2006

Scope selection in wireless sensor network reprogramming through the RF ripples

Peter Chung
Lehigh University

Chung, Peter

Scope Selection in Wireless Sensor Network Reprogramming through the RF Ripples

May 2006

Scope Selection in Wireless Sensor Network Reprogramming through the RF Ripples

By

Peter Chung

A Thesis

Presented to the Graduate and Research Committee

Of Lehigh University

In Candidacy for the Degree of

Master of Science

In

Department of Electrical & Computer Engineering

Lehigh University

May, 2006

This thesis is accepted and approved in partial fulfillment of the requirements for the Master of Science.

04/25/2006

Date

Thesis Advisor

Chair of Department

ACKNOWLEDGEMENTS

I would like to extend many thanks to Professor Liang Chang, who was gracious enough to extend a research opportunity to me for me to learn and further my studies and Qiang Wang who has provided many help and insight into my work and whom I have learned many things from outside the classroom.

Contents

LIST OF TABLES

ABSTRACT

Wireless sensors are becoming more common place in research and deployment in real world for myriad of purposes. As the sensors improve in functionality and the networks increase in size, networks which encompass vast number of sensors are being deployed and getting assimilated to different infrastructures. The networks are increasing in size and in different operating environments ranging from hostile battlefields to nature preserves, it is has becoming more clear that methods in administrative management of networks has to improve for swift upgrade of firmware and software for collection of pertinent information as well as delegation of tasks in real time. Ability to keep the network up to speed with knowing which parts of the network might be failing or experiencing difficulties is also crucial. Efforts to improve on the current implementations while keeping in mind the power consumption and hardware limitations would greatly benefit the deployment and management of sensor networks as well as adoption of WSN into scenarios that will pioneer the wide spread use of sensor networks.. Scoping of nodes within a WSN will be a means to increasing administrative powers over the network, as well as prolonging the life span of the individual nodes, facilitating the assimilation of WSN into deviant scenarios more viable.

Keywords

Dissemination Protocol, Wireless Sensor Networks (WSN), Network Reprogramming, Scope Selection

1 INTRODUCTION

Increasing research focus has been directed towards the sensor networks, their usability in different scenarios as well as possible trends in current technology. With wireless sensor network deployments increasing in size everyday under different environments, it has come to light the how difficult it has become to maintain the network and all the sensors. Too much resources and time would be wasted looking after the sensors the traditional way. This has led many researchers, including ourselves, to look for ways to manage the network, update/upgrade the firmware and the software of the sensors through renovate implementation of dissemination protocols[13], where the updates of binary code would be diffused throughout the network.

Implications of such improvement on the current sensor network would be far reaching. From deployments in battlefields where human intervention for maintenance would be slim to nature preserves covering vast grounds. The resources and time saved from being able to manage the network from one central access point would facilitate the adoption of the sensor network in other scenarios as well. Right now, human intervention is needed to correct flaws and update the binary images on sensors which make the current implementation of sensor networks not robust or scalable. With our goal in mind, we have to consider several aspects. First would be the dissemination protocol that would be used to distribute the code throughout the network. Second would be the actual algorithm used to update the firmware and software on the sensors. Keeping in mind the power consumption, bandwidth and hardware limitations that exist in current networks present a challenge. The storage hierarchy is as follows: communication packet (36 bytes) << RAM (4 K-Bytes) <<

program size (128 K-Bytes) < external flash (512 K-Bytes)[5][14][8]. Ram would act as the buffer, where program memory would store the "golden image" of the currently running program. External flash would hold two images of programs and the boot would tell when the sensor loads which image to boot from by storing the address of the first byte of the program.

Several ongoing researches in this area include XNP and Deluge for transmission protocol and Incremental Programming[6] for the update of the binary images on the sensors. XNP is the implementation that is available thru the TinyOS sensor network operating system. The limitation with this implementation is only for one hop making it not robust or scalable. More robust dissemination protocol would be Deluge that made diffusing of updates throughout the network possible. Yet another approach is Multi-hop Over-the-Air Programming (MOAP) but this approach is ad-hoc based and not applicable to the sensor networks.

2 RELATED WORK

Previous works mentioned include XNP that is currently available in TinyOS environment since TinyOS version 1.1.7. Improvement upon this work would include Deluge[1] and Multihop Over-the-Air Programming[3]. Each work has unique features that try to come up with an answer to the current research. Each has weakness and strength. The following lists brief descriptions of the current existing approaches

2.1 XNP

Acronym for Crossbow Network Programming, it is the current implementation of network programming, releases with TinyOS v1.1. This method is very crude and not scalable due to the fact that the primary node connected to the host program disseminated entire binary image, even with the most minor change, to nodes only single hop away. This is the biggest shortcoming with this implementation when taking into consideration power consumption and bandwidth utilization in large networks. However, this has been the base of how current implementation has started so this was a step in the right direction.

2.2 *Multi-hop Over-the-Air Programming (MOAP)*

The multi-hop code distribution method is considered as a special case of reliable data dissemination. MOAP[10] is intended for the mica2 platform and there already is the minor shortcoming in the implementation being not compatible. In consideration of saving power consumption and bandwidth, this implementation also took the approach of finding the difference between the current program image and new program image sending out instructions of copying, shifting, and patching memory address of the updates. However, this research was more geared towards the actual dissemination of the updates and different techniques used. MOAP tries to improve on past work of LOBcast where nodes keep track of available updates and requests the newest version, the request moving towards the sensor access point as referred in the LOBcast. Another approach is Bombilla, which is radio-stacked virtual machine sending out capsules filled with up to 24 instructions. Upon reception of the capsules will, sensor will check the version number and if newer than the current version, will update the program image.

MOAP was influenced by Directed Diffusion[7], where the bases would flood the network with interest packets and sensors would respond if in need of the data. Taking into consideration the fact that updated program image must reach all the sensors in the network, even nodes that were disconnected from the network for a time period. Also, all the optimization techniques used have to take into consideration the constraints of energy consumption and limited hardware resources, mainly dynamic memory so certain other requirements were sacrificed in order to minimize the impact of the two main constraints, which was latency.

In the dissemination of the data, the approach to limit the number of data packets flooding the network led them to consider suppression mechanisms and zoning concepts. Partitioning the network into neighborhoods (like zones) where few, preferably one, node is source, and that node is in charge of receiving and sending out updates, this would save the network from the implosion of data disseminated. Possible downfall would be if the source node(s) are down, then that neighborhood would not get the updates unless a mechanism is implemented for election of new node(s) once this fact is known to the nodes.

Reliability of data transmission is another important factor. In order to utilize the hardware resources efficiently while trying to keep the network from imploding with requests and acknowledgements, method called forward error-correction, a form of sliding window was used to hold certain amount of packets, where source node would send out K number of packets and wait for NACK (negative acknowledgements) of the packets that the sensor did not receive, making the receiver initiate the repairs. With this mechanism, segment management is an issue and in order to save power consumption, store a small bit map of the segments received to not accept duplicate packets or any random packets falling out side the window, wasting memory.

With the description of the design space and after running comparison, MOAP's implementation choices are ripple[3] dissemination, unicast retransmission policy and sliding window segment management. This implementation has not been tested out on actual sensor deployment of great size, only tested successful of 4 hops away from the base station, and the implementation choices made here with unicast retransmission and sliding window seems to create overhead.

2.3 Incremental Network Programming for Wireless Sensors

This approach moved away from the distribution of the changes to the binary image and focused primarily on the actual algorithm to generate the difference (delta B) between the previous versions of the code to the new updated version. Incremental programming approach breaks down the network programming process into three steps: encoding, dissemination and decoding.

Encoding entails the host PC to generate the difference between the previous versions of the code to the newer version using the Rsync algorithm. Rsync algorithm, originally used to update binaries between powerful PC over low bandwidth connectivity, creates a hash table for the fixed size blocks, calculates the checksum pair (checksum, hash) of each block for the updated version of the code and compares it to the current checksum pair for the block to see if there is a match. In the case that there is a match, host program sends out a copy message as oppose to sending out changes in data form all handled through script commands sent out in the messages over the network. It is assumed that the host program keeps track of the version of the current program image.

Dissemination is an area that was not heavily pursued in this research. Incremental network programming currently uses the XNP single hop network dissemination protocol to send out messages for updates to programs but claims that incorporating Deluge would be straight forward due to the fact that the implementation is very modular. This might be a possible area to study further, in how true their statement is. In case Deluge is not readily usable for Incremental network programming, try to come up with a possible solution.

Decoding is the most interesting part of this reading for this strictly deals with the sensors and demonstrates how approach alleviated the power consumption and bandwidth utilization constraints. Better the algorithm is to come up with the difference, resulting in less data being sent out in messages, network resources is minimized, but calls for many of external flash memory reads and writes to copy the blocks of code using consuming power. If the program is very different, then many messages will be sent, saturating the network but would not result in excessive I/O on the external memory which consumes considerable energy. Only few instances, of the final version of the code will be upgraded and written. Looking further into algorithm used to create the delta B might prove useful as well.

Once the program image has been updated, then the reboot command is issued along with the base address of the external flash memory where the program image should be loaded from, starting as the base address of the newer version of the program image. Then the network update is complete and their tables give numbers for different scenarios.

2.4 MNP: Multi-hop Network Reprogramming Service for Sensor Networks

Mnp[2] is a network reprogramming scheme that allows for one node to send out updated image of the code from one central node to the rest of the network. This protocol is an improvement on the Xnp protocol in that Mnp disseminates code in a hop-by-hop fashion. With the sender selection algorithm, to avoid the message collisions that derives from concurrent senders, the new program image is send from one node to multiple nodes one hope away, and all the nodes within hearing distance gets the exact image of the new program. Afterwards, new sender is selected based upon the criteria of the sender selection algorithm and the process is repeated until the entire network has been updated.

As with any network reprogramming, Mnp tries to satisfy several very important requirements which are: reliability, autonomy, low memory usage, energy efficiency, and speed. Most important is reliability for the correctness of the code dissemination to work. The protocol must guarantee 100% correct dissemination of the code or the protocol would not be viable. With that in mind, other requirements were listed and tradeoffs were made between the listed to work within the hardware limitations of the sensors and satisfy the working conditions of the protocol.

Mnp consists of four phases to disseminate the program and have the nodes boot up to the new image. These four phases are Advertisement/Request, Forward/Download, Query/Update, and Reboot. In Advertisement/Request phase, nodes announce the availability of the new version of the code. All nodes interested in the new image sends a request for the code. Nodes that advertised also keeps track of how many nodes made a request and makes few more advertisements with how many requests it has heard, letting other senders know how dense its listeners are. Node with the most number of requests would win and all other senders would be put to sleep while the one node is selected as the sender and start forwarding the code. Once the node decides to forward the code, it divides the program into segments and broadcasts it to its listeners. The requesting nodes store the incoming segments differentiated by sequence numbers, in the EEPROM in a linked list during the transmission. Once all the segments have been transmitted by the sender, it broadcasts a query message and the receivers check its linked list of segments to see which segment is missing and asks for updates from its parent (sending) node. Once the whole image has been received, the new program image is moved from the EEPROM where it is stored to program memory and the node boots into the new image during the reboot phase.

Mnp, a multihop network programming protocol that presented rate-based flow control, simple greedy sender selection scheme was a big improvement upon some of the protocols mentioned above such as MOAP which didn't consider sender selection scheme and Deluge, which put the current page(segment) in the RAM then writes the whole page to EEPROM has an edge considering the hardware limitation of limited memory even though, many writes do consume more power. These implementations based on their decision to put emphasis on certain requirements and make tradeoffs with others has many advantages then the previous implementations.

2.5 Deluge

This is the multi-hop network programming mechanism that performs two main functions, in transporting of block data and reprogramming boot loader. Deluge take into consideration the propagation of the code in a large sensor network and implements regulation of traffic control through optimization techniques such as adjustment of packet transmission rate and spatial multiplexing as well as suppression method in transmission to minimize the collision and congestion.

Deluge is a more versatile implementation of network reprogramming, getting accepted as the standard and deprecating XNP. The dissemination algorithm implemented in Deluge guarantees 100% update of new binary throughout the network, as well as facilitating the maximum utilization of bandwidth at near 90 Bytes/second. Due to the fact that Deluge puts heavy emphasis on the dissemination of the code and the way the algorithm was implemented, breaking up the "golden" binary image to pages and further breaking the individual pages to packets for transmission through the wireless network it makes is very scalable in and easy for integration to other algorithms that puts more importance on the actual update algorithm of the binary image.

However, this network programming implementation focuses primarily with the dissemination of the code rather good implementation of code update algorithm. Unlike

MOAP, this approach uses fixed size page as unit of buffer management and transmission. Also, it sends out the entire image rather than the sending out only needed parts of the updated code. In a scenario where the code was only modified slightly to change a detail of the code, sending out a new binary image to the wireless network would waste power which is a heavy overhead in wireless sensor network where power consumption must be limited to only essential activity in order to facilitate the longevity of the individual sensor nodes.

In the Incremental Network Programming approach, the researchers took this into consideration and came up with a differentiation algorithm to minimize the overhead in bandwidth and power utilization.

Comparison of the network programming protocols are drawn out into the table below, differentiated by "stages", based on methods of initiating updated, distribution range, different pars of the mote that gets updated and the rebooting of the mote to the updated image.

	Update	Dissemination	Binary Image	Encoding/Decoding
XNP	Push	Single-hop	Whole	Native code
MNP	Pull	Multi-hop	Whole	Native code
Deluge	Push	Multi-hop	Whole	Native code
MOAP	Push	Multi-hop	Whole	Native code
Incremental Network Programming	Push	Multi-hop	Difference	Native code

Table 1 : Comparison of the WSN reprogramming methods and the differences

After inspecting the specifications of the discussed protocols, I gauged the performance of protocols such as Mnp in respect to Xnp which it builds on top, to see if implementation of the protocol was justified with gains. Xnp performed slightly better in each of the 3 tests for disseminating code of different sizes to nodes a single hop away. The RAM and ROM memory usage by Xnp is 0.46KB and 15.32KB respectively. Mnp requires RAM and ROM memory of 0.65KB and 20.05KB for an additional overhead of 0.19KB and 4.73KB respectively. Mnp protocol used the TinyOS default packet size of 56 bytes for both, with 29 bytes for payload and the rest of the bytes used to hold fields for preamble, CRC, etc. as oppose to Xnp that send a capsule for individual lines of the code image. Both protocols is rated based to not incur overheads that come with sliding window and maintaining send packets information. Therefore, both use the similar query/update phase with updates being broadcast rather then unicast. Xnp's implementation is not as in depth, lacking sender suppression algorithm because it doesn't have any need for it and this contributed to Xnp protocol having better performance disseminating code a single hop away. This same cost will be incurred every time sender suppression algorithm is activated each hop throughout the rest of the network. However, this small cost is more then offset by the fact that Mnp allows for updating of program image through a WSN of any size, topology and density and take into consideration the broadcast storm and hidden terminal problems with sender suppression algorithm.

All the previous mentioned work only deals with dissemination of code and update algorithms to make more efficient use of the network and hardware resource. Each of the protocols use flooding, which is the standard method of communication in the current

WSN. In their implementation of the protocol, there are means of suppressing communication within certain nodes to handle Broadcast Storm[9] effect which leads to transmission collisions and unnecessary power consumption. However, none of the research has covered, scope selection where only certain nodes, group of nodes or nodes meeting certain conditions will be involved in communication that would lead to better autonomy and better scale to meet the diverse application needs of WSN.

One such paper that deals with this topic is "Scoping In Wireless Sensor Networks" [4]. This paper introduces what it might mean to implement scoping in WSN and the versatility that might be bestowed to the WSN. Scoping in this paper is introduced as "generic abstraction for the definition of group of nodes". Such abstraction would enable partitioning of WSN capability and usage. Careful consideration has to be given to tradeoffs between the two approaches of universal high level interface in declarative query processing such as TinyDB and application specific low level implementations that focus on updating individual nodes directly as described in the above mentioned works. Scoping would facilitate the optimization of routing and resource scheduling enabling diverse application requirements to coexist in a WSN deployment.

Current implementation of publish/subscribe system[12] does not exhibit control as one of its characteristics. Scopes in WSN would serve to provide a) design tool, to identify groups of components (b) lay down infrastructure to control the dissemination of data within the network. These abstractions would facilitate the means of both network-wide and local node-subsets regions. Possible selection rules of nodes in situations include but not limited to node features, geographical location, and network topology amongst others.

Nodes that differ in features, such as different sensors, sampling rate, processing capability and available software modules could make up a rule set of scoping. Geographical scope is defined by absolute position in a Cartesian coordinates or relative coordinates to a particular node, such as "nodes within 10m of the current node". Selection rule based on network topology also is based on node position but different in that this rule looks at node density within an area, or hop-count from another node.

In order to implement the rule-set mentioned above, and to leave room for these modules to evolve and mature, the implementation should be done in two steps. Scope deployment is the initial step of creating the scope for the first time on selected nodes. Scope maintenance updates the scope and adds/removes nodes based on the rules. These modules would fit in nicely with multipurpose sensor networks with nodes in different locations and functionalities are providing different levels of quality of service allowing more efficient use of resources and bandwidth.

3 Scope Selection

When looking at all these implementations of code dissemination within the Wireless Sensor Network (WSN). The code is disseminated from a central node to the rest of the network. The current implementations lack the flexibility of being able to update only certain nodes in the network. They do not take into consideration scope selection within the network. Either by base node being able to select nodes that meet certain criteria, possibly the node id or for base node to update parts of the network depending on nodes satisfying the role of being able to aggregate different information within the network such as temperature or light. Not only will this enable the administrator to have greater control over the operation of the network, it will uphold the main design goal of WSN in making the life of nodes within the network to be elongated by minimizing transmission and receiving data which occurs when entire network is updated, with packet transmission that reaches magnitudes higher due to implosion of packets where same message is received from different sources and overlap that occurs when node fires a packet, a neighboring node *b* receives packet and fires back to node *a* which is within hearing range, where its effect could continue indefinitely if a mechanism is not in place to stop this dilemma. In this paper, we try to look at the possibility of adding scope selection to the current implementations of data dissemination to add robustness and scalability which would lead to energy savings, better utilization of the limited bandwidth and promote autonomy of the network which would facilitate deployment of WSN in more diverse scenarios.

3.1 Deluge Protocol In Depth

Deluge is a binary image dissemination protocols that was designed very modularly, making it easy for the protocol to be adopted into different designs for its optimal and elegant design of dissemination of data within the network. It also has a framework where each transaction layer is very evident, and could be molded with little difficulty. It is a multi single-hop data dissemination approach where a node with a new image will advertise this fact to the rest of the network. Those nodes within hearing range will respond with a request message of the new image. Data of the new image is divided into smaller segments named pages and these pages are further broken down for transmission within a packet which could contain 27 bytes of information. This contributes to the versatility of the protocol for when there is loss of transmission due to interference or attenuation of signal strength in the carrier medium, the recovery could be carried out more manageably by only requiring the node to get updates of the incomplete page rather then the entire image. This fact directly contributes to less transmissions being made within the network, alleviating power consumption by unnecessary transmissions and receipt of those transmissions and as well as minimizing the processing overhead of tracking the progress of the image update. Record of all the different phases of the dissemination is stored in the meta-data within the individual nodes. Metadata information holds program image summary, which consists of image version, number of pages for each image, and the number of pages completed and are available to send. Node also has individual records of the number of images the node contains at the moment. When the image update is complete for the nodes a single hop away, the nodes with the new image or pages (parts of the image) will repeat this process until the entire network has been alerted and updated of the new image. With this design, when an image update takes place, the entire network is updated without discretion.

The design of scoping was layered on top of the existing phases of the transaction so as not to disturb the working implementation. Picking target(s) is always carried out at the base node, which is the access point for the administrator to the rest of the network. The target node information will be propagated within a broadcast message to the rest of the network. At this point, depending on the different approaches to the design, phases of the scoping interaction will vary. The implementations of the design described further tries to explore different options and see the advantages and also discover disadvantages of the designed proposed.

3.2 Ripple Dissemination

The first approach is named *Ripple* dissemination. The name comes from the fact the once a transmission is initiated, no matter where the node is situated within the network, the data will propagate outwards and the nodes that are within the hearing range, unhindered by distortion and interference, will sense the transmission in the transmission medium and will process the data accordingly.

3.2.1 Design

The design for this implementation include methods to propagate communication in only one direction, outwards from the transmitting node rather then random direction, thus eliminating the implosion effect, and data transmission from neighborhood to neighborhood, for a particular target node. Both ideas contributing to the mechanism called *Ripple*, for the ripple like propagation characteristic in a wireless medium this method of transmission portrays.

When there is a transmission ripple, only one ripple would be within the network at anytime. Depending on how close the target node is from the base node, it will set the path confirmation message timer, closer to the base, the longer the timer will be. When the path confirmation message is scheduled to fire after a time and hears a confirmation ripple, it will reset the timer to a value within a predefined range, and this process will repeat for other confirmation ripples until the particular target node is able to start a ripple of its own. This will ensure that node further away will establish routes and tier back towards the base, only one ripple is within the network which will help in alleviating packet storm and hidden terminal problem.

In order to establish routes, without knowledge of the network topology, transaction has to be carried out in a flooding fashion, where a node broadcasts and propagates the scope criteria to the rest of the network. When the target nodes acknowledge this broadcast after marking itself a target node, thus establishing route, this transaction then leads to the idea of *Ripple* where there will be only one transmitting node within a hearing range for a particular target node. However, this does not guarantee that there will only be one transmitting node within a network for multiple target nodes. Idea of multicast needs to be incorporated into this implementation in order to deal with this dilemma. Multicasting within WSN leads to mechanism to correct this deficiency where one node within a network will transmit within a neighborhood, making this protocol more robust and economical because it eliminate MAC contention and only one node as oppose to n number of nodes transmitting for n target nodes within a neighborhood. Also solves the hidden terminal problem for nodes $x+1$ hop away. A neighborhood consists of nodes within a transmission range.

Multicasting compared to flooding is a lot more efficient where communication is *one-to-many*. Since flooding is the main source of communication in WSN, the gains that could be achieved in this scenario has to be considered carefully while strictly adhering to the limitations of memory and hardware resources for a network node. Such implementation leads to huge power savings due to the fact that power consumption in sending packets is at least magnitude greater than any other operation a node performs in WSN. Only tradeoff to this design is, unlike the Internet where the network is stable, WSN is encumbered with wireless interference in the communication medium and power failure of nodes, where if one node within the communication route fails, the transaction would not be able to complete. If this is the case, the transaction would have to start from the beginning with a different route then the last. However, that tradeoff is only minor as more benefits could be achieved from such implementation being in place as will be shown in the evaluation (4) section.

3.2.2 Implementation

The basic means of communication between nodes within WSN is flooding. However, this technique is not tolerant to implosion where duplicated messages are retransmitted to the same node as well as overlap where same packet is received from multiple sources. In order to alleviate both problems, the transaction was made to propagate in only one direction, outwards and each node will participate in that scope transaction only once. This means, if the base node fires a scoping message, all the nodes will hear the broadcast and will respond to the message accordingly depending on the state that the node is in. However, if the node hears the same message again, from the same part of the scoping transaction, it will drop the packet and take no further action. This make the propagation of the message only in outward direction where destructive effects of overlap is directly taken care of and mitigates the adverse effects of implosion as well for the nodes that have already taken part in the transaction.

When the target node fires back path confirmation message to the sink node, the same idea applies where nodes will hear the message, node with its id in the route list will respond but only once. When the message is forwarded to the next node in the route list, and hears the same message again, it will take no further action thus making the transaction propagate only away from the originating source and will erase the overlap effect.

With only one node responding within a neighborhood, this directly attributes to the ripple mechanism providing a solution to the packet storm and MAC contention and coupled with the mechanism above, solves the hidden terminal problem as well. Another problem that had to be considered was the MAC contention. One of the properties of the ripple mechanism is the tradeoff between power consumption and latency. Rather then compete to claim the transmission medium, this mechanism favors to let the current transmission take place unhindered where transmission is selected on time criteria rather then availability of data. After certain time, where nodes involved in the transaction have finished sending and receiving data and extraneous packets have died down within the network as well, next transmission will begin thus taking a longer time to complete yet saves power by avoiding the two most power hungry operations in sending and receiving data unnecessarily where contention might void either or both operations. This requires strict timing of delay to achieve the optimal transaction completion time while maintaining the integrity of the protocol.

Due to the fact that time delay for transmissions will be calculated by the hop count from the base node, with bigger hop counts getting more aggressive time delays, the ripple mechanism will be resetting every target as it sweeps, and tier down toward the base node. As one ripple sweeps the network and resets the time delay accordingly, each ripple will start off closer to the base node until the transaction is completed. The time delay will give enough time to let the scope message die down after certain number of hops. After the initial transaction message dies down, and all the target nodes have responded back with route confirmation messages, base node will finally send out advertisement message of the new image that has to be disseminated. This method trades accuracy for speed where it is more conservative on the power consumption then the speed of the update.

This transaction takes place within the packet layer where processor is more involved and a little more power is consumed processing the packets but no mechanism has to be in incorporated to guarantee the arrival of the packet to the next hop. When this occurs within the network, the node will be stuck at that phase of the transaction and will not be able to progress. More mechanisms could have been put in place to fix this situation but more additions only attributed to complexity and bloated design.

3.3 Handshake Dissemination

This implementation was geared towards energy saving by removing a layer of the scoping protocol from the previous section and includes direct transmission to the node next hop away. Having established a goal, and carrying out the implementations, problems that were not present when transmission was taking place in the packet layer started surfacing which made the implementation details more complicated then anticipated.

3.3.1 Design

The message will originate from the base node and will contain information about the specific target nodes. Along with the scoping information will be a list where each node that the packet travels thru will enlist its node id as part of the route to the target node. This message will propagate throughout the network until all the targeted nodes mark itself and the packet will gradually die down within the network. Each target node will save two possible routes back to the base node. Target node will confirm the route with the shortest distance to the base. When the route has been chosen, the target node will filter the route list and send the packet to only the node one hop away. This is accomplished by addressing the packet with the MAC layer address (node id). This filtering process will continue from node to node until the route confirmation has reached the base node. Once this route confirmation reaches the base node, base node will mark off as having established a path with that target node. This process repeats for individual target nodes. When all targeted nodes have responded with the same message, base node will start the normal image update transaction but after the scoping transaction only targeted nodes and the forwarding nodes will partake in the rest of the communication.

Due to the fact that network problems like contention due to hidden terminal and broadcast storm still exist, the handshaking mechanism had to be implemented to ensure the transmission of the information packet.

3.3.2 Implementation

The first layer of transaction has to be broadcast. This is due to the fact that base node is dealing with an arbitrary network. Without knowledge of the network, the base node can not make any assumptions about the layout to jeopardize the transaction integrity. After the broadcast has settled, each target node will use one of the two aggregated route, picking the shorter of the two and start route confirmation, directing the transmission to the node one hop away. The nodes that are within hearing range but do not match the MAC address will just drop the message at the hardware layer. Because of this design, when a node is in the midst of listening to a packet that is not directed towards it and is immobilized for the 25ms where one transmission dominates the transmission medium, handshaking mechanism had to be put in place to ensure delivery of the message all the way to the base node for an established route. Base node only will go onto the next phase of the dissemination when all target nodes have responded. The simulations show that this handshaking mechanism created many collisions and retransmissions which led to high power consumptions for individual nodes involved in the dissemination. Immediate optimization to deal with this situation was to adjust the delay timer drastically but again, balance between speed accuracy had to be met.

Due to the design of *Handshake* dissemination, this scoping approach will work best for networks that are not dense due to the fact that there will be tremendous contention for nodes around the base node where all the information is being aggregated to from targeted nodes the same number of hops away. Scenario where one route could service multiple target nodes would be most ideal which again, brings the idea of adaptive multicasting[11] within the WSN where it would be a tremendous boost to the performance of not only scope selection within the WSN but any dissemination of information to the network or aggregation of data from the network, by reducing media contention, negating the retransmission costs once a collision does happen and less processing is needed by the node leading to big power savings.

4 EVALUATION

The simulations were carried out with a network semi-heavy node density. A node could transmit to eight nodes around it which was ample enough nodes to simulate hidden terminal and packet storm scenarios but not too dense to be bogged down by lost transmissions to slow down the testing. Simulations were focused on targeting specific nodes or sections of the network to draw comparison of effectiveness of scoping within the network.

4.1 Comparison

Msg_Type\ Target_Node(s)	3	15	5,10	5,10,15	3,12,15	10,11,14,15
SCOPE	54	51	53	53	60	68
SCOPE_RECV	146	126	122	125	140	157
PATH	18	18	34	62	50	77
PATH_RECV	63	63	100	164	170	277
ADV	172	187	107	189	331	308
REQ	43	46	23	46	122	119
DATA	3855	3913	2563	3913	6167	6638
DATA_RECV	3840	3840	2560	3840	8960	11520
Time to Completion (sec)	275	275	245	276	341	339
Route: Base->Target(s)	0->1->6->3	0->5->1 0->15	0->5 0->5-> 10	0->5 0->5->10 0->5->10 ->15	0->1->6 ->2->3 0->5->8 ->12 0->5->10 ->15	0->5->10 0->5->10 ->11 0->4->8 ->13->14 0->4->9 ->14->15

Table 2 : RIPPLE DISSEMINATION SIMULATION N RESULTS

Msg_Type\Target_Node(s)	3	15	5,10	5,10, 15	3,12,15	10,11,14,15
PATH	31		30		31	31
PATH_RECV	139		134		137	121
ROUTE	5		4		21	34
ROUTE_RECV	9		8		27	46
HANDSHAKE	4		4		12	19
ADV	312		277		614	334
REQ	60		40		213	72
DATA	3992		3340		9989	4577
DATA_RECV	5120		3840		11520	7680
Time to Completion (sec)	311		309		520	380
Route: Base->Target(s)	0->1->5->2->3		0->5		0->1->4 ->8->12	0->1->5->10
			0->1->5->10		0->1->5 ->2->3	0->1->5 ->10->11
					0->1->5 ->10->15	0->1->5 ->10->14
						0->1->5 ->10->15

Table 3 : HANDSHAKE DISSEMINATION SIMULATION RESULTS

Table 2 shows the simulation runs from the *Ripple* dissemination. One can notice the linear correlation between DATA packets sent and received. However, for *Ripple* dissemination, where one of the nodes served as forwarding node for two target nodes, it is very apparent to notice the discrepancy in the number of packets sent and received. This performance gain was achieved that came from the huge discrepancy resulted in much of power savings which is a very positive feedback. This implementation do not have mechanisms to control this feature but if multicasting could be achieved for scoping in wireless sensor network, benefits of power consumption, better utilization of bandwidth that directly contribute to less packet storm, collisions, and hidden terminal effect will be tremendous.

Looking at the output columns of target node 15 and target nodes 5, 10, & 15, the presented indicate only three network nodes were involved in the transmission. For just one target node, network nodes 5 and 10 were involved as the forwarding nodes and the target node itself was the termination point in this transmission. On the other hand, when three nodes were targeted, nodes 5, 10 & 15, the target nodes acted as the forwarding node as well so in this respect, this there was no extra image update packets generated and the only increase packet count between the two simulations came from the fact that target nodes had to confirm a route to the base node before the image update could take place.

The column containing output for target nodes 10, 11, 14 & 15, which were located the farthest away from the base node, created the most packets. Due to the fact that the network nodes were closely stationed, and also the transmission is almost being concentrated to a point, there were collisions which contributed to more network nodes being involved in the image update. With more nodes came increased packet exchange of information at every phase of the update and really contributed to more packet count when the actual image was being transmitted.

Table 3 is the compilation of results from the *Handshake* dissemination. It is interesting to notice the lack of results for target nodes 15 and 5, 10 & 15. Reason behind this lack of data will be discussed in more detail in the later section. Similar results were gathered when network nodes were targeted with this dissemination protocol. However, because of more contention in the media, unlike *Ripple*, where one transmission would propagate without much hindrance throughout the network, not the most efficient routes were chosen.

involving more nodes leading to more power consumption.

Running a parallel comparison between the two protocols, in most of the simulations, *Ripple* outperformed *Handshake* in most simulations with lesser packet count and energy consumption and faster update time. Interestingly, this was not the case in the simulation where several nodes in close proximity were scoped. *Handshake* still took longer to finish the image update. However, packet transmission was lesser by approximately 6000 thousand that lead to 33.8% power saving. This is an interesting anomaly where *Handshake* implementation shows promise scoping of nodes in a dense area such as this simulation tried to portray. If adaptive scoping is possible according to the network layout and available network information, such a design might prove to further boost the effectiveness of the implementation.

Below is the computation of how much power was consumed during the simulations. The operating power consumed with is the Mica2 motes. There are motes that have power saving hardware built into the motes to only consume power while transmitting and be in sleep mode to conserve power. However, Mica2 motes are just sitting in an idle state while not taking part in a communication or processing information.

Operations Power consumption	(nAh)
Read a data block from EEPROM	1.261
Write a data block to EEPROM	85.449
Send one packet (27 bytes)	20
Receive one packet (27 bytes)	8
Idle listen for 1 msec	1.25

Table 4 : Power Consumption Char [8]

Eq.1
Total SEND = (SCOPE+PATH+DATA)(READ)+(SCOPE+PATH+DATA)(SEND)+(ADV+REQ)(SEND)

Eq.2
Total RECV = (SCOPE+PATH+DATA)(WRITE)+(SCOPE+PATH+DATA)(RECV)

Eq.3
Total IDLE = (TIME*IDLE)

Equations above will sum up the total the power consumption of the activities that nodes within the network partake in. The total power consumption is calculated by adding the total of the individual activities. These equations only pertain to *Ripple* dissemination.

Node	Send		Receive	Power Consumption
	S+P+D	A+R	S+P+D	
3	3927	215	4049	466511.28
15	3982	233	4031	466171.56
5, 10	2650	130	2782	319223.8
5, 10,1 5	4028	235	4129	472153.32
3, 12, 15	6277	453	9270	1009214.9
10, 11, 14, 15	6312	427	11954	1270267.0

Table 5 : Power Consumption measurement for Ripple dissemination

Eq.4
Total SEND :
(PATH+ROUTE+DATA+HANDSHAKE)(READ)+(PATH+ROUTE+DATA)(SEND)+(ADV+REQ)(SE
ND)

Eq.5
Total RECV : (PATH+ROUTE+DATA)(WRITE)+(PATH+ROUTE+DATA)(RECV)

Eq.6
Total IDLE : (TIME*IDLE)

Equations 4 – 6 are to calculate the sums of energy consumption for the activities the nodes partake in for the *Handshake* dissemination.

Node	Send P+R+D+H	A+R	Receive P+R+D	Power Consumption
3	4032	372	5268	585763.7
15				
5, 10	3378	317	3982	450580.78
5, 10,1 5				
3, 12, 15	10053	827	11684	1322545.4
10, 11, 14, 15	4661	406	7827	840606.9

Table 6 : Power Consumption measurement for Handshake dissemination

Normal transactions of Deluge, which include advertisement of new image to the neighboring nodes, request of the available data to the advertising node, and sending and receiving of the data between the advertiser and the listeners. Therefore, the number of packets generated by each additional node within the network that participate in a transaction will produce linear growth of packet exchange. This phenomenon is notice by observing the column outputs of target node(s) 15, 5&10 and 3,12&15 of *Ripple* dissemination. Due to the orientation of the nodes in the simulations as portrayed by Picture 1, targeting nodes 5&10 involves the fewest nodes in the update transaction. Only three nodes are involved, and even though other update scenarios target only one node, the forwarding nodes also create the traffic of directing the flow of the update towards the target node. It is possible to conclude that the number of actual image update packets generated is the summation of the hop counts of each target node multiplied by the number of data exchange. This leads to the conclusion that if one node is able to service more then one listener, the packet transmission will drastically, noticeable by couple of thousand packets in the simulations.

4.2 *Observations*

In both designs, the timing was the key issue. Mechanism to calculate the most opportune time to transmit information to not cause collision. In this regard, *Ripple* dissemination was the better of the two designs.

There is no simulation result for *Handshake* dissemination, targeting nodes 5, 10 and 15. After reviewing the trace output file, an interesting phenomenon was observed which the protocol was not designed to handle. As with the bottle neck that exists near the base node, this effect is mirrored in the opposite end of the network if the network is in a grid. A lot of contention exists and for the *Handshake* implementation, this creates a rather interesting problem not found in the *Ripple* dissemination. *Handshake* implementation calls for the target nodes to store two possible routes, calculate which route has the shorter hop, and confirm the routes by only sending route confirmation message to the node one hop away, labeled by the MAC address (node id). However, because of the contention, node 15, located the furthest away from the base node, experiences a lot of the contention as the broadcast transmission from the base node reaches the section of the network. This is due to the fact that once the broadcast started from the base node, the transmission will spread out to the rest of the network and then again start concentrating around the corners of the network due to the grid layout. When this initial wave of transmissions and the collisions that ensue die down and the packets that has been bouncing around the network finally reach the this particular node, mechanism to kill off packets kicks in and the packet is no longer a viable packet. Immediate solution to this dilemma would have been to drastically change the random delay values of timer to forward the message, alleviating the contention but that approach was not a good solution to the problem. Another solution would have been to fire the one path it received after a certain time, or even start a special packet back towards the base node and the first possible aggregated route that reaches the base node will be the confirmed as the route by the base. Any other algorithm with the extra layer of transmission and its complexity only lead to bloated images and outweigh the gains. Temporary solutions to the problem will not fix the nature of the problem as the design will not be robust enough to adjust to different deployment scenarios. This observation revealed a rather interesting problem that needs to be addressed. There exist locations in the network where communication will be concentrated leading to many collisions and contention for the media. This leads to the conclusion that unicast and broadcast for scoping is not an ideal solution which leaves either multicast transmission or fundamental changes to the structure of the network layout and the way selection of target nodes are handled.

4.3 Further improvements

In the *Handshake* implementation, due to the nature of the transmission, where addressing is done at the MAC layer, to help reduce waste of power consumption of processing packets that is not destined for that particular node, it creates the need for message confirmation mechanism, where many collisions are caused, especially around the transmission concentration points like the corners of the network in a grid layout. Better design to ensure the delivery of the packet would make the *Handshake* dissemination more robust. Media contention is exacerbated if the network is densely populated with nodes within close range. Current design uses random delay value to deal with contention, which produces mediocre results. Mechanism to better calculate the delay values would strength this approach. The design should not involve heavy computation or result in heavy accounting of transmissions that the node is involved in, taxing the node resources, and but come up with optimal timing between each transmission.

Both implementations do not have a user interface. Each update is carried out by changing the target node parameter and recompiling the code before the dissemination takes place, which requires changing the source code by hand. A user interface with inputs for program name, version, which nodes to target, possible parameters that nodes within the network should satisfy before being targeted would make the administration easier.

5 FUTURE WORK

To mature the scoping in WSN for widespread deployment, a new infrastructure of network orientation might need to be explored. Clever designs and more accurate transmission timing might make scoping more viable but at this point, it presents too many obstacles and overhead to be used in common deployment. Especially big problem presented in the scenarios for transmission between the base node and the target node(s) is the bottleneck created by the nodes one hop away. The transmission concentration created as the information is exchanged between the base node and the rest of the network and the number of packets that has to be processed by these nodes grows tremendously as the network grows and more nodes are targeted for transmission.

One recommendation that could be made is to incorporate distributed management system. One possible path to explore is to have special nodes spread within the network at maximum transmission distance, even the use to signal amplifiers to let the transmitters have further reach, and carry out scoping in this manner. The further the specially designated nodes could transmit and cover more area of the network; fewer transmissions have to take place, which directly mitigates the hidden terminal and packet storm problem. Also, because the designated transmitters would be at optimal distances apart, there would be little or no media contention

Network infrastructure change is another path to study. Rather then randomly scattering the nodes, prearranged layout with physically layering the transmission between specially designated nodes. This layout would tier the transmission, and let the scoping and image updates take place at different layers of the hierarchy. This way, scoping could take place with less media contention, for each node would have different role in the transaction, and aggregating information from the network to the base node will take place without any bottleneck as is the case in the current designs.

Another approach might be to designate the entire network to be zoned. Nodes belonging to a particular zone would transmit but only nodes of that zone will be able to process the data. Again, a special node would have to be present where its duties include gathering relevant information from within its zone only and aggregate it back to the base node for processing.

The idea of special nodes within the network other then base node keep surfacing in ideas presented. As shown in the designs mentioned above, there is so many tradeoffs and overhead created with each gain that it is a very delicate to justify between gaining flexibility and ease of management with sacrificing power consumption, real time processing in case of profound implementations with heavy computations, and resource strains. Rather then the software approach, hardware solutions, with network layout schemes and designated nodes for assuagement specific tasks and communication schemes like multicasting might prove to be more beneficial.

6 CONCLUSION

The benefits of scoping in wireless sensor networks are apparent. There is the control and the flexibility that facilitates the administration of tasks for the administrator as well as having a more diverse and versatile network. Depending on how the implementation is done, tradeoffs exist: complexity in design vs. more versatile and scalable algorithms, strain on the limited resources of the node vs. efficient network management to mention a few. Depending on the size and purpose of the network, certain exchanges could be made to achieve this objective.

Scoping protocols for targeting specific nodes within the network presented in this paper, *Ripple* dissemination proved to be more efficient then the *Handshake* implementation. The *Handshake* dissemination could have been optimized to decrease the time slightly but the fact remains, the packet count got noticeably higher as more nodes were targeted. Keeping in mind that nodes will be managed by the base node, which means that the surrounding nodes one hop away will inevitably face high volumes of traffic designated to itself and the rest of the network, these nodes could be the bottle neck which can result in the rest of the network dealing with more traffic all the way back to the target nodes. This will be case when information packets are aggregating towards the base node and these nodes have to service a specific target node, then the next target node if multiple target nodes are selected, operating in first come, first serve (FIFO) queue style. Due to the wait created by the bottleneck, transmission will be delayed possibly even to the originating nodes of the transmission.

For this research, the main goal was to explore ideas of scoping and see which ideas are viable and could be studied further as well as figure out and explain the problems that exist. With these implementations of scoping in WSN, the paper presents that it is possible to achieve scoping within wireless sensor networks, with overhead involved. Section 3 of the paper detailed two approaches to the issue of scoping which still remains a very young and unexplored field. Of the two approaches, *Ripple* design proved to be more robust of the two implementations. In design, the presence of the extra layer of communication transaction between nodes is an obvious disadvantage. However, the nature of the design addresses many of the problems that persist within the WSN such as broadcast storm and the hidden terminal. These problems had to be explicitly addressed in the *Handshake* dissemination.

Achieving the objective led the study to discover what kinds of problems exist on top of normal limitations that apply to WSN. These problems are listed in the observations section as well as the source of the problems and possible solutions as listed in the section that follows. Possible suggestions included different approaches to the network layout and roles of the nodes within the network, to address the issue of scoping to extending the manageability of larger networks and ease the administration costs which will allow the networks to be more autonomous. The ability to target particular nodes, sections of nodes, or even only nodes that meet certain parameters to carry out hardware-specific, application-specific or event-specific tasks with minimal overhead will allow WSN to be applicable in many more deployment scenarios. The hardware involved in the networks are improving at a fast pace as well which might make some of the limitations mentioned here obsolete in the future. Scoping in wireless sensor networks is still a very new field of research and needs to be explored further for the advantages of optimized scope selection within a network with minimal overhead would make adoption of sensor networks more accessible to myriad other scenarios then the current deployment cases.

REFERENCES

[1] J. W. Hui, and D. Culler. "The Dynamic Behavior of a Data Dissemination Protocol for Network Programming at Scale", Proc. ACM SenSys 2004, pp. 81-94

[2] Sandeep S. Kulkarni, and Limin Wang. "MNP: Multihop Network Reprogramming Service for Sensor Networks," *icdcs*, 25th IEEE International Conference on Distributed Computing Systems (ICDCS'05), 2005, pp. 7-16.

[3] T. Stathopoulos, J. Heidemann, and D. Estrin. "A remote code update mechanism for wireless sensor networks". Technical Report CENS Technical Report 30, 2003.

[4] J. Steffan, L. Fiege, M. Cilia, and A. Buchmann. "Scoping in Wireless Sensor Networks: a Position Paper", Middleware for pervasive and ad-hoc computing, 2004.

[5] Qiang Wang, Yaoyao Zhu, and Liang Cheng, Reprogramming Wireless Sensor Networks: Challenges and Approaches, To appear in IEEE Network, May 2006

[6] J. Jeong, and D. Culler. "Incremental network programming for wireless sensors", Proceedings of the The First IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, 2004, pp. 25-33.

[7] C. Intanagonwiwat, R. Govindan, and D. Estrin. "Directed diffusion: a scalable and robust communication paradigm for sensor networks", In Mobile Computing and Networking, pages 56-67, 2000.

[8] Crossbow Technology, Inc documents. "Mote In-Network Programming User Reference" and "Mica2 Wireless Measurement System Datasheet", 2003.

[9] S. Y. Ni, Y. C. Tseng, Y. S. Chen, and J. P. Sheu, "The Broadcast Storm Problem in a Mobile Ad Hoc Network," Proc. IEEE/ACM MOBICOM'99, pp. 151-162.

[10] Niels Reijers and Koen Langendoen. "Efficient code distribution in wireless sensor Networks", In WSNA '03: Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications, pages 60-67, New York, NY, USA, 2003. ACM Press.

[11] J. Jetcheva, D. Johnson. "Adaptive Demand-Driven Multicast Routing in Multi-Hop Wireless Ad-Hoc Networks", Proceedings of the Second Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), 2001.

[12] J. Kulik, W. R. Heinzelman, and H. Balakrishnan, "Negotiation-based Protocols for Disseminating Information in Wireless Sensor Networks," Wireless Networks, v.8 n.2/3, Mar-May 2002, pp. 169-185.

[13] T. Stathopoulos, J. Heidemann, and D. Estrin, "A Remote Code Update Mechanism for Wireless Sensor Networks," Technical Report CENS-TR-30, UCLA, Center for Embedded Networked Computing, Nov. 2003.

VITA

I was born in Seoul, capital city of South Korea. My father's name is Byong Chul Chung, later baptized and now known as Joseph Chung. My mother's name is Yun Cha Chung and also baptized and known as Mary Chung. I have a little sibling named Thomas Chung who is six years younger then my age. I arrived in New York city, United States at the age of nine and attended Public School 150 in Woodside, New York for elementary school, Intermediate School 125 in Woodside, New York for junior high school, Brooklyn Technical High School in Brooklyn, New York for high school. I attended Lehigh University from Fall semester of year 2000 to Spring semester of year 2004 and received Bachelor's of Science in Computer Engineering. I started my masters program at Lehigh University in the Fall semester of 2004 in the department of Electrical and Computer Engineering and will be graduating in May of 2006 with Masters of Science in Computer Engineering. With interest in wired and wireless internetworking, I am currently pursuing the CCNA certification path and plan to take the following exams with further study.

END OF
TITLE