ReMo : An Energy Efficient *Re*programming Protocol for *Mo*bile Sensor Networks

Pradip De, Yonghe Liu and Sajal K. Das
Center for Research in Wireless Mobility and Networking(CReWMaN)
Department of Computer Science and Engineering
University of Texas at Arlington, TX 76019-0015
{pradipde, yonghe, das}@cse.uta.edu

Abstract

Existing code update protocols for reprogramming nodes in a sensor network are either unsuitable or inefficient when used in a mobile environment. The prohibitive factor of uncertainty about a node's location due to their continuous movement coupled with the obvious constraint of a node's limited resources, pose daunting challenges to the design of an effective code dissemination protocol for mobile sensor networks. In this paper, we propose ReMo*, an energy efficient, multihop reprogramming protocol for mobile sensor networks. Without making any assumptions on the location of nodes, ReMo uses the LQI and RSSI measurements of received packets to estimate link qualities and relative distances with neighbors in order to select the best node for code exchange. The protocol is based on a probabilistic broadcast paradigm with the mobile nodes smoothly modifying their advertisement transmission rates based on the dynamic changes in network density, thereby saving valuable energy. Contrary to previous protocols, ReMo downloads pages regardless of their order, thus, exploiting the mobility of the nodes and facilitating a fast transfer of the code. Our simulation results show significant improvement in reprogramming time and number of message transmissions over other existing protocols under different settings of network mobility.*

Keywords

Code Dissemination, Network Reprogramming, Mobile Sensor Network.

I. Introduction

Special features of wireless sensor networks [1], as opposed to traditional networks, like place of deployment, scale, node density, resource constraints, etc., pose new problems to the issue of re-programmability of the nodes. The fact that individual sensor nodes, once deployed, are practically inaccessible, necessitates the need for a protocol that can reprogram the whole network over the air. Not only should these protocols be able to fulfill this functionality, but they also need to be reliable and robust against different adverse network conditions, efficient in terms of speed, scalable in terms of both network size and code size propagated, etc. In light of these motivating factors, several network programming protocols [2–5] have evolved in recent years for updating code across the whole network. The dissemination mechanism primarily revolves around periodic advertisements of code metadata in the neighborhood and, when a conflict is discovered, the code pages are propagated in an ordered manner across the network in a pipelined spatially multiplexed fashion. The reason behind maintaining page order in the pipelined transfer is the notion that, in a static network, the code is propagating from a source in the form of a wave. Therefore, in the absence of any kind of unicast routing, there is very low probability of pages arriving at a node out of order. Thus, these protocols reduce the overhead of nodes contending for different pages by enforcing an ordered transfer.

Several sensor networks applications, however, require nodes to be mobile. Despite the new challenges posed by the mobility factor, it also provides a number of advantages. It increases the extent of coverage of the whole area. It also increases the reliability of coverage of a given point and more importantly, a lesser number of sensors can achieve the desired sensing performance. At the same time, from a reprogramming perspective, the random and continuous mobility of the nodes could render the feature of acquiring pages in-order, as the existing protocols for static sensor networks are designed, inefficient. Furthermore, node mobility infuses considerable uncertainty on the location of a node at any given time. Thus, a code update protocol running at a node should not only consider the issue of optimally utilizing the node's resources, but also engage in tackling the uncertainty brought about by the nodes' mobility. In other words, nodes have to optimally choose when and which neighbor to request a page transfer from, such that redundant and useless transmissions are reduced as much as possible and the time required for the code update is minimized.

One could trivialize the requirement of this protocol for mobile sensor networks arguing that the nodes could be temporarily immobile and use the reprogramming protocol for static networks. They could also move towards the base station and reprogram themselves. However, these two scenarios might not be feasible in most applications. First of all, a mobile sensor is generally constructed by mounting an off-the-shelf wireless sensor on a mobile entity where the sensor has no robotic control over the mobility of the device. An example could be the monitoring of animals where sensors are embedded on them to monitor their moving patterns or biomedical sensors mounted on people for health monitoring. Thus, it is necessary to reprogram the sensors while they are moving around according to the independent mobility pattern of the moving objects on which they are mounted. Moreover, even if the sensor had robotic control over the mobile device and could steer it towards the base station to get itself updated, it could be wasting valuable energy. Instead, a protocol that considers the independent and unconstrained mobility of the sensor devices would be more effective.

In this paper, we propose *ReMo*, a reprogramming protocol specifically designed for mobile sensor networks. Without assuming nodes know their respective locations, ReMo tries to infer relative distance with neighboring nodes and the link qualities with them from parameters measured from received packets. Particularly, contrary to previous reprogramming protocols, in ReMo, we relax the constraint of in-order propagation of pages and try to take advantage of the mobility by allowing nodes to download pages out-of-order. We address two main issues for code exchange between neighbors. First, nodes try to ascertain which neighbor has the best link quality as well as a higher chance of staying within communication range for a longer duration. Secondly, a node also tries to ascertain which neighbor has the highest potential for providing pages for download. The primary local goal of ReMo running at a node is to optimally choose a neighbor based on the above two requirements while simultaneously trying to minimize its energy usage by intelligently optimizing the number of control messages transmitted in a neighborhood. Since each node takes local decisions based only on neighborhood information, ReMo can scale to large network sizes. Moreover, the optimal choice of a neighbor to exchange pages with, makes it extract the most out of the uncertain mobile environment and thus, efficient in terms of speed of propagation. Furthermore, the epidemic style of page propagation automatically achieves the desired reliability to correctly disseminate the code to the entire network.

The rest of the paper is organized as follows. Section II presents related work. In section III, we discuss about parameters crucial for estimating the link qualities and relative distances between nodes. In section IV, we analyze the effect of mobility on Deluge. In section V, we present the ReMo protocol in detail. In section VI, we discuss the simulation and corresponding results and conclude in section VII.

II. Related Work

Code update protocols in sensor networks have been given special focus in recent years. All these protocols have concentrated on reprogramming a set of static nodes scattered in a terrain. Trickle [2] is a code maintenace algorithm which works on a polite gossip based periodic advertisement of metadata. Deluge [4] is based on Trickle and is meant for transferring bulk code by dividing it into fixed size pages and pipelining the pages across the network.

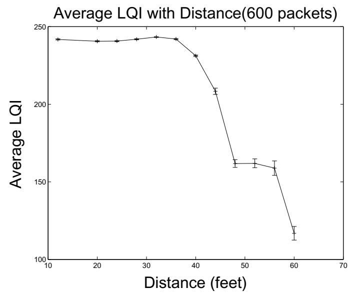
However, Deluge suffers from the hidden terminal problem in dense networks. MNP [5] improves on Deluge's hidden terminal problem by implementing a sender selection algorithm in a neighborhood by which one single node would transmit data. It transfers the whole image in a phase-by-phase manner across single hops. Although it saves on energy, the code propagation process takes longer. An extension of MNP was proposed in Gappa [7] where parts of a code can be communicated to a subset of sensors on multiple channels. Although most of these protocols provide a reliable transfer of data across the network and can handle occasional node failures, they are unsuitable or inefficient when used in a mobile sensor network scenario where nodes are dynamic all the time. Probabilistic broadcast models have been proposed in [6]. Probabilistic broadcasting in a mobile environment have been dealt with in [8]. In [9], the authors proposed a dynamic scheme to change the rebroadcasting probability based on node distribution and movement.

III. Link Quality and Relative Distance Estimate

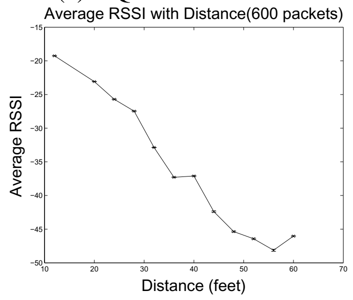
Since the mobile nodes are not assumed to know their own location, it is important to determine not only the relative distance between two neighbors but also the link quality between them in terms of the packet reception rate. These two parameters would help a node to select the best neighbor for code download. We have performed some outdoor experiments to characterize the link between two sensor nodes with 802.15.4 radios at varying distances. Our goal was to find out how the RSSI and LQI values of a radio packet varied with distance and correlated with the packet reception rate over the corresponding link.

For our experiments on link quality measurements, we used the *Sun Small Programmable Object Technology* (SunSPOT) nodes [10]. A SunSPOT has a 180 Mhz 32 bit ARM920T processor with 512K RAM and 4M Flash. The radio chip is the CC2420 [12].

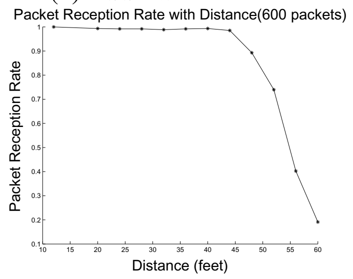
As part of the ReMo protocol, each node would continuously snoop on all radio packets transmitted in the neighborhood in order to construct important statistics like relative distance and link quality with its neighbors. However, snooping is not cheap since nodes have to listen for packets that are not necessarily addressed to itself. Several low-power listening techniques exist[14] that would allow nodes to snoop at a much lower cost. However, much of the traffic in the case of ReMo is broadcast and thus nodes would not need to do any extra snooping in order to construct the link characteristics with their neighbors.



(a) LQI with Distance



(b) RSSI with Distance



(c) Packet Reception Rate with Distance

Fig. 1. Outdoor Measurements of LQI, RSSI and Packet Reception Rate with Distance

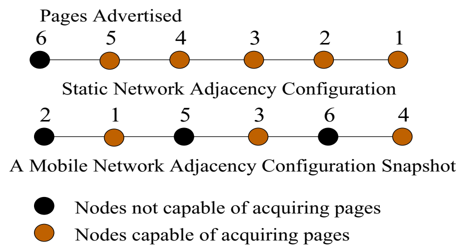
As shown in Fig. 1, we measured the average RSSI and LQI values of 600 packets for a distance of 12 feet to 60 feet at steps of 4 ft in an open area. The nodes were kept at a height of 3 ft above the ground. Our observations indicate that RSSI is a good indicator of relative distance between two nodes but it is not correlated with the packet reception rate as accurately as the average LQI values. Thus, ReMo uses the average LQI values as an indicator of link quality and the RSSI measurements as indicators of relative distance.

IV. Deluge in a Mobile Sensor Network

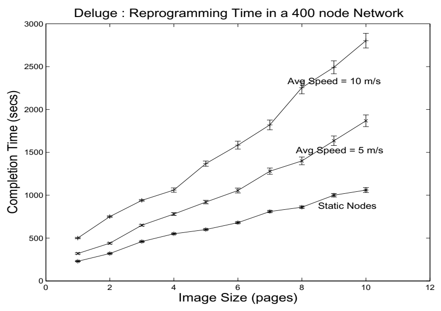
In order to demonstrate the requirement of a suitable reprogramming protocol for mobile sensor networks, we evaluated the performance of Deluge [4], a popular reprogramming protocol, in a network with mobile nodes. Most other existing protocols [2, 3, 5] follow a similar paradigm for code propagation. We have performed simulation experiments under varying degrees of node mobility. Our primary goal was to observe the completion time of the code update for the entire network under varying average speeds for the mobile nodes. Our results indicate that Deluge shows a significant increase in network programming time as the average speeds of the mobile nodes are increased. One of the primary reasons behind this fall in performance is that Deluge distributes pages strictly in order. This strict page-ordered download makes sense in a static scenario where the new code propagates from a specific source in a wave across the network. Thus, there is a very low probability of pages arriving at a given node out of order. However, in a mobile scenario, this constraint no longer holds and a node might have neighbors potentially capable of providing arbitrary pages for download.

Moreover, since Deluge has a simplistic rate control mechanism for its metadata advertisements, it is unsuitable to the dynamic changes in neighborhood density brought about by the mobility of the nodes. Moreover, it is very susceptible to the hidden terminal problem when the neighborhood density increases.

Fig 2 (a) is a snapshot of a network running Deluge with adjacent nodes connected by an edge. The first set of nodes shows the normal operation of Deluge in a static network where the code update propagates in a particular direction. Each node is marked with the page number it is advertising. Thus all the nodes having a page number of 5 or lower can acquire missing pages from their neighbor nearer to the source. The next array of nodes shows a mobile scenario where nodes have moved into a new adjacency configuration. Thus, the number of nodes potentially able to download new pages are less. This example proves that the regular paradigm of code propagation in an increasing order of pages in unsuitable when the nodes are mobile. Fig. 2 (b) shows the effect of mobility on the completion time


Pages Advertised
6 5 4 3 2 1
Static Network Adjacency Configuration
2 1 5 3 6 4
A Mobile Network Adjacency Configuration Snapshot
Nodes not capable of acquiring pages
Nodes capable of acquiring pages

(a) Downloaded/Required Pages of mobile nodes



(b) Completion Time of Deluge in a mobile sensor network

Fig. 2. Deluge in a Mobile Sensor Network

taken by Deluge with increasing image size in a 400 node network in a square of area $4000m^2$. Each page is of size 1 KB.

V. The *ReMo* Protocol

Mobility among nodes poses new challenges to the efficient design of a reprogramming protocol for sensor networks. Sensors, with their stringent resource constraints have more responsibility in not only tackling the uncertainty due to mobility, but also minimizing the resource utilization in overcoming that overhead and propagate the code throughout the network as fast as possible. In this section, we first introduce a few assumptions and design directions of ReMo before describing the working principle of the protocol.

A. Node Mobility Model

The most commonly used mobility model in ad hoc networks is the *random waypoint* (RWP) *model*[15]. In this model, a node in a movement period i, randomly chooses a waypoint P_i, moves towards it at a velocity V_i chosen uniformly randomly, and pauses at the waypoint P_i for a random period T_i. This process is repeated for each subsequent movement period.

In our mobile sensor network, we assume that nodes move according to the RWP model. Moreover, nodes are *not* required to be aware of their locations. We do not make any assumptions on network connectivity being maintained at all times but assume that the nodes move around within the confines of the region.

B. Data Representation

Similar to the data representation of Deluge [4], ReMo divides the code image into fixed sized packets of size S_{pkt}. The protocol uses a basic unit of transfer called a *page* which is of size $N.S_{pkt}$ where N is a fixed number of packets. Breaking the code image into pages enables pipelining the transfer of the file over multiple hops across the network. A *version number* is used to distinguish between different code updates and must be monotonically increasing to maintain an order for all updates. A node needs to compare version numbers to decide on whether it requires an update.

C. Page Download Potential (PDP)

We define the page download potential (PDP) ω_j^i of node i w.r.t node j as a measure of the potential that j has in providing pages to i. Let the code image C_{img} consist of κ pages, such that $C_{img} = \{P_1, P_2, \ldots, P_\kappa\}$ where P_l is the l^{th} page. Let S_i and S_j denote the set of pages of node i and node j, respectively. Then the page download potential ω_j^i of i w.r.t j is denoted by

$$\omega_j^i = \frac{|S_j - S_i|}{|C_{img}|} \tag{1}$$

where $S_j - S_i$ is the set difference between the page sets of node j with node i. Thus, for each node i with σ neighbors, the page download potential vector Ω is denoted by

$$\Omega_i = \left\{\omega_1^i, \omega_2^i, \ldots, \omega_\sigma^i\right\} \tag{2}$$

D. Neighbor Link Profile (NLP)

Each node i snoops in its neighborhood for packets transmitted by other nodes in order to build a Neighbor Link Profile (NLP) vector Φ_i. ϕ_j^i, which is an element of Φ_i, represents an entry of the NLP of node i w.r.t. node j. It is a 2-tuple $< lq_j^i, dm_j^i >$ where lq_j^i is a normalized representation $\in [0,1]$ of the link quality with node j, and dm_j^i is a direction of motion indicator. lq_j^i is calculated based on the average of the LQI values of packets received from node j. The average LQI values, as depicted in Fig 1, have been used for estimating the link quality with a given neighbor as they have a better correlation with the packet reception rate of a link. The link quality estimate is updated based on a window mean exponentially weighted moving average (WMEWMA) of the LQI values in each slot of a neighbor node, the window being the duration τ of each slot. Thus, the link quality update of node j is given by

$$lq_j^i(t+1) = \gamma lq_j^i(t) + (1-\gamma)\frac{lq_j^{avg}(t)}{lq_{max}} \tag{3}$$

where $lq_j^{avg}(t)$ is the average of the LQI values of the packets received from node j in the current slot. The weight factor γ decides the contribution of the previous estimate of the link quality.

On the other hand, dm_j^i is calculated based on RSSI values of multiple packets from the same node. The RSSI values of the $802.15.4$ packets from the $CC2420$ [12] radio chip have been confirmed to have shown an agile linear correlation with the distance as depicted in Fig 1. Accordingly, consecutive values have been used to approximately indicate whether a neighboring node is approaching or departing. Thus, dm_j^i takes values -1(departing), 0(uncertain), and $+1$(approaching). However, the measured RSSI values need to be sufficiently spaced in time in order to reflect a change in position of the mobile node. Accordingly, RSSI values are taken from packets that are at least spaced by a time duration of $\frac{\Delta}{\varphi}$. where Δ is a constant distance (say 8m) and φ is the average speed of a node.

If no packet is heard from an existing neighbor in the NLP in the last time slot, it is marked as *stale*. Moreover, a *stale* neighbor is again made *fresh* when a packet is received from it. After a maximum number of W slots with a neighbor remaining *stale*, it is evicted from both the NLP and the PDP lists.

E. Probability of Metadata Broadcast

One of the central features on which ReMo is based is an adaptive metadata advertisement scheme. This is a probabilistic technique for broadcasting metadata in the neighborhood considering local node density and the presence of new metadata. Time is divided into slots of fixed duration τ, and nodes essentially broadcast their metadata at every slot t based on their current advertisement probability β_t.

In this section, we formulate β_t, which is the probability of metadata broadcast by a node at each time slot t. Each node dynamically updates β_t at each time slot based on gathered observation at the previous time slot. Not only is the computation of this transmission probability based on the presence of new code information in the neighborhood, but also on the relative density of the current neighborhood. Thus, each node tries to proactively ascertain if there is any neighbor with new code and also control the level of gossip in order to avoid collisions and packet loss.

Let N_t^d and N_t^s, respectively, denote the number of *different* and *similar* metadata advertisements heard by a node during slot t. Morever, let A_t denote the sum of all advertisement messages heard by the node in slot t. Thus, $A_t = N_t^d + N_t^s$. Algorithm 1 depicts the update of the broadcast probability β at each time slot. The notion is to allow a node to increase its broadcast probability whenever it senses that there is new code in the neighborhood. This increase is inversely proportional to the number of nodes advertising new code. In other words, nodes increase their probability by a smaller amount when the number of neighbors advertising new code is high and vice versa. Moreover, in order to avoid packet collisions, nodes decrease their probability of transmission if N_t^d and β_t cross a threshold value of N_{Th} and β_{high}, respectively.

More importantly, nodes keep track of the number of neighbors that are advertising the same metadata, N_t^s. They, accordingly, decrease their transmission probability in order to minimize redundantly broadcast metadata in the neighborhood. δ denotes a small step used to increase or decrease the probabilities.

Thus, for each slot t, nodes gather these information about their neighbors and update their advertisement probability for the next slot $t+1$.

For the initial transmission probability β_0, each node is assigned a value of $\beta_0 = \frac{1}{\rho}$ where $\rho = \frac{\pi R^2 N}{D}$ denotes the average number of neighbors that a node has within its transmission radius R. D denotes the area in which the nodes are deployed and N is the total number of nodes..

Algorithm 1 TransmissionProb β_{t+1} for $(t+1)^{th}$ slot

Input: $\beta_t, N_t^d, N_t^s, \delta, slots_{NoADV}, max_{NoADV}, maxNbrs$
Output: β_{t+1}

```
 1: Initialize $slots_{NoADV} = 0$;
 2: At the Expiry of the $(t+1)^{th}$ Timer
 3: Set $N_{t+1}^d = 0$;
 4: Set , $N_{t+1}^s = 0$;
 5: if $A_t > 0$ then
 6:     if ($N_t^d > N_{Th}$ and $\beta_t > \beta_{high}$) then
 7:         $\beta_{t+1} = \frac{N_t^d}{A_t}\beta_t\left(1 - \frac{N_t^d}{A_t}\delta\right) + \frac{N_t^s}{A_t}\beta_t(1-\delta)$;
 8:     else
 9:         $\beta_{t+1} = \frac{N_t^d}{A_t}\beta_t\left(1 + \frac{\delta}{N_t^d}\right) + \frac{N_t^s}{A_t}\beta_t(1-\delta)$;
10:     end if
11: else
12:     $slots_{NoADV} = slots_{NoADV} + 1$;
13:     if ($slots_{NoADV} < max_{NoADV}$) then
14:         $\beta_{t+1} = \beta_t(1+\delta)$;
15:     else
16:         Sleep for a short duration;
17:         Set wakeup broadcast probability $\beta_0 = \frac{D}{\pi R^2 N}$;
18:     end if
19: end if
20: if ($\beta_{t+1} > 1$) then
21:     $\beta_{t+1} = 1$;
22: end if
23: if ($\beta_{t+1} < 0$) then
24:     $\beta_{t+1} = 0$;
25: end if
```

We note that a node reacts by decreasing or increasing β as described above, based on corresponding advertisement counts received in the last time slot. When a node does not receive any advertisements in a slot, it assumes that it is either alone or in a very sparse location. In this case, it increases its probability of advertisement transmission in order to reach out to any other node in the vicinity. However, after trying for a threshold max_{NoADV} number of slots with no received advertisements, the node sleeps for a short duration and then wakes up to broadcast at the initial probability of β_0.

As shown in Algorithm 1, we note that, in the situation where a node receives advertisement packets in the last slot, its probability update is composed of a weighted sum of two components. The first component is based on the contribution of advertisements that contain new information, N_t^d, whereas the second one is for advertisements with same metadata, N_t^s. The weights of these two componenets are based on the normalized counts of the corresponding types of advertisement messages. Moreover, we note that decrease of β is done more aggressively for the component related to advertisements with same metadata. The obvious reason is that these advertisements pose as redundant transmissions in the neighborhood and could only cause packet collisions without providing extra information. β is only increased when the node hears new metadata and the number of neighbors are below a required threshold. However, as mentioned before, β is increased cautiously by setting the increase factor inversely proportional to the count of new metadata advertisements so that it does not result in a gossip storm and subsequent packet collisions.

F. Protocol Description

In ReMo, the goal of a node is to periodically keep its neighbors updated on the version of its code and its location information. Whenever a new code is propagating through the network, a mobile node needs to optimally choose a suitable neighbor to download pages from, given the fact that the neighborhood is very dynamic as the nodes are constantly mobile.

Broadly, each node lies in either of three major states, viz., $Advertise(ADV)$, $Receive(RX)$, and $Transmit(TX)$. Fig 3 shows the detailed state transition diagram of the protocol.

In the *ADV* state, a node performs important functions like periodic advertisement of code metadata, neighborhood assessment, and optimal decision making for different actions like choosing an appropriate neighbor to download code from or modifying its advertisement transmission rate based on dynamic information about its current neighbors. In this state, a node broadcasts an advertisement message M_{data} containing some meta information in each slot t of duration τ with a given probability β_t. It selects a random time $\in [\frac{\tau}{2}, \tau]$ for transmitting M_{data} to account for the *short listen* problem [2]. M_{data} is primarily comprised of two components : 1) Version Number, and 2) Downloadable data information which consists of a bit vector $< p_0, p_1, \ldots, p_{k-1} >$ of the k pages of the object image. We also note that the duration τ of each slot has an important role to play in the efficient working of the protocol. The value of τ is generally based on the average velocity of the nodes in the network. For instance, in a network where nodes move at very high speeds, τ is smaller because the neighborhood states change more frequently. Whereas, in networks with relatively slower movement speeds, τ is of a longer duration. At each timer expiry, a node updates its count variables N_t^d and N_t^s and recalculates β_{t+1} for the next slot as described in Algorithm 1.

The *Request* messages are of two types, 1) A *Half Request (HReq)* message, and 2) A *Full Request (FReq)* message. Since nodes do not know their location, and estimate their relative distance and link qualities with their neighbors from received packets, ReMo uses these two types of *Request* messages. When a node is sure that the destination neighbor is close enough for a reliable page download, it sends a *FReq* message, whereas it sends a *HReq* message when it is unsure of the reliability. In the latter case, it is upto the neighbor to respond with the requested page or ignore the *HReq* message. Each *Request* message contains a bit vector indicating the required set of pages. Since both the *Advertisement* and the *Request* packets from a neighboring node contain the page information, the PDP is updated upon hearing any of these packets from the neighbor.

The choice of a neighbor to transmit a *Request* message to, is based on which neighbor has a high page download potential as well as a sufficiently good link quality. Thus, a node i computes $p_j^i = \omega_j^i \cdot lq_j^i$ and selects a neighbor $n_j = \{j | j = \text{argmax}_j(p_j^i)\}$ for sending a *Request* message after transiting to the *RX* state. However, after selection of j, node i sends a *FReq* message to j if $lq_j^i > lq_{thresh}$ and $dm_j^i \neq -1$. Otherwise, a *HReq* message is transmitted. The *Request* message is also broadcast in the neighborhood with the address of node j incorporated as one of the fields of the packet.

If a node in the *ADV* state finds that it has a non-empty PDP and NLP list and it needs code to download, it would transit to the *RX* state at the next timer expiry and send out an appropriate *Request* message targeted at the chosen neighbor and wait for the requested data to be downloaded. For each *Request* message, a node generates a random sequence number and includes it as a field in the message. However, if it also receives a *Request* message in this interim before the timer expires, it would first service the arrived request provided the sequence number in the incoming request message is higher than its own generated sequence number.

A node in the *RX* state would transmit a maximum of R_{max} unserviced requests before transiting back to the *ADV* state.

When a node receives a *FReq* message in the *ADV* state, it transits to the *TX* state and transmits all the packets of the first page of the requested page vector. The recipient node sends a $Data_{ACK}$ message for the page sent. On receiving this acknowledgment, the sender transmits the data packets of the next page in the requested vector provided the link quality of the destination node (as per the last measured estimates) and the *RSSI* of the received $Data_{ACK}$ packet is above the required threshold. However, if no acknowledgment arrives or, the sender transits to the *ADV* state and starts broadcasting its metadata with probability β_0. A recipient node transits to the *REDEEM* state to download missing packets of a page.

Upon receiving a *HReq* message, a sender node migrates to an intermediate *CHECK* state to decide whether the recipient node is suitable for a page transfer by evaluating the link quality and the *RSSI* of the received *HReq* packet. If the link quality is satisfactory and the node is approaching, then the page is transmitted.

The protocol messages with their essential fields are defined as follows:

- **Advertise Message**: (a) ImageName (b) Version Number (c) Image Size (d) Image Page Bit vector.
- **HReq Message**: (a) Requested Version (b) ImageName (c) Sequence Number (d) Destination Address (e) RSSI of Recvd Adv (f) Requested Vector of Pages.
- **FReq Message**: (a) Requested Version (b) ImageName (c) Sequence Number (d) Destination Address (e) Requested Vector of Pages.
- **Data Message**: (a) ImageName (b) Version Number (c) Page Number (d) Packet Index (e) Data Size (f) Data.

VI. Performance Evaluation

We have evaluated the performance of ReMo in comparison with Deluge and MNP using the packet level network simulator GloMoSim [11]. All the three protocols have been implemented at the application layer of GloMoSim. The CSMA MAC protocol model was used with a communication range of $30m$. The terrain is assumed to be a rectangular area of 4000 sq. meters. We have used the Random Waypoint mobility model [15] to evaluate the performance of the protocols with average speeds ranging from 2 to 20 m/s with a maximum pause time of 100 ms. Nodes are initially uniformly randomly deployed in the terrain. Network reprogramming time and the total number of packets transmitted to achieve that were the primary metrics of measurement for evaluating the protocols. Each simulation result was taken over 30 runs with a 95% confidence interval. Table I shows the values of the different protocol parameters for our simulation.

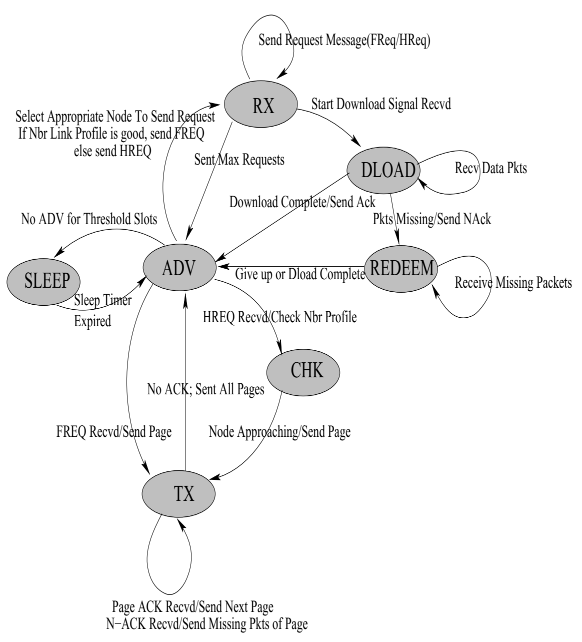

Send Request Message(FReq/HReq)
RX
Start Download Signal Recvd
Select Appropriate Node To Send Request
If Nbr Link Profile is good, send FREQ
else send HREQ
Sent Max Requests
DLOAD
Recv Data Pkts
Download Complete/Send Ack
Pkts Missing/Send NAck
No ADV for Threshold Slots
SLEEP
ADV
Give up or Dload Complete
REDEEM
Receive Missing Packets
Sleep Timer
Expired
HREQ Recvd/Check Nbr Profile
CHK
No ACK; Sent All Pages
FREQ Recvd/Send Page
Node Approaching/Send Page
TX
Page ACK Recvd/Send Next Page
N−ACK Recvd/Send Missing Pkts of Page

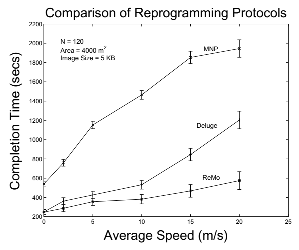
Fig. 3. ReMo State Transition Diagram

TABLE I. Parameter Settings

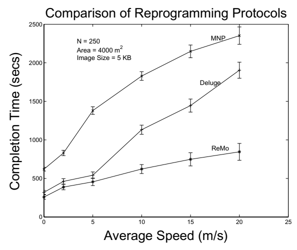
Model Parameter	Value
τ	$\frac{10}{AvgNodeSpeed(m/s)}$ sec
γ	0.4
W	6
β_{high}	0.9
N_{Th}	ρ
δ	0.1
R_{max}	3

A. Reprogramming Completion Time

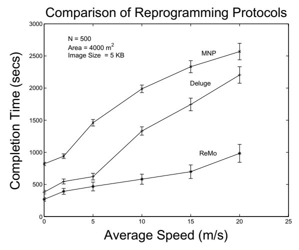
In this section, we focus on the completion time for transferring an image of size 5 pages with each



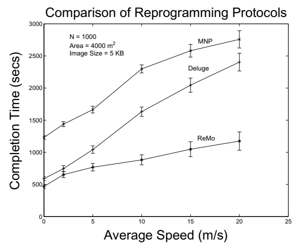
(a) 120 Nodes



(b) 250 Nodes



(c) 500 Nodes



(d) 1000 Nodes

Fig. 4. Completion Time of Deluge, MNP and ReMo in a Mobile Sensor Network

page of size 1 KB comprised of 16 packets of size 64 bytes each. The initially uniformly randomly deployed nodes move with average speeds of 2, 5, 10, 15 and 20 meters per sec $\pm$ 10%. We also vary the number of nodes moving in the fixed area of the terrain. This lets us study the effects of increasing average node density on the performance of the protocol. Comparative results are shown in Fig. 4 for 120, 250 and 500 nodes.

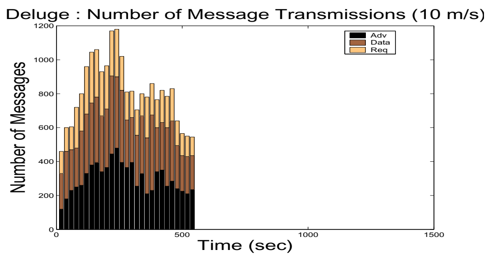
Fig. 4 (a) depicts the effects of a low average node density on the reprogramming completion time. Without mobility, and at this low node density, we observe that both ReMo and Deluge take almost the same time for propagating 5 pages through the network. However, as the mobility of the nodes increases, ReMo outperforms Deluge. One of the primary reasons is Deluge's constraint of downloading pages in order. Moreover, ReMo adapts better to the dynamic changes in local node density in curbing redundant transmissions of advertisement messages thus promoting faster transfer of data.

MNP gets the most adversely affected by node mobility as the sender selection algorithm fails to optimally select a node in a neighborhood. As per the MNP protocol, nodes wait to gather multiple request packets while having the sender selection algorithm choose a particular sender and other competing advertising nodes go to sleep. However, this delay in serving the requests proves useless because when a sender is finally chosen and scheduled to transmit, it is in a different neighborhood. Thus, certain nodes go to sleep when they do not have to, and some nodes end up transmitting in a neighborhood where they are not required. Subsequently, nodes end up sending more messages for a longer duration until the whole network is finally updated. However, we also notice that the slope of the MNP curve decreases at higher mobility. The possible reason is that the effect of mobility on the sender selection mechanism becomes less dependent on the speed of the nodes after a certain value.

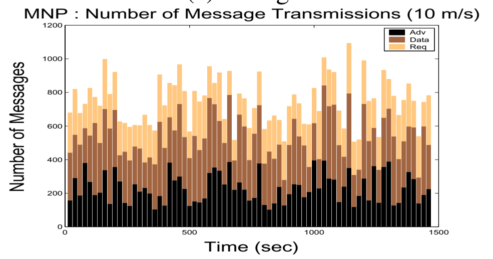
Fig. 4 (b) shows the effects of an increased level of average node density. We observe that even at zero mobility, ReMo performs slightly better than Deluge. The reason is that the hidden terminal problem becomes conspicuous and ReMo's probabilistic advertisement mechanism copes with it better than Deluge. However, we observe that a slight increase in node mobility to 5 m/s helps Deluge cope with the effects of the hidden terminal problem brought about by the increased average node density. Thus, its completion time does not increase significantly and the two curves stay close to each other. However, a further increase of node mobility to 10 m/s and higher causes Deluge to degrade in performance and the difference in completion time with ReMo becomes significant. In Fig. 4 (c) and (d), the average node density is increased further and we observe that ReMo continues to show significant improvements over Deluge and MNP. The flexible order for page download and the smoothened probabilistic advertisement mechanism based on neighborhood density helps ReMo overcome the effects of mobility and node density fluctuations better than Deluge and MNP.

B. Number of Message Transmissions

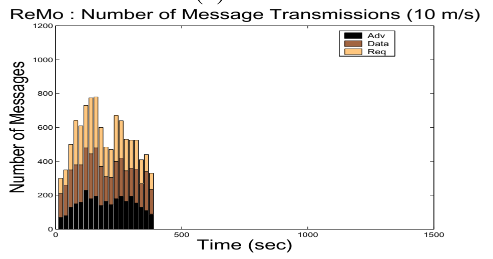
We look into the number of messages transmitted by each protocol in fixed sized time windows of 20 seconds for the entire duration of the code update. Fig 5 depicts the message transmission distribution for each protocol. There are 120 nodes in a $4000m^2$ area and the nodes move with an average speed of 10 m/s $\pm$ 10%. In Fig 5 (a), the overall transmitted



(a) Deluge



(b) MNP



(c) ReMo

Fig. 5. Number of Transmitted Messages (120 Nodes)

messages for Deluge are shown. Comparing with Fig 5 (b), we see that although the average number of messages sent in each 20 second slot was less in MNP, the entire duration was much longer. However, in Fig 5 (c), the number of messages transmitted by ReMo shows that both the average rate of message transmission and the entire duration of update is less than both MNP and Deluge.

The lower average message transmission rate also indicates that ReMo is more energy efficient than Deluge or MNP in a mobile environment.

VII. Conclusions

In this paper, we presented *ReMo*, a reprogramming protocol specifically suited for mobile sensor networks. We showed how the existing reprogramming paradigm for static networks fails to adapt to a mobile scenario. The protocol takes advantage of the mobility of the nodes by having them download pages out-of-order, thus expediting the download process. The protocol also smoothly adapts its periodic metadata advertisements to cope with the constantly varying node density of the mobile environment and suppress redundant transmissions as much as possible, thereby saving valuable energy. Our comparative results indicate significant improvements in completion time of reprogramming the whole network and the number of messages transmitted over existing reprogramming protocols like Deluge and MNP.

As part of our future work, we are implementing the ReMo protocol on a testbed of SunSPOTs.

References

[1] C. Chong and S. Kumar, "Sensor networks: Evolution, opportunities, and challenges", In *Proceedings of the IEEE*, vol. 91, no. 8, 2003.

[2] P. Levis, N. Patel, D. Culler, and S. Shenker. "Trickle: A self-regulating algorithm for code maintenance and propagation in wireless sensor networks", In *First USENIX/ACM Symposium on Network Systems Design and Implementation*, (NSDI) 2004.

[3] P. Levis and D. Culler "The Firecracker Protocol", In *The 11th ACM SIGOPS European Workshop*, 2004.

[4] J. W. Hui and D. Culler, "The dynamic behavior of a data dissemination protocol for network programming at scale", In *The Second ACM Conference on Embedded Networked Sensor Systems*, (SenSys) 2004.

[5] S. S. Kulkarni, L. Wang, " MNP: Multihop Network Reprogramming Service for Sensor Networks", In *The 25th IEEE International Conference on Distributed Computing Systems* (ICDCS), 2005.

[6] P. Kyasanur, R. R. Choudhury, and I. Gupta. "Smart gossip: An adaptive gossip-based broadcasting service for sensor networks", In *The Third IEEE International Conference on Mobile Ad-hoc and Sensor Systems* (MASS) 2006.

[7] L. Wang and S. S. Kulkarni. "Gappa: Gossip based multi-channel repro- gramming for sensor networks". In *Second IEEE International Conference in Distributed Computing in Sensor Systems* (DCOSS) 2006.

[8] Y. C. Tseng, S. Y. Ni, and E. Y. Shih, "Adaptive Approaches to Relieving Broadcast Storms in a Wireless Multihop Mobile Ad Hoc Network," *IEEE Transactions on Computers*, vol. 52, no. 5, pp. 545-557, May 2003.

[9] Q. Zhang and D. P. Agarwal, "Dynamic Probabilistic Broadcasting in MANETs", In *Journal of Parallel and Distributed Computing*, vol. 65, no. 2, pp. 220-233, 2005.

[10] Sun^{TM} Small Programmable Object Technology (Sun SPOT), www.sunspotworld.com.

[11] X. Zeng, R. Bagrodia, and M. Gerla, "GloMoSim: A Library for Parallel Simulation of Large-scale Wireless Networks," In *Workshop on Parallel and Distributed Simulation* (PADS), 1998.

[12] C. Inc. Cc2420 data sheet. *http://www.chipcon.com/files/CC2420_Data_Sheet_1_0.pdf*, 2003.

[13] K. Srinivasan and P. Levis. "Rssi is under appreciated". In *Proceedings of the Third Workshop on Embedded Networked Sensors* (EmNets), 2006.

[14] E. Shih, P. Bahl, and M. J. Sinclair. "Wake on wireless:: an event driven energy saving strategy for battery operated devices". In *Proceedings of the eighth annual international conference on Mobile computing and networking*, pages 160171. ACM Press, 2002.

[15] T. Camp, J. Boleng and V. Davies, "A Survey of Mobility Models for Ad Hoc Networks Research", *Wireless Communications and Mobile Computing*. Volume 2, Number 5. 2002