Single versus Multi-hop Wireless Reprogramming in Sensor Networks

Rajesh Panta, Saurabh Bagchi
Dependable Systems Computing Lab
Purdue University
465 Northwestern Avenue, West Lafayette, IN 47907.
{rpanta,sbagchi}@purdue.edu

Issa Khalil
College of Information Technology
United Arab Emirates University
United Arab Emirates
ikhalil@uaeu.ac.ae

Luis Montestruque
Emnet LLC
12441 Beckley Street, Granger, IN 46530
lmontest@heliosware.com

ABSTRACT
Wireless reprogramming of the sensor network is useful for uploading new code or for changing the functionality of the existing code. In recent years, the research focus has shifted from single hop reprogramming to multi-hop reprogramming primarily because of its ease of use. Practical experience from a multi-hop sensor network for monitoring water pollution, called CSOnet, deployed in South Bend, IN, indicates that single-hop reprogramming may be preferable under certain conditions to minimize reprogramming time and energy. In this, the user gets close to a node to be reprogrammed and wirelessly reprograms a single node at a time. The choice between single hop and multi-hop reprogramming depends on factors like network size, node density and most importantly, link reliabilities. We present a protocol called *DStream* having both single and multi-hop reprogramming capabilities. We provide mathematical analysis and results from testbed experiments and simulations to give insights into the choice of the two reprogramming methods.

Categories and Subject Descriptors
C.2.2 [**Network Protocols**]: Applications

General Terms
Algorithms, Performance, Design, Experimentation,

Keywords
Sensor networks; single hop reprogramming; multi-hop reprogramming; link reliability.

1. INTRODUCTION
Large scale sensor networks may be deployed for long periods of time during which the requirements from the network or the environment where the nodes are deployed may change. The change may necessitate uploading a new code or re-tasking the existing code with different sets of parameters. Wirelessly reprogramming the nodes is particularly useful because the network may be deployed over a wide geographical region and some nodes may be in difficult to reach places. A sensor node has limited power supply and memory. So, it is important to minimize the energy and memory consumption for network reprogramming. In recent years, the focus of the sensor network reprogramming has shifted from single hop reprogramming (only nodes within the transmission range of the base node (BN) are reprogrammed) to multi-hop reprogramming (all nodes in the multi-hop network are reprogrammed) because of various reasons. First and perhaps the biggest advantage is that from a user's point of view, it is tedious to perform many rounds of single hop reprogramming to completely reprogram the multi-hop network. Second, multi-hop reprogramming protocols like Deluge [4], Freshet [5] and Stream [9] spatially pipeline the code transfer (also called *spatial multiplexing*) and thus reduce the time to reprogram the network. That is, a node does not need to completely download the code image before starting to send the code to its neighbors.

But in some deployment conditions, like in combined Sewage Overflow (CSO) project implemented in South Bend, Indiana, multi-hop reprogramming can be costly in terms of reprogramming time and energy. In CSO, a multi-hop sensor network, called CSOnet, with nodes mounted on traffic lights and lamp-posts, is used to collect alerts from monitoring sensors planted in the manholes of the municipal sewage system. The network then forwards these alerts to gateways at major traffic intersections which make distributed control decisions to channel the flow to temporary reservoirs so that dumping the waste water into rivers or lakes can be avoided.

At first glance, it may appear pointless to sacrifice the relative ease of the multi-hop reprogramming in favor of node by node reprogramming. The conditions in which a sensor network is deployed may change over time. For example, the link reliabilities between the nodes in the network may change because of varying environmental factors. When link reliabilities are low, sending entire application image over multiple links imposes a heavy burden in terms of retransmissions increasing both reprogramming energy and time. In fact, for all current reprogramming protocols, except Stream, what needs to be transferred over the network is the entire application image plus the reprogramming protocol image. This exacerbates the problem by increasing the number of packets that needs to be transmitted reliably through the network. The increase is sometime by a factor of 20 [9].

This specific problem reared its head in the CSOnet deployment where it was observed that the batteries were being drained much faster than the theoretical calculations had predicted. Our

Tridentcom 2008, March18–20, 2008,Innsbruck, Austria. ISBN: 978-1-60558-009-8

investigation revealed that regular code updates being sent using the multi-hop method were the culprit for parts of the network, particularly the parts having linear topology and unreliable links. We decided to explore the possibility of judiciously using single hop reprogramming. In the single-hop method, the user visits each node and remotely reprograms it being physically as close as possible to the node. Technically this is single *node* reprogramming. However, the term single hop reprogramming follows the standard usage in the literature.

In this paper, we present a protocol called *DStream* having both single and multi-hop reprogramming capabilities. DStream is built on top of Stream [9]. It does not sacrifice the advantages of Stream with respect to code size and memory footprint. We use the terms DStream-SHM and DStream-MHM to represent the single and multi-hop reprogramming modes of Stream. Using mathematical analysis, testbed experiments and simulations, we draw valuable inferences about the two reprogramming approaches. The common insight that all three give us is that single hop may be more energy efficient and faster than multi-hop in some scenarios. For a given topology, the cutoff depends on the link reliability of the links in the network. High link reliability favors multi-hop reprogramming. Second, for networks that are linear (or close to linear), single hop reprogramming tends to be favored. The rest of the paper is organized as follows. Section 2 surveys related work. Section 3 provides the detailed DStream design. Section 4 presents the mathematical analysis. Section 5 explains the testbed and the simulation results. Section 6 concludes the paper.

2. RELATED WORK

In recent years, there has been significant research work aimed at developing protocols for reprogramming sensor networks. To the best of our knowledge, all of the existing reprogramming protocols provide either single or multi-hop reprogramming features, but not both. Importantly existing work is silent on the choice between the two approaches for different deployment conditions. The earliest network reprogramming protocol XNP [1] operated over a single hop. The Multi-hop Over the Air Programming (MOAP) protocol extended this to multiple hops [12]. The three protocols that define the state-of-the-art today are Deluge, MNP, and Freshet. They are all based on the idea of epidemic based reliable multicast whereby code images are flooded through the network in a controlled manner guaranteeing reliability through the use of epidemic multicast. Deluge [4] was the earliest and laid down some design principles used by the other two. It uses a monotonically increasing version number, segments the binary code image into pages, and pipelines the different pages across the network. It builds on top of Trickle [8], a protocol for a node to determine when to propagate code over a single hop. The design goal of MNP [6] is to choose a local source of the code which can satisfy the maximum number of nodes. Freshet [5] aggressively optimizes the energy consumption for reprogramming by allowing a node to sleep till the code reaches its neighborhood. It also reduces the energy consumption by exponentially reducing the meta-data rate during conditions of stability in the network when no new code is being introduced. Stream [9] uses the principles of Deluge for code propagation but greatly reduces the reprogramming time and energy compared to Deluge. Section III presents a brief description of Stream.

There have been some studies which show how low link reliabilities cause problems in multi-hop networks. [2] showed that shortest path algorithm in a network with lossy links selects a path with poor reliability. In [13], the authors evaluate Deluge and MNP for different densities and packet organizations. But as far as we know, there has been no prior work to study the effect of parameters like link reliabilities on the performance of multi-hop reprogramming.

3. PROTOCOL DESIGN

3.1 Background and Rationale

It is desirable to have the sensor nodes equipped with the facility of both single and multi-hop reprogramming so that a choice can be made at runtime based on the current network conditions (topology, link reliabilities etc). The obvious approach is to have two separate reprogramming protocols (a single hop protocol like XNP and a multi-hop protocol like Stream) stored in each node's permanent storage (external flash) so that it can run the appropriate protocol when required by loading that protocol from external flash to the program memory. This is not an attractive solution because requiring a node to store two reprogramming protocols decreases the storage (e.g. external flash for Mica2 is 512KB) for the application running on the nodes. Our proposed approach is to have a single protocol with both single and multi-hop reprogramming capabilities. Existing single-hop reprogramming protocols, such as XNP, were not designed with the ability of propagating the code updates through the network in a multi-hop manner. Therefore they cannot serve as a starting point for our protocol. Multi-hop reprogramming protocols like Deluge, Stream and Freshet are more suited for this purpose. Since Stream is the most energy efficient and fastest among these protocols, we chose Stream to build on to create DStream.

The main disadvantage of multi-hop reprogramming protocols like Deluge, MNP and Freshet is the overhead involved in reprogramming. Each protocol transfers the entire reprogramming protocol image together with the new user application image. Since the reprogramming protocols are of considerable complexity, the inflation in the program image size that gets transferred over the wireless medium increases greatly. The idea in Stream is to have all nodes in the network be pre-installed with the Stream-ReprogrammingSupport (Stream-RS) component that includes the complete functionality for network reprogramming. Stream-RS is installed as image 0. The application image augmented with the Stream-ApplicationSupport (Stream-AS) component that provides minimal support for network reprogramming is installed as image 1. The addition to the size of the program image over the application image size with Stream is significantly less than for previous protocols. When a new program image is to be injected into the network, all the nodes in the network running image 1 reboot from image 0 and the new image is injected into the network using Stream-RS. The new image again includes Stream-AS and the protocol avoids the entire reprogramming component from being transferred to all the nodes each time the network needs to be reprogrammed. The exact saving in terms of the number of pages transferred depends on the application. Any application that uses radio communication will need to add about 11 more pages if Deluge is used while while Stream-AS adds only one more page [9].

3.2 Design Approach of DStream

Let all nodes initially have Stream-RS as image 0 and the application with Stream-AS as image 1. Each node is executing the image 1 code. Consider that a new user application has to be injected into the network.

1. If multi-hop reprogramming is to be used, in response to the reboot command from the user, all nodes in the network reboot from image 0. This is accomplished as follows:
   a. From the computer, the user sends the command to reboot from image 0 to the BN.
   b. The BN executing image 1 broadcasts the reboot command to its one hop neighbors and itself reboots from image 0.
   c. When a node running the user application receives the reboot command, it rebroadcasts the reboot command and reboots from image 0.
2. If single hop reprogramming is to be used, in response to the reboot command from the user, a single node specified by the user reboots from image 0. This is accomplished as follows:
   a. From the host computer, the user sends the command to reboot a single node, say node α, from image 0 to the BN.
   b. The BN running image 1 broadcasts the reboot command along with the user specified node id α to its one hop neighbors. The BN then reboots from image 0.
   c. Each node that receives the reboot command, determines if the reboot command is targeted to it. If yes, it reboots from image 0. Otherwise, it ignores the reboot command.
3. Stream-RS starts to reprogram the node(s) that has rebooted from image 0. Thus, Stream-RS which forms the bulk of the reprogramming protocol does not need any modification to support the single-hop mode of operation.
4. Stream-RS uses the three way handshake method for reprogramming [9] where each node broadcasts the advertisement about the code pages that it has. When a node hears the advertisement of newer data than it currently has, it sends a request to the node advertising newer data. Then the advertising node broadcasts the requested data. Each node maintains a set S containing the ids of the nodes from which it has received requests.
5. Once the node downloads the new user application completely, it performs a single-hop broadcast of an ACK indicating it has completed downloading. In single-hop reprogramming, only one node sends the ACK while in multi-hop all nodes in the network are ultimately reprogrammed and send the ACK message. When a node n_1 receives the ACK from node n_2, n_1 removes the id of n_2 from the set S. When the set S is empty and all the images are complete, the node reboots from image 1 (user application).

From the above discussion, it is clear that DStream can provide both multi-hop and single hop reprogramming features. If the user specifies the id of the node to be reprogrammed in the reboot command, DStream reprograms only the specified node (single hop reprogramming). Besides this, the user can also specify an option for automatic switching between single and multi-hop approaches. When this option is specified, DStream starts with multi-hop reprogramming. When a node n_1 receives a request from a node n_2 for a page of the new image, n_1 keeps track of how many packets are requested for the same page in the next request by n_2. This gives n_1 the estimate of the link reliability between n_1 and n_2. If the estimated link reliability is less than some threshold (user specified), a message is sent back to the BN informing it about the current link reliability between n_1 and n_2. The BN then forwards that message to the computer. This suggests the user to switch to single hop reprogramming for n_2. In this way, nodes with low link qualities are reprogrammed using single hop method and other nodes are reprogrammed using multi-hop method.

4. MATHEMATICAL ANALYSIS

Here we present the final results of the approximate analysis of the reprogramming time and energy for DStream-SHM and DStream-MHM for linear and grid networks. Since the detailed mathematical analysis is fairly complicated and this paper focuses mainly on the practical aspects, we present only the final results and omit the derivations. The complete derivation is available at [10]. For linear networks, we assume that the spacing between consecutive nodes is equal to the transmission range and for grid networks, it is $\sqrt{2}$ times the grid spacing. Let the network have N nodes, application consist of N_p pages with A_{pkt} packets per page, L_{RS} and L_{RM} be the link reliability of single hop reprogramming (for the link between the BN and the single node being reprogrammed) and multi-hop reprogramming (we assume identical link reliability for all links) respectively, and P_s be the probability of successful transmission of a packet over a single link, which is equal to L_{RS} in single hop mode and L_{RM} in multi-hop mode.

4.1 Reprogramming Time

The relative reprogramming time of single-hop to that of multi-hop is given by

$$T_{conv(S/M)} = \frac{N \cdot N_P \cdot \sum_{i=1}^{\infty}\left[1-\left[\sum_{j=1}^{i-1} L_{RS}\left(1-L_{RS}\right)^{j-1}\right]^{A_{pkt}}\right]}{(3\cdot(N_P-1)+h_{\max})\sum_{i=1}^{\infty}\left[1-\left[\sum_{j=1}^{i-1} L_{RM}\left(1-L_{RM}\right)^{j-1}\right]^{A_{pkt}}\right]} \quad (1)$$

where h_{max} is the maximum number of hops in the network from the base node. Using Equation (1), Figure 1-a and Figure 1-b show the relative reprogramming time (single hop/ multi-hop) respectively for linear and grid topologies as a function L_{RM} for different network sizes with L_{RS}=0.95, Np=12 pages, A_{pkt}=48 packets, h_{max}=N-1, for the line topology, and $h_{\max}$ = m-1 for the $n\times m$ grid (ignoring the edge effects). For the linear topology, as the network size increases the multi-hop mode reprogramming is faster due to the pipelining effect of multiple pages. However for the 5 node network, when the multi-hop link reliability is less than 0.8, single hop reprogramming is preferred from the delay point of view. For the grid topology, the reprogramming time of the multi- hop mode is always better than that of the single hop mode due to two factors—the spatial multiplexing and multiple nodes receiving the same single broadcast of the code packet. The spatial multiplexing becomes more efficient with increasing network size, which explains the advantage of multi-hop reprogramming as

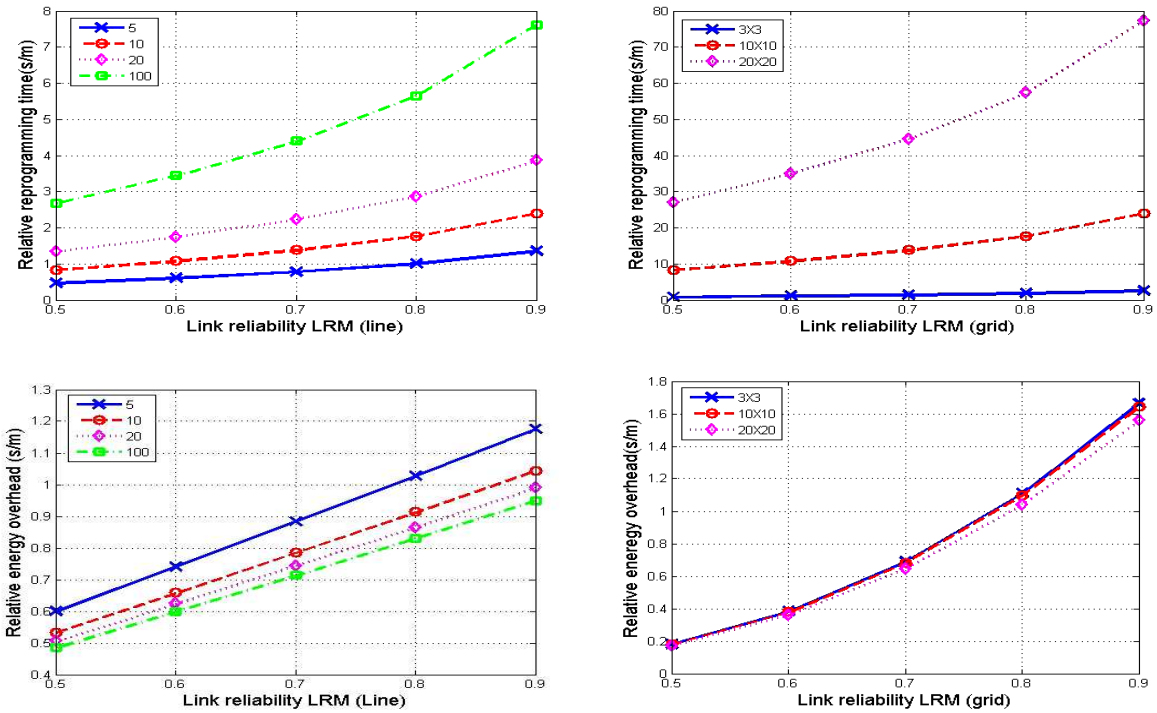

Figure 1: Relative reprogramming time (single hop : multi-hop) as a function of link reliability for (a) linear and (b) grid topologies. Relative energy overhead as a function of link reliability for (c) linear and (d) grid networks.

network size increases.

4.2 Energy Cost

Let S_h is the set of nodes at hop h that can be reprogrammed by one node at hop h-1 and $|S_h|$ be the average size of the set. The relative energy consumption of single hop to multi-hop is

$$E_{S/M} = \frac{N \cdot L_{RM}^{|S_h|}\left(A_{pkt}+1+\sum_{i=1}^{\infty}\left[1-\left[\sum_{j=1}^{i-1}L_{RS}\left(1-L_{RS}\right)^{j-1}\right]^{A_{pkt}}\right]\right)}{L_{RS}\sum_{h=1}^{h=h_{max}}(\alpha_h)\left(A_{pkt}+1+\sum_{i=1}^{\infty}\left[1-\left[\sum_{j=1}^{i-1}L_{RM}^{|S_h|}\left(1-L_{RM}^{|S_h|}\right)^{j-1}\right]^{A_{pkt}}\right]\right)} \quad (2)$$

By plotting Equation (2), Figure 1-c shows that the single hop mode is more efficient than the multi-hop mode for the linear topology with link reliability less than 0.8. Moreover, the difference increases, in favor of the single hop mode, as the network size increases. In linear topologies, only one node can be satisfied by the transmission by a node and this negatively impacts the energy consumption of the multi-hop mode. Figure 1-d shows that for a grid topology, almost irrespective of its size, the single hop mode is better when the link reliability is less than or equal to 0.8 and the multi-hop mode is better otherwise. For a deployment with higher transmission ranges and hence higher values of $|S_h|$, the balance will shift in favor of multi-hop reprogramming.

5. EXPERIMENTS AND RESULTS

We implement DStream having both multi-hop and single hop features using the nesC programming language in TinyOS. In this section, we compare the performances of DStream-SHM and DStream-MHM using both testbed experiments and simulations. The metrics that we use to compare single and multi-hop reprogramming approaches are reprogramming time and energy.

5.1 Reprogramming Time and Energy

For multi-hop reprogramming, time to reprogram the network is the time interval between the instant t_0 when the BN sends the first advertisement packet to the instant t_1 when the last node (the one which takes the longest time to download the new application) completes downloading the new application.Time to reprogram the network using single hop method is $R=N*ts$ where N is the number of nodes in the network and t_s is the time to reprogram a single node. Of course, we do not include the time required by the user to move from one node to another since such travel times differs from deployment to deployment. To compare the reprogramming times for single and multi-hop approaches for a given sensor network deployment, one should add these travel times to the single hop reprogramming times mentioned in this paper. Alternately, the reprogramming of the nodes can be done concurrently through multiple base stations at a higher resource cost. Among the various factors that contribute to the energy used in the process of reprogramming, two important ones are the amount of radio transmissions in the network and the number of flash-writes (the downloaded application is written to the external flash as image 1). Since the radio transmissions are the major sources of energy consumption and the number of writes to the external Flash is the same in the two cases (DStream-SHM and DStream-MHM), we take the total number of packets transmitted by all nodes in the network as the measure of energy used in reprogramming.

5.2 Testbed Description

We perform the experiments using Mica2 nodes having a 7.37 MHz, 8 bit microcontroller; 128KB of program memory; 4KB of RAM; 512KB external flash and 916 MHz radio transceiver. Testbed experiments are performed for three different network topologies: grid, linear and actual CSOnet networks (Figure 2). For each network topology, we define neighbors of a node n_1 as those nodes which can receive the packets sent by n_1. In our testbed experiments, if a node n_1 receives a packet from a node n_2 which is not its neighbor, the packet is dropped. Otherwise if n_1 and n_2 are neighbors, n_1 generates a random number u uniformly distributed in the interval [0,1] and if $u<L_{RM}$, then n_1 accepts the packet, otherwise the packet is dropped. This emulates different link reliabilities, since it is difficult to generate experimental conditions with exact link reliabilities. For the grid network used in our experiments, the transmission range R_{tx} of a node satisfies $\sqrt{2}d < R_{tx} < 2d$, where d is the separation between the two adjacent nodes in any row or column of the grid. For the linear networks, $d<R_{tx}<2d$. For multi-hop reprogramming of grid network, a node situated at one corner of the grid acts as the BN while the node at one end of the line is the BN for linear networks. For DStream-SHM, the link reliability of the single wireless link from the user to the one node being reprogrammed is kept constant (0.95) in the experiments. In practice, this is a high value since the user can get close to the node with the BN and there is no other transmission going on. For example, in CSOnet networks, the sensor nodes are situated on top of the traffic posts and the user can go close to the traffic post to do single-hop reprogramming of that node. In DStream-MHM, the link reliabilities L_{RM} of all links are identical and we vary it from 0.6 to 1.0 (perfect link).The link reliabilities shown in Figure 2 are derived from data collected over a summer period by doing a ping test with two radios with no other traffic in the CSOnet network. Sensor networks are well known to experience variation in link qualities—both temporally and spatially. The two CSOnet networks (Figure 2) are just one time snapshot of the network. The effect of temporal variation can also be studied by taking another snapshot of the network.

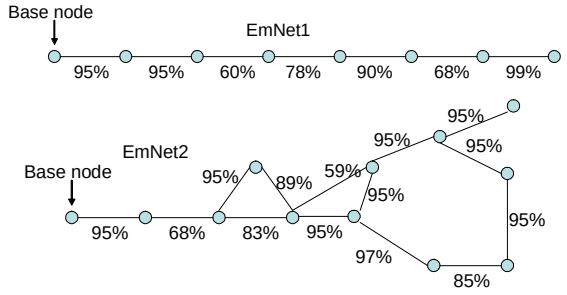

Figure 2: Two CSOnet networks: EmNet1 and EmNet2

5.3 Testbed Experiment Results

Figure 3-a and Figure 3-b compare the average reprogramming time and energy for 2x2, 3x3 and 4x4 grid networks using DStream-SHM and DStream-MHM with different values of link reliabilities. These figures show that multi-hop reprogramming takes more time and energy to reprogram the network if link reliability is decreased because of more retransmissions (and hence more time) required for a packet to be successfully received by the sensor node. Figure 3-a shows that in small networks (2×2 in the experiment), for L_{RM}<0.8, single hop reprogramming is faster than multi-hop reprogramming. However, for larger networks, DStream-MHM is always better for the range of L_{RM} (0.6-1.0) considered in these experiments. But it should be noted that even in large grids, if we carry out the experiments for link reliabilities less than 0.6, then below some value r_t, single hop becomes faster than multi-hop reprogramming. Figure 3-b shows that there exists some value of link reliability L_{RM} >0.6 for which multi-hop reprogramming takes less energy than single hop reprogramming. For good link reliabilities, multi-hop approach is faster and more energy efficient than single hop because of the following reasons: 1) Multiple listening nodes: In multi-hop reprogramming, a single broadcast of the data packet by a node can be received by all its neighbors simultaneously. On the other hand, in single hop reprogramming, a single broadcast of the data packet is received by only one node at a time. 2) Spatial multiplexing: In multi-hop reprogramming, spatial multiplexing of the code transfer makes reprogramming faster. Note that spatial multiplexing contributes in reducing the reprogramming time, not the energy. As link reliability decreases, the difference between single and multi-hop approaches in terms of both reprogramming time and energy decreases and for $r < r_t$, single hop reprogramming becomes faster and for $r < r_e$ single hop reprogramming is more energy efficient. An experimental observation is that $r_t \neq r_e$ in general; thus system designers have to make a decision depending on which metric is more important, energy or delay. In linear networks, the only advantage that multi-hop reprogramming has over single hop reprogramming is spatial multiplexing of the code transfer. By definition, a single broadcast cannot satisfy more than one node in linear networks and thus this factor cannot provide an advantage to DStream-MHM. Hence as shown in Figure 3-c and Figure 3-d, the advantage of DStream-MHM over DStream-SHM is not as pronounced as in grid networks. Further, spatial multiplexing helps to make reprogramming faster but does not contribute in reducing the reprogramming energy. As a result, as shown in Figure 3-d single hop reprogramming is always more energy efficient than multi-hop reprogramming for linear networks. Since spatial multiplexing of the code transfer is effective for larger networks, multi-hop reprogramming incurs less delay than single hop reprogramming for large networks (for example in Figure 3-c, for networks having at least 4 nodes) for good link reliabilities.

Figure 3-e and Figure 3-f compare reprogramming time and energy for the two CSOnet networks (Figure 2). Since EmNet1 is a linear network, reprogramming energy for EmNet1 is always less for single hop case than the multi-hop case. Reprogramming time of EmNet1 is also less for single hop reprogramming than multi-hop reprogramming because some link reliabilities are very low (e.g. 60% and 68%). Even though multi-hop reprogramming for EmNet1 has the advantage of spatial multiplexing of the code transfer which helps to reduce the reprogramming time, the disadvantage due to low link reliabilities outweighs this advantage. For EmNet2, multi-hop reprogramming is faster than single hop reprogramming because multiple listening nodes can receive the single broadcast of the data packet simultaneously and spatial multiplexing of the code transfer make multi-hop reprogramming faster. The reprogramming energy for single and

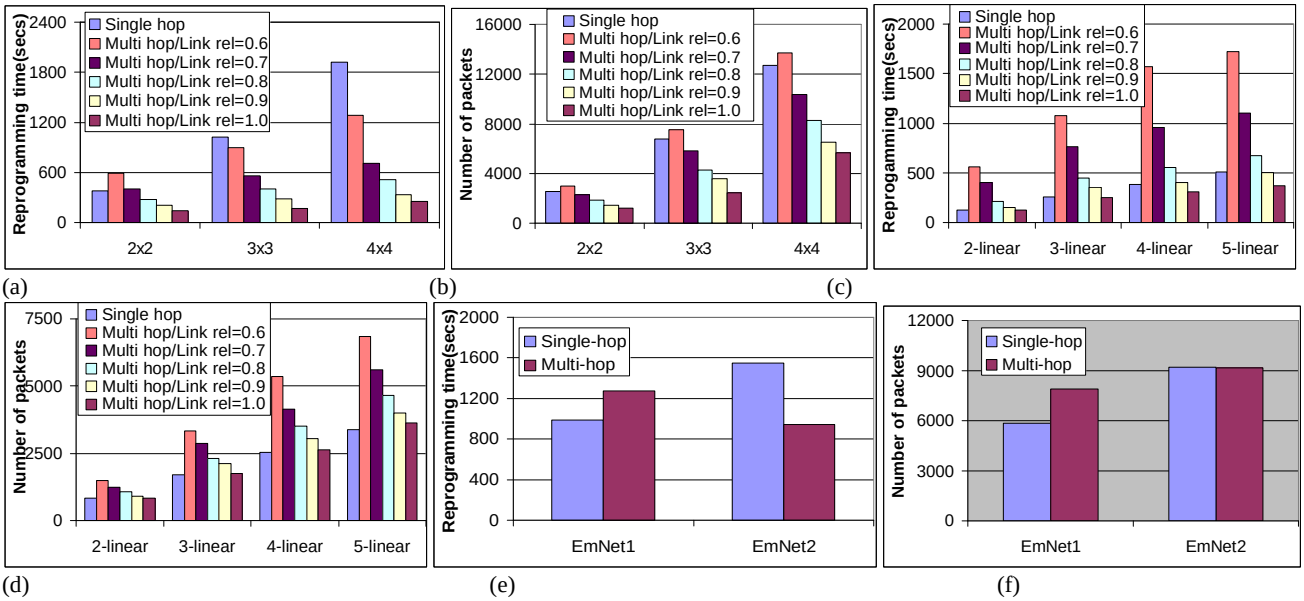

Figure 3: Testbed results. Reprogramming time for (a) grid, (c) linear and (e) CSOnet networks. Number of packets transmitted during reprogramming for (b) grid, (d) linear and (f) CSOnet networks. The leftmost bar is reprogramming time for single hop and the remaining bars are multi-hop reprogramming times with increasing link reliabilities. The order of the legends is the order of the bars from left to right.

multi-hop reprogramming are almost equal for EmNet2.

We can conclude that for linear networks (or networks which are approximately linear, i.e. most of the nodes have degree 2) single hop reprogramming is always more energy efficient than multi-hop reprogramming and except for very high link reliabilities among the nodes, single hop method is also faster than multi-hop method. On the other hand, multi-hop reprogramming is faster and more energy efficient for reasonable link reliabilities in grid networks, with the advantage increasing with network size. However consider that for practical deployments other factors, such as travel times may be added to the cost of DStream-SHM.

5.4 Simulation Results

We used TOSSIM simulator to examine the trend of overhead energy and reprogramming time for larger sized networks beyond the size of our testbed. We perform simulations for three different network topologies: grid, linear and random. The random topology is generated by uniformly distributing nodes with some given density over a square field. Figure 4-a to Figure 4-d compare DStream-SHM and DStream-MHM for linear and grid networks with L_{RM} = 0.9 and L_{RS}=0.95. These results confirm with the analytical and testbed results. The performance of multi-hop reprogramming improves as the network density increases. This is due to the increase in the number of nodes that can listen to the single broadcast of the code packet as the network density increases. For a random network, multi-hop reprogramming is always faster and gets better as the multi-hop link reliability increases-again due to the pipelining of the code in multi-hop reprogramming. Figure 4-h shows that overhead energy of single hop reprogramming is lower than that of multi- hop reprogramming when the link reliability is less than or equal to 0.7. Below a link reliability of 0.7, the number of the nodes that can simultaneously receive the single broadcast of the code packet is not enough to compensate for the lower reliability.

6. CONCLUSION

Complementary to the prevalent idea explored in wireless reprogramming protocols, this paper posits that single hop ogramming can be a better choice under specific network conditions. To identify the conditions which favor single hop reprogramming, we performed mathematical analysis, testbed experiments (including experiments on real-world sensor networks) and simulations. If the network is linear or approximately linear, single hop reprogramming is favored in terms of energy. For smaller linear networks, single hop is faster than multi-hop if link reliabilities are poor. Our testbed results show that for a linear network consisting of 5 nodes, single hop is faster if link reliability is less than 0.9. Even for larger networks, if some of the links are very unreliable (as in the CSOnet deployments), single hop can be faster than multi-hop reprogramming. However as the network size increases, multi-hop improves relative to single hop since pipelining becomes more efficient. For non linear networks, unless the link reliabilities are very poor, multi-hop reprogramming is both more energy efficient and faster than single hop. But single hop is worth considering if some links are really unreliable. The exact cross-over link reliability below which single hop outperforms multi-hop depends on what metric we are interested in. If it is reprogramming time, then the cross-over value is lower than that for reprogramming energy. With increasing density, multi-hop performs better since

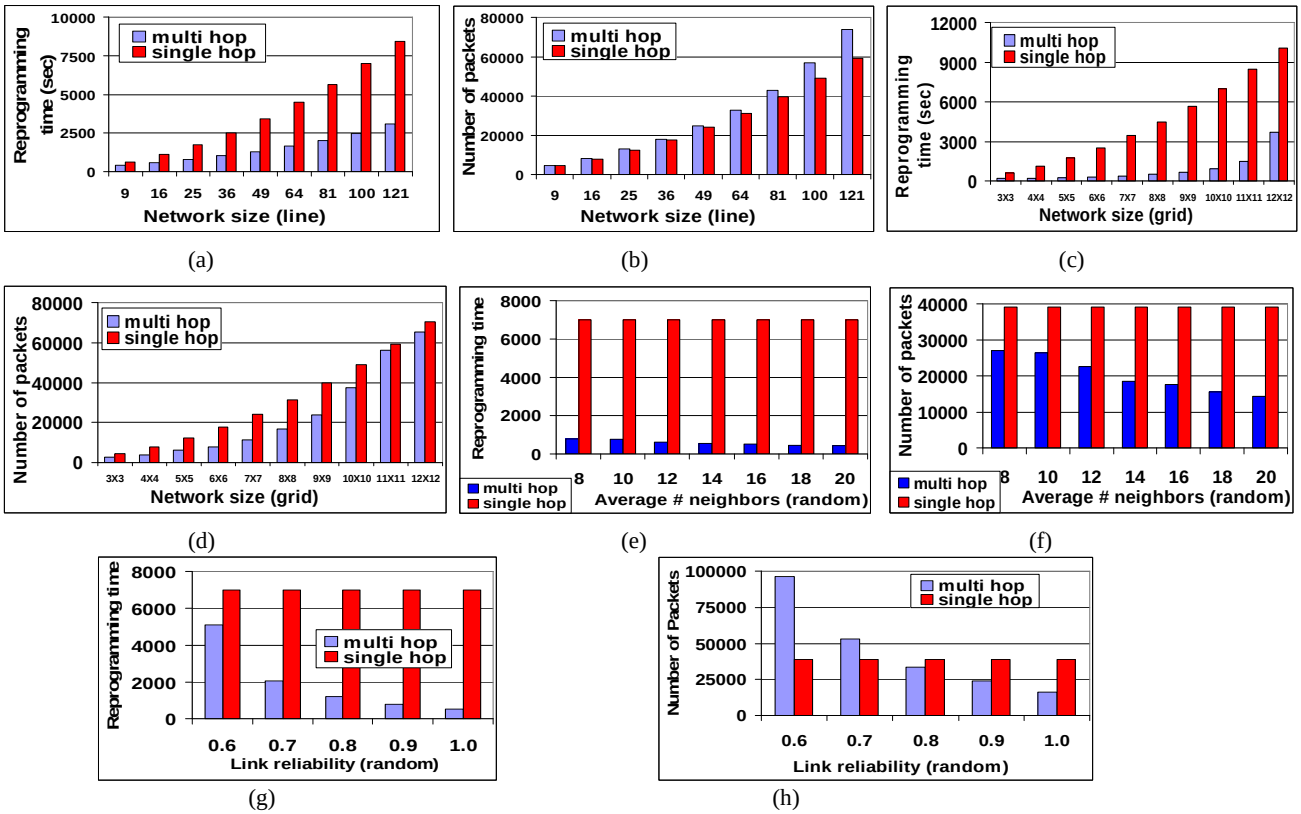

Figure 4: Simulation results. Reprogramming time as a function of network size for (a) linear and (c) grid networks (LRM=0.9). Number of transmitted packets as a function of network size for (b) linear and (d) grid networks (LRM=0.9). For random topology, (e) reprogramming time and (f) number of transmitted packets as a function of network density (LRM=0.9); (g) Reprogramming time and (h) number of transmitted packets as a function of link reliability for 100-random topology (Mean number of neighbors=8). The multi hop result bar is to the left of the single hop result bar.

more number of nodes can be satisfied by a single broadcast of the code image. Also, this reaffirms the claim of Stream and Deluge that they are able to handle high network densities by appropriate collision arbitration schemes.

We are performing work currently on supporting reprogramming in heterogeneous networks, including for nodes that have multiple channels as in wireless mesh networks.

7. REFERENCES

[1] Crossbow Tech Inc., "Mote In-Network Programming User Reference," http://www.tinyos.net/tinyos-1.x/doc/Xnp.pdf.

[2] Couto,D.S.J. De, Aguayo, D. , Chambers B.A. and Morris, R. Performance of multi-hop wireless networks: Shortest path is not enough. HotNets, 2002.

[3] EPA, "National Pollutant Discharge Elimination System", Combined Sewer Overflow Demographics, At: http:/cfpub1.epa.gov/npdes/cso/demo.cfm?program_id=5.

[4] Hui, J. and Culler, D. The dynamic behavior of a data dissemination protocol for network programming at scale. Sensys, 2004.

[5] Krasniewski, M.D., Panta, R.K., Bagchi, S., Yang, C.L., and Chappell, W.J. Energy-efficient,On-demand Reprogramming of Large-scale Sensor Networks. ACM TOSN,2008.

[6] Kulkarni, S. S. and Limin, W. MNP: Multi-hop Network Reprogramming Service for Sensor Networks. At IEEE ICDCS pp. 7-16, 2005.

[7] Levis, P., Lee, N., Welsh, M. and Culler, D. TOSSIM: Accurate and scalable simulation of entire tinyos applications. At the Proc. of SenSys, 2003.

[8] Levis, P., Patel, N., Shenker, S. and Culler, D. Trickle: A Self-Regulating Algorithm for Code Propogation and maintenance in Wireless Sensor Network. At the Proc. of the First USENIX/ACM NSDI, 2004.

[9] Panta, R.K., Khalil, I., Bagchi, S. Stream: Low overhead Wireless Reprogramming for Sensor Networks. At INFOCOM, 2007.

[10] Panta, R.K., Khalil, I., Bagchi, S. and Montestruque, L. Single versus Multi-hop Wireless Reprogramming in Sensor Networks. Purdue ECE Technical Report 08-04, 2008.

[11] Ruggaber,T.P. and Talley,J.W., "Detection and Control of Combined Sewer Overflow Events Using Embedded Sensor Network Technology" Proceedings of the World Environmental and Water Resources Congress, 2005.

[12] Stathopoulos, T., Heidemann, J. and Estrin, D. A remote code update mechanism for wireless sensor networks. Technical Report CENS Technical Report 30, no., 2003.

[13] Wang, Q., Zhu, Y., Cheng, L. Reprogramming wireless sensor networks: challenges and approaches. Network, IEEE, Vol.20, Iss.3, 2006.