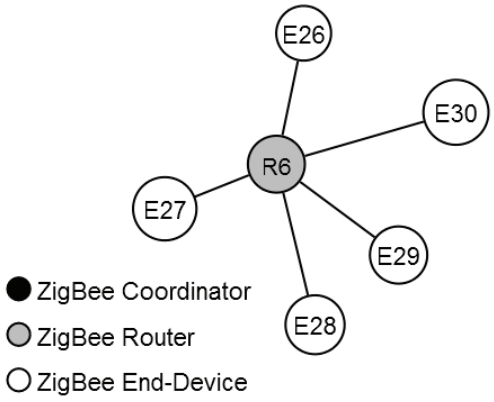

Figure 11.36: ZigBee star topological entity 6, TOP_{S6}.

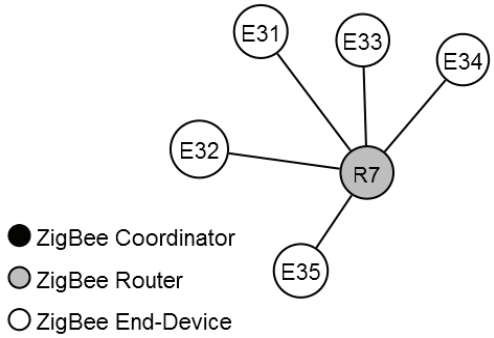

Figure 11.37: ZigBee star topological entity 7, TOP_{S7}.

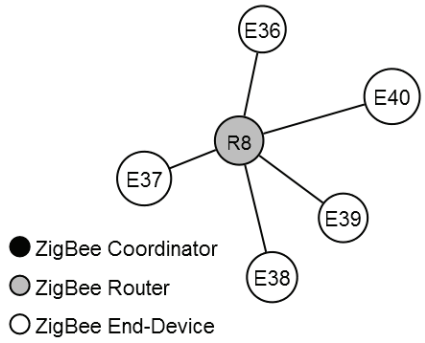

Figure 11.38: ZigBee star topological entity 8, TOP_{S8}.

With the ZigBee tree fundamental topological entity defined, existing algorithms can be used to search and find the four ZigBee tree topological entities contained within the basic communication graph of the quadruple size example network as shown in Figure 11.26. These algorithms form the basis for CAD tools for integrated circuit design [46]. The order (number of Markov processes) of the network is reduced from 52 base level entities to 4 ZigBee tree topological entities as shown in Figure 11.39.

The interaction between the four ZigBee tree topological entities resembles the fully connected mesh topology. The four ZigBee tree topological entities can be combined into a single topological entity, the ZigBee multi-tree topological entity. The ZigBee multi-tree topological entity must perform the same tasks as the ZigBee tree topological entity and the Markov process for the ZigBee multi-tree topological entity and ZigBee tree topological entity are identical. The Markov process for the ZigBee multi-tree topological entity is shown in Figure 11.40. The probability and reward matrices have the following form.

$$P = \begin{bmatrix} P_{II} & P_{IR} & P_{IT} \\ P_{RI} & 0 & 0 \\ P_{TI} & 0 & 0 \end{bmatrix} \tag{11-123}$$

$$R = \begin{bmatrix} R_{II} & R_{IR} & R_{IT} \\ R_{RI} & 0 & 0 \\ R_{TI} & 0 & 0 \end{bmatrix} \tag{11-124}$$

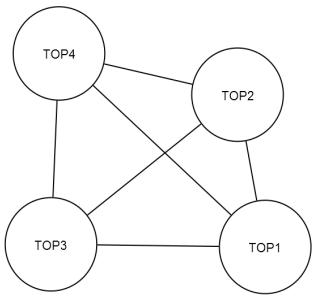

Figure 11.39: Covering of the quadruple size network with the four ZigBee tree topological entities.

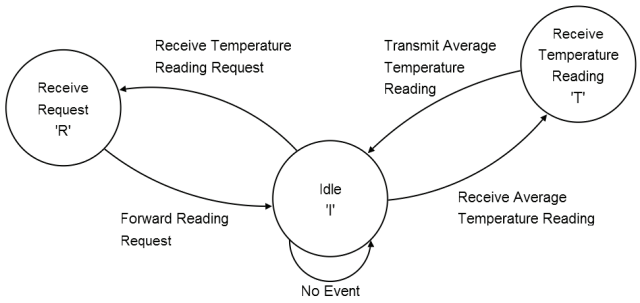

Figure 11.40: Markov process describing the ZigBee coordinator tree topological entity.

Each of the ZigBee tree topological entities that are contained in the ZigBee multi-tree topological entity contribute to the operation of the ZigBee multi-tree topological entity. Thus, the probability matrix for the ZigBee multi-tree topological entity is obtained by adding the probability matrices of each of the ZigBee tree topological entities contained in the ZigBee multi-tree topological entity together, and then dividing the sum by the number of ZigBee tree topological entities contained in the ZigBee multi-tree topological entity. Because each task represents multiple ZigBee tree topological entities performing that task the rewards matrix for the ZigBee multi-tree topological entity is simply the sum of the rewards matrices of each of the ZigBee tree topological entities contained in the ZigBee multi-tree topological entity. With the ZigBee multi-tree fundamental topological entity defined existing algorithms can be used to search and find the single ZigBee multi-tree topological entities contained within the basic communication graph of the quadruple size example network as shown in Figure 11.39. These algorithms form the basis for CAD tools for integrated circuit design [46]. The order (number of Markov processes) of the network is reduced from 4 ZigBee tree topological entities to a single ZigBee multi-tree topological entities as shown in Figure 11.41.

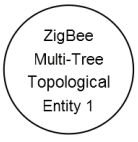

Figure 11.41: Covering of the network with a single ZigBee multi-tree topological entity.

The parameters of all temperature sensors are identical to those used for the used for the temperature sensors in Section 11.1.4. The parameters for the odd number routers (R1, R3, R5, and R7) are identical to those used for router R1 in Section 11.1.5. The parameters for the even number routers (R2, R4, R6, and R8) are identical to those used for router R2 in Section 11.1.5. The parameters for the four ZigBee coordinators are identical to the parameters used for the ZigBee coordinator in Section 11.1.6. The parameters for the odd numbered ZigBee star topological entities (TOP_{S1}, TOP_{S3}, TOP_{S5}, and TOP_{S7}) are identical to those used for the ZigBee star topological entity TOP_{S1} in Section 11.1.10. Likewise, the parameters for the even numbered ZigBee star topological entities (TOP_{S2}, TOP_{S4}, TOP_{S6}, and TOP_{S8}) are identical to those used for the ZigBee star topological entity TOP_{S2} in Section 11.1.10. The parameters used for the four ZigBee tree topological entities are identical to those used for the ZigBee tree topological entity in Section 11.1.15.

The time to evaluate the model using the base level entities and the model using the tree topological entities was determined. Each model was executed 100 times in Matlab and the total time for all 100 runs was obtained, dividing this time by 100, gives the average time for the model to be evaluated once. Each model was run 100 times in order to remove influences such as start-up times and context switches in the computing environment. The reported energy consumption of each network for a period of 1 day is also reported and used to verify the accuracy of the tree topological entity model with the base level entity model for each case. The time to evaluate the single size example networks 100 times and the average time for evaluation are listed in Table 11.11. The time to evaluate the double size example networks 100 times and the average time for evaluation are listed in Table 11.12. The time to evaluate the quadruple size example networks 100 times and the average time for evaluation are listed in Table 11.13.

Table 11.11: Time to evaluate single size example networks.

Entity	Time for 100 Evaluations (seconds)	Average time for 1 Evaluation (seconds)
Tree Topological Entity T1	0.076856 seconds	0.000769 seconds
Base Level Entities	0.446643 seconds	0.004466 seconds

Table 11.12: Time to evaluate double size example networks.

Entity	Time for 100 Evaluations (seconds)	Average time for 1 Evaluation (seconds)
Multi-Tree Topological Entity	0.175654 seconds	0.001757 seconds
Base Level Entities	0.779505 seconds	0.007795 seconds

Table 11.13: Time to evaluate quadruple size example networks.

Entity	Time for 100 Evaluations (seconds)	Average time for 1 Evaluation (seconds)
Multi-Tree Topological Entity	0.185993 seconds	0.001860 seconds
Base Level Entities	1.319124 seconds	0.013191 seconds

The time to evaluate each of the three different size models scales better than linearly with the size of the network. For example, determining the energy consumed by the single size example network using the tree topological entity requires 0.8 ms (milliseconds), and the evaluation of the energy consumption of the quadruple size example network requires 1.9 ms (milliseconds). If the execution time of the model scales linearly with the size of the network, the quadruple size example should take four times as much time as the single size example, in this case 3.2 ms (milliseconds). The time required to evaluate the quadruple size network is 2.375 times the time required to evaluate the single size network. Hence, the execution time of the method verses the size of the network clearly scales better than linearly in relation to network size.

The energy consumed by the single, double, and quadruple size example networks as well as the percent difference, D, between the base level and tree topological entity models for each of the three cases is shown in Table 11.14.

The percent differences between the model using the base level entities and the model using the single tree topological entity are within 1 % for the three different size example networks.

Table 11.14: Energy consumption and percent difference between models for single, double, and quadruple size example networks.

Size	Entity	Energy Consumed (mJ)	Percent Difference (%)
Single	Base Level Entities	151006451.54 mJ	0.0865 %
	Tree Topological Entity T1	150875856.85 mJ	
Double	Base Level Entities	302012903.08 mJ	0.0865 %
	Tree Topological Entity T1	301751713.69 mJ	
Quadruple	Base Level Entities	604025806.15 mJ	0.0865 %
	Tree Topological Entity T1	603503427.38 mJ	

11.5 BURST SWITCH RECEIVER

All three components in the example ZigBee network required an active receiver to receive requests and readings. The active receiver consumes approximately, 99.9 mW (milliWatts) when active, compared to only 13.5 μW (microwatts) when asleep [61]. The components in the example network spend the majority of the time in the idle state, listening for an incoming message. The burst switch, developed at the University of Pittsburgh RFID Center of Excellence, offers an alternative to the active receiver. The burst switch detects the presence of a message rather than the actual data in the message (1s and 0s). The burst switch is a passive receiver that is coupled with a low power processor and monitors for the presence of a message.

Once the presence of a message is detected, the processor takes whatever action necessary to react to the message. A prototype burst switch has been developed and technically consumes only 6 nA when monitoring for a message, and 100 μA (microamps) when processing a message [67].

Both the routers and coordinator must receive readings and require the data in the message. The burst switch could be used to wake up the active receiver in the routers and coordinator once the presence of the preamble has been detected. The preamble consists of 4 bytes and at the specified ZigBee data rate of 250 kbps, the preamble is transmitted for 1024 μs (microseconds) which is long enough for the device to detect the preamble and wake-up the active receiver. However, the transceiver used in the devices in this example decodes the packet internally and thus needs the preamble to detect and decode messages [66]. Hence, the burst switch is not a viable alternative for the router or the coordinator when using the MC13192 transceiver. The temperature sensors only receive the command for a temperature request and this command is distinguished from a temperature reading by the length of the message. Therefore, only the transmitter portion of the transceiver will be used in the temperature sensor. Thus, the burst switch will be added to the temperature sensor only.

The energy consumed by the ZigBee tree topological entity described in Section 11.1.13 through Section 11.1.15 will is calculated when the active receivers on the temperature sensors are replaced with the burst switch receiver. The process to calculate the energy consumption is the same as described in Section 11.1.13 through Section 11.1.15. By using the burst switch instead of the active receiver in the temperature sensors, the energy consumed by the temperature sensors will decrease. This situation results in a longer lifetime for the temperature sensors using the same battery. The power consumption values for the temperature sensor that are changed as a result of using the burst switch are listed in Table 11.15. The other parameters for the ZigBee tree topological entity are identical to those used in the analysis in Section 11.1.15. The temperature sensor still must receive the message for the same amount of time as when using the active receiver to be able to distinguish between a request and a reading.

The energy consumed by the network over 1 day, calculated using the base level entities model and the tree topological model with the temperature sensors employing the burst switch is listed in Table 11.16.

Table 11.15: Power consumption of the temperature sensor with the burst switch.

Parameter	Value
$PW_{PR\text{-}T}$	17.85 mW

Table 11.16: Energy consumption of the network with the temperature sensors employing the burst switch calculated using the base level entities and the tree topological entity.

Model	Energy Consumption (mJ)
Base Level Entities	65941530.28 mJ
Tree Topological Entity	65849528.15 mJ

The percent difference, D, between the energy consumption calculated from the model where the Markov process for each base level entity was evaluated, $E_{Base\text{-}Level}$, and the model using the tree topological entity, $E_{Tree\text{-}Topology}$, is calculated using,

$$D = \frac{\left| E_{\text{Base-Level}} - E_{\text{Tree-Topology}} \right|}{E_{\text{Base-Level}}} \tag{11-125}$$

The difference in the reported energy consumption of the two models, D, is 0.1395 %. This result indicates that the model employing the tree topological entity is accurate.

12.0 ANALYSIS OF THE METHOD

Exploration of design alternatives requires investigating a group of components, investigating the energy consumption of the network in the short term and in the long term, and simulating different topological configurations of the network. The method developed in this work, illustrated in the two examples presented in Sections 10 and 11, enables the designer to quickly and efficiently analyze these three sets of alternatives.

Selecting the best component(s) for an application is difficult and often many components can perform the needed task. In a sensor or RFID network, selecting the component that can perform the task and consume the least amount of energy is critical to maximizing the lifetime of the network. When using a simulator, the network must be resimulated for each of the components and variations. Often tradeoffs are involved and improving one area have a negative affect on performance in another area. Looking at the entire network is the best way to evaluate these tradeoffs. With many different components and variations, this leads to a large number of simulations each of which may take several minutes, hours, or even days. Changing hardware components involves changes to the rewards matrix and possibly to the time matrix (time matrix must be updated only if time for a task changes). Using the method developed in this work, the designer would need to update the appropriate elements of the reward matrix, find the new q-vector, update the time matrix, recalculate t_A, and finally recalculate the energy consumption. The π-vector depends only on the probability matrix, and once it is found, the π-vector can be saved and reused as long as the probability matrix is not changed. If the probability matrix is changed the π-vector must be recalculated. Structural changes to a topological entity normally result in changes to the probability matrix requiring recalculation of the π-vector.

Recall that the reward matrix, R, has the following form,

$$R = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1N} \\ r_{21} & r_{22} & \cdots & r_{2N} \\ \cdots & \cdots & \cdots & \cdots \\ r_{N1} & r_{N2} & \cdots & r_{NN} \end{bmatrix} \tag{12-1}$$

Both the reward matrix, R, and the probability matrix, P, are N x N matrices, where N is the number of states in the Markov process describing the network. Recall that the q-vector, q, is computed from, P, and R.

$$q_k = \sum_{j=1}^{N} \left(p_{kj} * r_{kj} \right) \tag{12-2}$$

$$q = \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_N \end{bmatrix} = \begin{bmatrix} \sum_{j=1}^{N} \left(p_{1j} * r_{1j} \right) \\ \sum_{j=1}^{N} \left(p_{2j} * r_{2j} \right) \\ \vdots \\ \sum_{j=1}^{N} \left(p_{Nj} * r_{Nj} \right) \end{bmatrix} \tag{12-3}$$

Assuming that no changes are made to, P, (therefore π-vector does not have to be recalculated) and that element, r_{jk}, of the reward matrix is changed, then only the value of the element, q_k, of the q-vector must be updated. The example of the simple temperature sensor from Section 4.3 will be used to demonstrate the recalculation of the q-vector. Recall the example of the temperature sensor discussed in Section 4.3, the Markov process describing the simple temperature sensor is shown in the following figure. The probability and rewards matrices, and the π-vector (from the example in Section 4.3) are repeated for clarity,

$$P = \begin{bmatrix} P_{WTWT} & P_{WTRD} & P_{WTTX} \\ P_{RDWT} & P_{RDRD} & P_{RDTX} \\ P_{TXWT} & P_{TXRD} & P_{TXTX} \end{bmatrix} = \begin{bmatrix} 0.70 & 0.29 & 0.01 \\ 0.20 & 0.20 & 0.60 \\ 0.70 & 0 & 0.30 \end{bmatrix} \tag{12-4}$$

$$R = \begin{bmatrix} R_{WTWT} & R_{WTRD} & R_{WTTX} \\ R_{RDWT} & R_{RDRD} & R_{RDTX} \\ R_{TXWT} & R_{TXRD} & R_{TXTX} \end{bmatrix} = \begin{bmatrix} 2 & 6 & 3 \\ 5 & 5 & 17 \\ 4 & 0 & 17 \end{bmatrix} \tag{12-5}$$

$$\pi = \begin{bmatrix} 0.5926 & 0.2148 & 0.1926 \end{bmatrix} \tag{12-6}$$

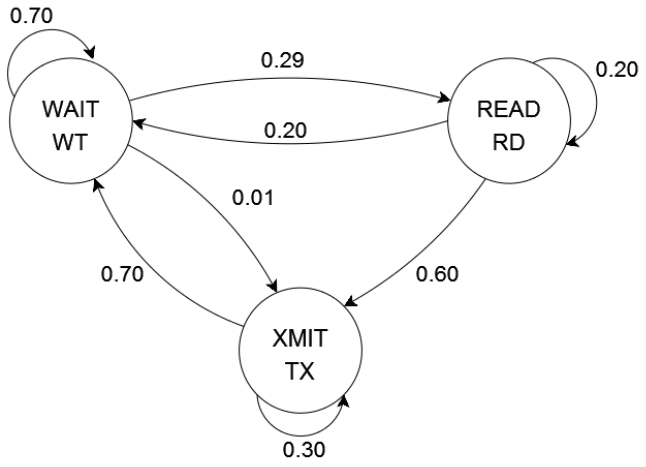

Figure 12.1: State diagram of the Markov process for the simple temperature sensor.

The q-vector found in Section 4.3 for the probability and reward matrices above is,

$$q = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix} = \begin{bmatrix} 3.17 \\ 12.20 \\ 7.90 \end{bmatrix} \tag{12-7}$$

For example, if a different temperature sensor is used that consumes more energy to take a reader then the rewards matrix must be updated to reflect this. Assuming that the new temperature sensor only affects the RD to RD transition, then only element, R_{RDRD} (row 2 column 2), of the reward matrix must be changed. For example, assume this change requires changing, R_{RDRD}, from 5 to 8. The new rewards matrix, R', is shown below (with the modified element in bold),

$$R' = \begin{bmatrix} R_{WTWT} & R_{WTRD} & R_{WTTX} \\ R_{RDWT} & \mathbf{R_{RDRD}} & R_{RDTX} \\ R_{TXWT} & R_{TXRD} & R_{TXTX} \end{bmatrix} = \begin{bmatrix} 2 & 6 & 3 \\ 5 & \mathbf{8} & 17 \\ 4 & 0 & 17 \end{bmatrix} \tag{12-8}$$

The entire q-vector, q', is recalculated to illustrate that only element, q'_2, changes and that only element, q'_2, really needs to be recalculated.

$$q' = \begin{bmatrix} q'_1 \\ q'_2 \\ q'_3 \end{bmatrix} = \begin{bmatrix} 3.17 \\ 12.80 \\ 7.90 \end{bmatrix} \tag{12-9}$$

Likewise, if two of the entries on row 2 of R (row 2 columns 2 and 3) are changed, shown in R'', (altered values are in bold) q''_2, is still the only affected element of the q-vector, q'' and this is shown below.

$$R'' = \begin{bmatrix} R_{WTWT} & R_{WTRD} & R_{WTTX} \\ R_{RDWT} & \mathbf{R_{RDRD}} & \mathbf{R_{RDTX}} \\ R_{TXWT} & R_{TXRD} & R_{TXTX} \end{bmatrix} = \begin{bmatrix} 2 & 6 & 3 \\ 5 & \mathbf{8} & \mathbf{20} \\ 4 & 0 & 17 \end{bmatrix} \tag{12-10}$$

The new q-vector, q'', is below, note that only element q''_2 of the q-vector is changed.

$$q'' = \begin{bmatrix} q''_1 \\ q''_2 \\ q''_3 \end{bmatrix} = \begin{bmatrix} 3.17 \\ 14.60 \\ 7.90 \end{bmatrix} \tag{12-11}$$

Again, only the value of element, q''_2, changes. Changing multiple elements on the same row, row j, of the reward matrix still alters only one value in the q-vector, q_k. Thus, for each row of the reward matrix that is changed, N, scalar multiplications and, N, scalar additions are required to find the updated, q_k. Changing the values of the elements on M different rows of the reward matrix results in the alteration of M of the elements in the q-vector. Thus, if M rows of the rewards matrix are modified, M elements in the q-vector (corresponding to those M rows of the rewards matrix) must be recomputed. If, M, rows are changed in the reward matrix a total of

M*N scalar multiplications, and M*N scalar additions are required. With the new q-vector and the previously obtained π-vector, the gain, g, is obtained,

$$g = \pi q \tag{12-12}$$

Calculating the gain requires multiplying the π-vector, which is a row vector, with the q-vector, a column vector, with each vector having length N. This results in the dot product of two vectors. The dot product of two vectors, of length N, requires N scalar multiplications and N scalar additions. Finally, the gain, g, is multiplied by the number of transitions to obtain the energy consumption for the network for a specific time. This step requires a single scalar multiplication.

Hence, if values in M rows of the reward matrix are changed and the probability matrix are unchanged, the total number of operations to obtain the energy consumed by the new network requires (M*N + N + 1) scalar multiplications and (M*N + N) scalar additions. These operations can be computed very quickly on a modern processor. A simulator would normally require that the entire network be simulated again with the new values requiring at least as long as the original simulation run (most likely more than one second). Therefore, the method described in this work allows the designer to quickly and efficiently evaluate many different components in order to select the component that minimized overall energy consumption.

Recall that the time matrix described in Section 4.3 and the probability matrix were used to determine the average time for a single transition, t_A, in the Markov process. This average time, t_A, was then used to determine the value of n. The time matrix, T_A, is shown below,

$$T_A = \begin{bmatrix} t_{11} & t_{12} & \dots & t_{1N} \\ t_{21} & t_{22} & \dots & t_{2N} \\ \dots & \dots & \dots & \dots \\ t_{N1} & t_{N2} & \dots & t_{NN} \end{bmatrix} \tag{12-13}$$

Recall that each entry, t_{ij}, in, T_A, represents the average time of the operation of all entities performing a task for the i to j transition. The average time per transition, t_A, should be weighted to those transitions that happen more frequently to obtain better accuracy in the

solution. To appropriately weight, t_A, to those transitions the average time per transition q-vector, q_A, must be found,

$$q_A = \begin{bmatrix} q_{A1} \\ q_{A2} \\ \vdots \\ q_{AN} \end{bmatrix} = \begin{bmatrix} \sum_{j=1}^{N} (p_{1j} * t_{1j}) \\ \sum_{j=1}^{N} (p_{2j} * t_{2j}) \\ \vdots \\ \sum_{j=1}^{N} (p_{Nj} * t_{Nj}) \end{bmatrix} \tag{12-14}$$

With, q_A, the weighted average time per transition, t_A, is simply,

$$t_A = \pi q_A \tag{12-15}$$

Once, t_A, is calculated the number of transitions is found by simply dividing the desired length of time to investigate, L_{Time}, by, t_A,

$$n = \frac{(L_{Time})}{t_A} \tag{12-16}$$

If any of the times for the subtasks are altered the time matrix, T_A, should be recalculated. Using the new, T_A, the q-vector, q_A, can be recomputed and then the weighted average time per transition, t_A, can be found. Recalculating t_A is advisable as the value of t_A will affect the accuracy of the energy consumption obtained from the model.

Recalculating the time matrix, T_A, may require several additions and one division to compute the average time for all subtasks represented in a given transition. If, M, rows are changed in the time matrix, T_A, a total of M*N scalar multiplications, and M*N scalar additions are required to recalculate, q_A. Calculating t_A, from q_A (it is assumed that the π-vector is unchanged) requires taking the dot product of the row vector, π, and the column vector, q_A, each having length N. This requires N scalar multiplications and N scalar additions. Therefore, the total number of operations, excluding the operations necessary to recalculate T_A, is (M*N + N)

scalar multiplications and (M*N + N) scalar additions. One scalar division is required to compute n.

Often, designers find it useful to investigate the energy consumption of a network for different lengths of time. In this method, once the gain, g, has been calculated, the energy consumed by the network for different lengths of time is easily calculated by changing, n, to the appropriate value using (12-16). Once, the appropriate value for n has been found, a single scalar multiplication is required to find the energy consumed by the network for that period of time. Finding the value of, n, usually requires only one scalar division and a handful of scalar multiplications to put the time terms in the numerator (L_{Time}) and denominator (t_A) in the same units. The value of, n, can be found by using (12-16) with the appropriate value for L_{Time} and t_A.

Scalar multiplications may be needed to convert L_{Time} into the proper units of time (i.e. milli-seconds, seconds, micro-seconds). The quantities, L_{Time}, and t_A, must be in the same time units. A single scalar division is required to evaluate (12-16). The example in Section 4.3 determined the energy consumption for the example simple temperature sensor for periods of 1-day, 1-month, 1-year, and 10-years by simply changing the value of, L_{Time}, to the appropriate value for the time in question and recalculating, n, using (12-16).

This results in a faster exploration of the design alternatives than current simulators allow. With the information obtained from the exploration the designer can choose those components that optimize the performance of the sensor or RFID network in question with respect to minimizing energy consumption.

13.0 CONCLUSIONS AND FUTURE WORK

13.1 REVIEW

Current sensor and RFID networks contain large numbers of entities, while future sensor and RFID networks are projected to contain many hundreds or thousands or millions of entities. An on-board battery usually powers the sensors and RFID devices in these networks. Hence, the energy consumption of the network as a whole must be minimized to maximize the lifetime of the network. An efficient and reliable method of determining the energy consumption of the entire network is critical because optimizations to one entity may result in an increase of the energy consumption of several other entities, ultimately increasing the energy consumption of the entire network. Several factors contribute to energy consumption, and these factors are often interrelated so tradeoffs must be evaluated. Evaluating these tradeoffs requires the evaluation of a large number of design alternatives.

Current methods to evaluate the energy consumption of the network require the simulation of all entities within the network. This results in simulations having large numbers of entities passing very large numbers of messages among themselves. In a discrete event simulation the main bottleneck is in determining which messages are safe to process, and to which simulation entities those messages must be passed [4]. As the size of the network grows, more messages are sent between entities, increasing the execution time of the simulation. The resulting long runtime prevents designers from exploring multiple design alternatives.

13.2 THE RESEARCH

The method developed in this research provides for order reduction (in terms of the number of entities in the network) and facilitates a reduction of the number of intermediate entities that must be evaluated. Entities within a sensor or RFID network implicitly group themselves together into topological groups (topologies) to perform the tasks required by the network. By identifying these topological groups and providing an analytical methodology, it is possible to replace all entities within a given topological group with a single reduced topological unit. This reduces the order (number of entities in the network) significantly, and through repeated application, these groupings can reduce the number of entities within the network to a handful of topological units, ideally, to a single topological entity. An algorithm has been developed in this research to identify the topological entities in a given network. This algorithm is presented in Section 9.

A modeling framework was developed using Markov processes with rewards to determine the energy consumption of the entire network and was presented in Section 4. In order to keep the dimension of the Markov processes describing the energy consumption of the topological entities within a reasonable degree, the concept of tasks was developed. Each topology must perform a set of specific tasks (i.e. take a reading or process a message). Identifying these tasks makes it convenient to model the behavior of a topological entity using a Markov process while keeping the dimensionality within limits. The Markov process contains one state per task, plus one state for the idle state. If a task is very complex, it can be broken into multiple states. Tasks also help the designer to identify and determine how best to deploy each topological entity.

A breakdown of sensor and RFID networks into fundamental classes was presented in Section 3 and aids in identifying the tasks a particular network or topological entity must perform. Topological entities perform a number of different tasks, and a set of basis tasks was developed from which any task can be constructed. The Markov process contains a set of states, ideally a single state, for each task that a topological entity must perform. Generic Markov process structures for the base entities and base topological entity were developed and are presented in Section 5. These generic Markov processes provide a starting point from which more complex Markov processes can be developed. The concept of the basic communication

graph, presented in Section 6, was developed to identify interactions between entities and topologies within a network. The distance graph method presented in Section 7.1.2 is possible due to the inherent nature of wireless communication and uses the basic communication graph to identify fundamental and practical topologies in an existing network. The result is in terms of the vector of absolute state probabilities, π.

Once the π-vector has been obtained from the probability matrix, strategic management alternatives involving modification of only the reward matrix and/or the simulated period of time for which the network must be evaluated require only a handful of scalar additions, multiplications, and divisions as long as the probability matrix is not modified. The procedure for evaluating these alternations is presented in Section 12. Alterations to the reward matrix represent changing the power consumed for each task. Some examples of alternatives that require altering only the reward matrix are investigating the energy consumption when different components are used, or investigating the energy consumption using a different communication protocol. One example of investigating different design alternatives is the example presented in Section 11.5 where the active receiver in the temperature sensors in the ZigBee network is replaced with the burst switch receiver to determine the energy savings possible employing the burst switch in place of the active receiver.

Structural changes to the network can be localized to a single topological entity or group of topological entities. Once the topological entities affected by the structural changes are updated, the energy consumed by the entire network can be updated using the new energy consumption values for the affected topologies. Those topological entities unaffected by the structural changes do not need to be reevaluated. The partition trees presented in Section 7 assist in identifying the topological entities affected by the structural change.

The method developed in this research facilitates an order reduction of the number of entities that must be evaluated to determine the energy consumed by a given sensor or RFID network. With the reduced network, the designer can quickly evaluate different design alternatives and select the alternative minimizing the energy consumption of the entire network. The methodology developed in this work is highly scalable as a function of the number of entities simulated. The example cases presented in Section 10 and 11 demonstrate that the execution time required for the example networks scale slower than linearly with the size of the network. The example cases demonstrate that evaluation of the energy consumption of the

networks using the topological entities is accurate, and the energy consumed by the entire networks found using the topological entities is, at worst, within a few percent of the energy consumed by the entire networks found when all base level entities were evaluated. These small differences can be attributed to round-off and combination of constants which reduce the occurrences of mathematical operations.

The execution speed of this method allows the results to be recalculated quickly. This is especially useful when the designer makes a typo or sets a parameter to the incorrect value. In a simulator, errors such as these may not be detected until well into the simulation run (several hours or even days), forcing the designer to correct the errors and then restart the simulation. With the quick evaluation time of this method the time wasted is minimal and mistakes can quickly be found.

The methodology developed in this research provides designers of sensor and RFID networks with a tool to evaluate the energy consumption of the network. Topological entities are used to reduce the order of the network, which results in a faster execution time for determining energy consumption. The tool allows rapid exploration of the design space requiring only a handful of scalar operations (multiplication, addition, and division) to recomputed the energy consumption of the network. The rapid exploration of the design space allows designers to make better, more informed design decisions. This results in the design and creation of better performing sensor and RFID networks.

13.3 FUTURE WORK

The method developed in this research requires the identification of topologies in a sensor or RFID network. For a network that already exists or is already deployed, the designer must work with the basic communication graph for that deployment. The bottom-up method must be used to extract topological entities from the basic communication graph. Algorithms exist to search a graph and identify structures matching the fundamental topological entity structures. Integrated circuit design CAD tools utilize these algorithms to search a designer's circuit for structures matching standard cells contained in a technology library, this is called 'technology mapping'

[46]. No such automated CAD tool exists specifically for use with the method described in this work. Such an automated tool could be created using two different approaches.

The first method is to modify the basic communication graph structure to match the input structure of an integrated circuit CAD tool that performs technology mapping. The fundamental topological entity structures would also need to be converted into a technology library for use with the CAD tool. This approach reuses the existing CAD tool but requires construction of two translator programs. The first translator program would convert the sensor or RFID network from a basic communication graph format to a format recognized by the integrated circuit CAD tool. The second translation program would convert the fundamental topological structures from a basic communication graph representation to a technology library format that is understood by the integrated circuit CAD tool.

The second approach to automating the extraction of topological entities in a network is to implement the existing algorithms for use on the basic communication graph. This approach does not require the two translation steps that the other approach requires, but does require the implementation of the algorithm to search and extract the topological entities from the basic communication graph.

This research focused on calculating energy consumed by a sensor or RFID network. In the case of a passive RFID network, the RFID tags are powered by the reader's inquiry signal and do not contain an on-board battery. In a passive RFID network, energy consumption is not as important as in an active RFID network (ISO 18000-7) or a sensor network. However, read rate, defined here, as the number of successful reads of a RFID tag out of 100 attempted reads, is an important metric. The primary application of passive RFID networks to date is to track inventory, hence a read rate as close to 100% is desired to minimize missing any items in an inventory.

Electromagnetic CAD packages such as Ansoft and Sonnet provide accurate results, but require a significant amount of computing power and time to evaluate a complex simulation. The method developed in this work used topological entities to reduce the order of sensor and RFID networks by grouping base and topological entities together to form larger topological entities. This same method can be applied to a passive RFID network by redefining the base entities to consist of environmental entities as well as readers and tags. Examples of environmental entities could be RFID tags placed on metal or shampoo. Both of these materials

interfere with RFID. An obstacle such as water or a wall between the RFID reader and the RFID tags is another example of an environmental entity.

A survey of typical environments in which passive RFID networks are deployed would reveal fundamental environmental entities, from which fundamental topological entities would be constructed. Existing CAD tools (Ansoft and Sonnet) would then be used to determine the effect of the environmental entities on the read rate for each fundamental topological entity. The fundamental topological entity simulations would be able to be kept small to reduce the computing power and time needed for the CAD tools (Ansoft and Sonnet) to complete the simulation. With this data and the fundamental topological entities, the read rate for large passive RFID networks could be quickly and easily obtained, just as the energy consumption for sensor and RFID networks was in this work.

The methodology developed in this research could be extended in the future to networks containing mobile entities. The methodology developed in this research may be applicable to reduce the time required for the evaluation of a power grid. By grouping together components of the power grid into topological entities, the number of entities that must be evaluated is reduced and should reduce the execution time. Chip designs are often modeled as state machines and chips interact with other chips and simulation of these designs is complicated and requires a great deal of time. Thus, the behavior of each chip can easily be expressed as a Markov process. Topological entities can be identified based on the interconnections. Markov processes describing the behavior of these topological entities should be able to be constructed. The methodology developed in this research may be able to be used to reduce the number of devices that must be simulated and decrease the time required to analyze the system. Further, the methodology may be applicable to the system on a chip (SOC) design process where topological entities could represent individual IP blocks that are combined to form the system. Lastly, the sizes of the cache and memory bus are important in the performance of a processor. Markov processes can be used to model the cache and memory interface with a single processor to determine the optimal size for the cache and memory bus. In a parallel computer multiple processors are linked together. If topological entities can be identified and Markov processes can be formulated to describe the behavior of the processors, caches, and memories the methodology can be used to reduce the number of Markov processes that must be evaluated to analyze processor, cache, and memory interaction.

REFERENCES NOT CITED

[1] J. Gehrke, S. Madden, "Query processing in sensor networks," *IEEE Pervasive Computing,* , vol. 3, no. 1, Jan.-Mar. 2004, pp. 46- 55.

[2] Y. Ma and J. H. Aylor, "System lifetime optimization for heterogeneous sensor networks with a hub-spoke technology," Mobile Computing, IEEE Transactions on, vol. 3, pp. 286-294, 2004.

[3] C. Schurgers, V. Tsiatsis, S. Ganeriwal, M. Srivastava, "Optimizing sensor networks in the energy-latency-density design space," *IEEE Transactions on Mobile Computing,* vol. 1, no. 1, Jan.-Mar. 2002, pp. 70- 80.

[4] S. Ci, H. Sharif, and K. Nuli, "Study of an adaptive frame size predictor to enhance energy conservation in wireless sensor networks," Selected Areas in Communications, IEEE Journal on, vol. 23, pp. 283-292, 2005.

[5] A. Cerpa, D. Estrin, "ASCENT: adaptive self-configuring sensor networks topologies," *IEEE Transactions on Mobile Computing,* vol. 3, no. 3, July-Aug. 2004, pp. 272- 285.

[6] S. McGirr, K. Raysin, C. Ivancic, C. Alspaugh, "Simulation of underwater sensor networks," *OCEANS '99 MTS/IEEE Riding the Crest into the 21st Century*, vol. 2, 1999, pp. 945- 950.

[7] K. Sohrabi, J. Gao, V. Ailawadhi, and G. J. Pottie, "Protocols for self-organization of a wireless sensor network," Personal Communications, IEEE [see also IEEE Wireless Communications], vol. 7, pp. 16-27, 2000.

[8] S.M. Das, H. Pucha, Y.C. Hu, "MicroRouting: A Scalable and Robust Communication Paradigm for Sparse Ad Hoc Networks," *Proc. 19th IEEE Int'l Parallel and Distributed Processing Symposium,* Apr. 2005, pp. 245b- 245b.

[9] S.D. Muruganathan, D.C.F. Ma, R.I. Bhasin, A.O. Fapojuwo, "A centralized energy-efficient routing protocol for wireless sensor networks," *IEEE Communications Magazine,* vol. 43, no. 3, Mar. 2005, pp. S8- S13.

[10] M.J. Miller, N.H. Vaidya, "A MAC protocol to reduce sensor network energy consumption using a wakeup radio," *IEEE Transactions on Mobile Computing,* vol. 4, no. 3, May-June 2005, pp. 228- 242.

[11] M. Hempel, H. Sharif, P. Raviraj, "HEAR-SN: A New Hierarchical Energy-Aware Routing Protocol for Sensor Networks," *Proc. of the 38th Annual Hawaii Int'l Conf. on System Sciences* (HICSS '05), Jan. 2005, pp. 324a- 324a.

[12] N. Thepvilojanapong, Y. Tobe, K. Sezaki, "HAR: hierarchy-based anycast routing protocol for wireless sensor networks," *Proc. of The 2005 Symposium on Applications and the Internet,* Jan.- Feb. 2005, pp. 204- 212.

[13] C. Jaikaeo, C. Srisathapornphat, C. C. Shen, "Diagnosis of sensor networks," *IEEE Int'l Conf. on Communications* (ICC 2001), vol.5, 2001, pp. 1627- 1632.

[14] X. Hong, M. Gerla, R. Bagrodia, T. J. Kwon, P. Estabrook, and P. Guangyu, "The Mars sensor network: efficient, energy aware communications," Military Communications Conf. Communications for Network-Centric Operations: Creating the Information Force, (MILCOM 2001), 2001.

[15] M.A. Youssef, M.F. Younis, K.A. Arisha, "A constrained shortest-path energy-aware routing algorithm for wireless sensor networks," *IEEE Wireless Communications and Networking Conference* (WCNC2002), vol. 2, Mar. 2002, pp. 794- 799.

[16] S. Cho and A. P. Chandrakasan, "Energy efficient protocols for low duty cycle wireless microsensor networks," Proc. IEEE Int'l Conf. on Acoustics, Speech, and Signal Processing (ICASSP '01), 2001.

[17] M. Younis, M. Youssef, K. Arisha, "Energy-aware routing in cluster-based sensor networks," *Proc. 10th IEEE Int'l Symposium on Modeling, Analysis and Simulation of Computer and Telecommunications Systems,* (MASCOTS 2002), 2002, pp. 129- 136.

[18] V. Srinivasan, P. Nuggehalli, R. Rao, "Design of optimal energy aware protocols for wireless sensor networks," *IEEE VTS 53rd Vehicular Technology Conf.* (VTC 2001), vol. 4, 2001, pp. 2494- 2498.

[19] R. Iyer, L. Kleinrock, "QoS control for sensor networks," *IEEE International Conf. on Communications* (ICC '03), vol. 1, May 2003, pp. 517- 521.

[20] R. S. Bhuvaneswaran, J. L. Bordim, J. Cui, N. Ishii, and K. Nakano, "An energy-efficient initialization protocol for wireless sensor networks," Int'l Conf. on Parallel Processing Workshops, 2001.

[21] A. Sobeih, W.-P. Chen, J. C. Hou, L.-C. Kung, N. Li, H. Lim, H.-Y. Tyan, and H. Zhang, "J-Sim: a simulation environment for wireless sensor networks," Proc. 38th Annual Simulation Symposium, 2005.

[22] S. Park, A. Savvides, and M. B. Srivastava, "Simulating networks of wireless sensors," Proc. of the Winter Simulation Conf., 2001.

[23] P. Baldwin, S. Kohli, E. A. Lee, X. Liu, and Y. Zhao, "Modeling of sensor nets in Ptolemy II," Proc. of the 3rd Int'l Symposium on Information Processing in Sensor Networks (IPSN'04), 2004.

[24] L. Girod, T. Stathopoulos, N. Ramanathan, J. Elson, D. Estrin, E. Osterweil, T. Schoellhammer, "A system for simulation, emulation, and deployment of heterogeneous sensor networks," *Proc. of the 2nd Int'l Conf. on Embedded Networked Sensor Systems* (SenSys '04), Nov. 2004, pp. 201- 213.

[25] B. White, J. Lepreau, S. Guruprasad, "Lowering the barrier to wireless and mobile experimentation," *SIGCOMM Computer Communication Review* vol. 33, no. 1, Jan. 2003, pp. 47- 52.

[26] X. Chang, "Network simulations with OPNET," Proceedings of the 1999 Winter Simulation Conf., 1999.

[27] S. Park, A. Savvides, M.B. Srivastava, "SensorSim: a simulation framework for sensor networks," *Proc. of the 3rd ACM Int'l Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems* (MSWIM '00), Aug. 2000, pp. 104- 111.

[28] H. Y. Song, "A probabilistic performance model for conservative simulation protocol," Proc. 15th Workshop on Parallel and Distributed Simulation, 2001.

REFERENCES

[1] E. H. Callaway, Jr., Wireless Sensor Networks Architectures and Protocols: Auerbach Publications, 2004.

[2] C. Chee-Yee and S. P. Kumar, "Sensor networks: evolution, opportunities, and challenges," Proceedings of the IEEE, vol. 91, pp. 1247-1256, 2003.

[3] R. Szewczyk, E. Osterweil, J. Polastre, M. Hamilton, A. Mainwaring, and D. Estrin, "Habitat monitoring with sensor networks " Communications of the ACM, vol. 47, pp. 34-40, 2004.

[4] F. Zhao and L. Guibas, Wireless Sensor Networks An Information Processing Approach: Morgan Kaufmann Publishers, 2004.

[5] H. Tim Tau, "Using sensor networks for highway and traffic applications," Potentials, IEEE, vol. 23, pp. 13-16, 2004.

[6] K. Sohrabi and G. J. Pottie, "Performance of a novel self-organization protocol for wireless ad-hoc sensor networks," presented at Vehicular Technology Conf. , 1999.

[7] F. Mondinelli and Z. M. Kovacs-Vajna, "Self-localizing sensor network architectures," Instrumentation and Measurement, IEEE Transactions on, vol. 53, pp. 277-283, 2004.

[8] S. Slijepcevic and M. Potkonjak, "Power efficient organization of wireless sensor networks," presented at IEEE Int'l Conf. on Communications (ICC 2001), 2001.

[9] J. Qiangfeng and D. Manivannan, "Routing protocols for sensor networks," presented at 1st IEEE Consumer Communications and Networking Conf., 2004.

[10] S. De, Q. Chunming, and W. Hongyi, "Meshed multipath routing: an efficient strategy in sensor networks," presented at IEEE Wireless Communications and Networking (WCNC 2003), 2003.

[11] V. Tsiatsis, S. A. Zimbeck, and M. B. Srivastava, "Architecture strategies for energy-efficient packet forwarding in wireless sensor networks," presented at International Symposium on Low Power Electronics and Design, 2001.

[12] A. Sinha and A. Chandrakasan, "Dynamic power management in wireless sensor networks," Design & Test of Computers, IEEE, vol. 18, pp. 62-74, 2001.

[13] A. Boulis and M. B. Srivastava, "Node-level energy management for sensor networks in the presence of multiple applications," presented at Proc. of the 1st IEEE International Conf. on Pervasive Computing and Communications (PerCom 2003), 2003.

[14] J. Agre and L. Clare, "An integrated architecture for cooperative sensing networks," Computer, vol. 33, pp. 106-108, 2000.

[15] C.-C. Shen, C. Srisathapornphat, and C. Jaikaeo, "Sensor information networking architecture and applications," Personal Communications, IEEE [see also IEEE Wireless Communications], vol. 8, pp. 52-59, 2001.

[16] Z. Feng, S. Jaewon, and J. Reich, "Information-driven dynamic sensor collaboration," Signal Processing Magazine, IEEE, vol. 19, pp. 61-72, 2002.

[17] S. S. Pradhan, J. Kusuma, and K. Ramchandran, "Distributed compression in a dense microsensor network," Signal Processing Magazine, IEEE, vol. 19, pp. 51-60, 2002.

[18] C. Haowen and A. Perrig, "Security and privacy in sensor networks," Computer, vol. 36, pp. 103-105, 2003.

[19] A. D. Joseph, "Energy harvesting projects," Pervasive Computing, IEEE, vol. 4, pp. 69-71, 2005.

[20] R. Want, K. I. Farkas, and C. Narayanaswami, "Guest Editors' Introduction: Energy Harvesting and Conservation," Pervasive Computing, IEEE, vol. 4, pp. 14-17, 2005.

[21] M. Rahimi, H. Shah, G. S. Sukhatme, J. Heideman, and D. Estrin, "Studying the feasibility of energy harvesting in a mobile sensor network," presented at IEEE Int'l Conf. on Robotics and Automation (ICRA '03), 2003.

[22] J. A. Paradiso and T. Starner, "Energy scavenging for mobile and wireless electronics," Pervasive Computing, IEEE, vol. 4, pp. 18-27, 2005.

[23] A. Kansal and M. B. Srivastava, "An environmental energy harvesting framework for sensor networks," 2003.

[24] R. M. Fujimoto, Parallel and distributed simulation systems. New York: John Wiley & Sons Inc., 2000.

[25] G. F. Riley, "Large-scale network simulations with GTNetS," presented at Proceedings of the 2003 Winter Simulation Conference, 2003.

[26] Z. Ji, J. Zhou, M. Takai, and R. Bagrodia, "Scalable simulation of large-scale wireless networks with bounded inaccuracies," in Proceedings of the 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems. Venice, Italy: ACM Press, 2004.

[27] V. Naoumov and T. Gross, "Simulation of large ad hoc networks," in Proceedings of the 6th ACM international workshop on Modeling analysis and simulation of wireless and mobile systems. San Diego, CA, USA: ACM Press, 2003.

[28] H. Akhtar, "An overview of some network modeling, simulation and performance analysis tools," presented at Proc., 2nd IEEE Symposium on Computers and Communications, 1997.

[29] F. Dia and J. Wu, "On Constructing k-Connected k-Dominating Set in Wireless Networks," presented at Proc. 19th IEEE Int'l Parallel and Distributed Processing Symposium (IPDPS 2005), 2005.

[30] R. Min and A. Chandrakasan, "Energy-efficient communication for ad-hoc wireless sensor networks," presented at Conf. Record of the 35th Asilomar Conference on Signals, Systems and Computers, 2001.

[31] "GloMoSim Manual Version 1.2," UCLA Parallel Computing Laboratory http://pcl.cs.ucla.edu/projects/glomosim/GloMoSimManual.html.

[32] X. Zeng, R. Bagrodia, and M. Gerla, "GloMoSim: a library for parallel simulation of large-scale wireless networks," presented at Proc. Twelfth Workshop on Parallel and Distributed Simulation (PADS 98), 1998.

[33] R. Bagrodia, R. Meyer, M. Takai, Y.-A. Chen, X. Zeng, M. Jay, and H. Y. Song, "Parsec: a parallel simulation environment for complex systems," Computer, vol. 31, pp. 77-85, 1998.

[34] D. Cavin, Y. Sasson, and A. Schiper, "On the accuracy of MANET simulators," in Proceedings of the second ACM international workshop on Principles of mobile computing. Toulouse, France: ACM Press, 2002.

[35] C. F. Chiasserini and M. Garetto, "Modeling the performance of wireless sensor networks," presented at 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2004), 2004.

[36] D. L. Isaacson and R. W. Madsen, Markov Chains, Theory and Applications, 1976.

[37] R. A. Howard, Dynamic programming and Markov processes. Cambridge: M.I.T. Press, 1966.

[38] B. L. Fox and D. M. Landi, "An Algorithm for Identifying the Ergodic Subchains and Transient States of a Stochastic Matrix," Communications of the Association for Computing Machinery, vol. 11, pp. 619-621, 1968.

[39] J. G. Kemeny, J. L. Snell, and G. L. Thompson, Introduction to Finite Mathematics, 3rd ed. Englewood Cliffs: Prentice-Hall, 1974.

[40] M. H. Mickle and T. W. Sze, Optimization in Systems Engineering. Scranton: Intext Educational Publishers, 1972.

[41] A. V. Oppenheim, R. W. Schafer, and J. R. Buck, Discrete-Time Signal Processing. Upper Saddle River, NJ: Prentice-Hall, Inc., 1999.

[42] E. Kamen, Introduction to Signals and Systems. New York, NY: Macmillan Publishing Company, 1987.

[43] K. Dantu, M. Rahimi, H. Shah, S. Babel, A. Dhariwal, and G. S. Sukhatme, "Robomote: enabling mobility in sensor networks," 2005.

[44] J. Luo and J. P. Hubaux, "Joint mobility and routing for lifetime elongation in wireless sensor networks," 2005.

[45] W. J. Kaiser, G. J. Pottie, M. B. Srivastava, G. S. Sukhatme, J. Villasenor, and D. Estrin, "Networked Infomechanical Systems (NIMS) for Ambient Intelligence," Technical Report 31, UCLA-NSF Center for Embedded Networked Sensing December 2003.

[46] G. De Micheli, Synthesis and Optimization of Digital Circuits: McGraw-Hill, 1994.

[47] B. A. Forouzan and S. C. Fegan, TCP/IP Protocol Suite. New York, NY: McGraw-Hill, 2003.

[48] W. Stallings, Data & Computer Communications, 6th ed. Upper Saddle River, NJ: Prentice Hall, 2000.

[49] T. H. Cormen, C. E. Lieserson, R. L. Rivest, and C. Stein, Introduction to Algorithms: McGraw-Hill, 2001.

[50] B. Bollobás, Modern Graph Theory. New York, NY: Springer, 1998.

[51] R. E. Neapolitan and K. Naimipour, Fundamentals of Algorithms Using Java Pseudocode: Jones and Bartlett Publishers, Inc., 2004.

[52] D. Geer, "Users Make a Beeline for ZigBee Sensor Technology," Computer, vol. 38, pp. 16-19, 2005.

[53] J. L. Hennessy and D. A. Patterson, Computer Architecture A Quantitative Approach, 3rd ed. San Fancisco, CA: Morgan Kaufmann Publishers, 2003.

[54] Z. Alliance, "ZigBee Specification v1.0," 2005.

[55] K. Finkenzeller, RFID Handbook : Fundamentals and Applications in Contactless Smart Cards and identifcation: Wiley, 2003.

[56] Savi, "Savi SensorTag ST-674," 2005.

[57] Savi, "Savi SensorTag ST-673," 2006.

[58] Savi, "Savi SensorTag ST-676," 2006.

[59] "Information Technology – Radio-frequency Identification for Item Management – Part 7: Parameters for Active Air Interface Communications at 433 MHz," ISO/IEC FDIS 18000-7, 2004.

[60] H. Cho and Y. Baek, "Design and implementation of an active RFID system platform," presented at Applications and the Internet Workshops, 2006. SAINT Workshops 2006. International Symposium on, 2006.

[61] Atmel, "ATmega128(L) Datasheet," vol. 2006, 2006.

[62] Semtech, "XE1203F Datasheet," vol. 2006, 2005.

[63] "ZigBee Alliance," http://www.zigbee.org/.

[64] "IEEE Standard for Information Technology - Telecommunications and Information Exchange Between Systems Local and metropolitan area networks - Specific requirements Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)," IEEE Std 802.15.4-2003, 2003.

[65] National Semiconductor, "LM70 SPI/MICROWIRE 10-Bit plus Sign Digital Temperature Sensor," vol. 2006, 2006.

[66] Freescale Semiconductor, "MC13192/MC13193 Datasheet," vol. 2006, 2005.

[67] P. J. Hawrylak, L. Mats, J. T. Cain, A. K. Jones, S. Tung, and M. H. Mickle, "Ultra-Low Power Computing System for Wireless Devices," International Review on Computers and Software, vol. 1, pp. 1-10, 2006.