MuTrans: A Multi-Channel Network Coding Approach to Mobile Data Collection

Mansour Abdulaziz
Department of Computer Science
George Mason University
Fairfax, Virginia 22030
mabdulaz@gmu.edu

Robert Simon
Department of Computer Science
George Mason University
Fairfax, Virginia 22030
simon@gmu.edu

***Abstract*—The advantages of using of the mobile sinks (MSs) to perform data collection from Wireless Sensor Networks (WSNs) are now widely recognized. This is because the MS data collectors can service isolated systems and reduce energy expenditures by minimizing the need for multi-hop networking. However, it remains a challenging problem to support collection activities efficiently within the relevant class of predictable but uncontrollable MSs. This paper presents *MuTrans*, a novel multi-channel protocol that uses both clustering and network coding techniques to increase the reliability and reduce the latency of data collection in predictable and uncontrollable MS systems. We first show how to model the expected upload latency of each cluster head. Then, we present a synchronized dynamic round-robin scheduling policy for uploading data to a mobile collector that is based on assigning a method for load balancing. We describe the implementation of our algorithms by using multiple channels for increased throughput and incorporating network coding for improved reliability. Our evaluation of data aggregation in the presence of packet errors shows that MuTrans can significantly reduce the latency for data collection, thus providing strong support for mobile data collection.**

I. Introduction

The term Internet of Things (IoTs) is recognized as a collection of networking technologies for embedded physical objects that interact and exchange data with other objects. Wireless Sensor Networks (WSNs) have a major contribution in exploiting the IoTs. From a broader perspective, WSNs form an important subclass of a rapidly emerging system class called Low-power and Lossy Networks (LLNs). LLNs are embedded systems with limited resources such as processing, power, and memory [20]. They are lossy because they use wireless communication that has a high error rate and unpredictable reception.

Due to these constraints, data collection in dense and highly distributed WSNs that require long multi-hop routing paths to data sinks can significantly increase management complexity, expend excessive amounts of power in relaying packets or, in the case of battery-powered WSNs, degrade network lifetime [4]. On the other hand, using mobile data gathering can save sensors battery life and diminish the number of relay-only nodes that need to be deployed. Mobile data collection is also suitable to retrieve data from networks deployed in isolated and hard to reach locations.

Despite the known advantages of mobile data collection for many types of LLNs, it remains a challenging problem to effectively support mobile data collection. This is due to the fact that LLN nodes generally suffer from high bit and packet error rates. To address this problem, a wide range of mobile data collection application classes has been broadly investigated [7].

In this paper, we focus on uncontrollable predictable mobile applications. This type of MS traverses the network in a predetermined path with an arrival schedule that is known in advance, without motion control that can be modified by our algorithms and protocols. An example of such an application: a public transportation vehicle (e.g. bus or train) in Smart Cities [19]. While the vehicle is in service, it collects data from surrounding nodes and delivers it to the appropriate base station (BS).

One critical issue in data collection in the presence of uncontrollable predictable MSs is to reduce the latency of data collection. This is because the MS is only in the range of each node for a fixed amount of time, so if some data is not collected it must wait until the next time the mobile comes in ranges. Our paper addresses this problem by designing and evaluating a novel multi-channel LLN protocol that also takes advantage of network coding for uncontrollable predictable mobile data collection. We call our protocol *MuTrans* for *Mu*lti-channel *Trans*port.

By transmitting on different frequencies, multi-channel LLN networking can both reduce delivery latency and improve system throughput [14]. Likewise, network coding [5] is a mathematical technique where nodes combine bits from multiple packets to increase network's reliability. It works by treating bit strings as elements in a Galois field and conducts finite field operations over different sets of bits. For wireless networks that have high bit error rates, network coding is known to significantly improve throughput [13].

We target inexpensive single transceiver nodes that are capable of selecting multiple transmission channels. Our sink is a powerful multi-radio mobile data collector. We assume a clustered system with two types of nodes: head nodes and member nodes. The head nodes are the nodes that have direct communication with the mobile sink, while member nodes are unreachable by the mobile sink but are one hop away from head nodes where multi-hop routing is not required. To our knowledge, this is the first work to explore using multi-channel data transmission and network coding to support an uncontrollable predictable mobile data collection in LLNs.

Since the mobile data sink moves in a known in advance arrival, data collection latency becomes substantial in regards to the network instability, due to the dynamic changes in the network environment.

The MuTrans protocol consists of two heuristic techniques: data load balance and weighted dynamic round robin scheduling. The first heuristic reduces the data overload at the head nodes, while the second provides a fairness uploading schedule that dynamically changes to reduce the overall uploading latency. MuTrans carefully assigns member nodes to the head nodes in a load balanced way for both saving energy and reducing the concurrent uploading latency.

Using a campus-based mobile data collection scenario, we evaluated the performance of MuTrans against several other approaches. Our results show that MuTrans outperforms these other methods in terms of packet delivery rates, network throughput, and the number of trips required by the mobile collector.

II. Background and Related Works

Our approach assumes the ability to form one hop clusters in a controllable predictable mobile data collection environment. Example work in this area includes Load Balanced Clustering with Dual Data Uploading (*LBC-DDU*) [22]. LBC-DDU constructs energy-balanced clusters while having a mobile data collector to aggregate data from cluster heads in a reliable communication using two antennas. LBC-DDU carefully picks what so called *polling points* from a set of candidate polling points. The polling points are the locations the mobile collector has to stop by and retrieve the data to send them later to the base station. LBC-DDU converts any form of routing into a one-hop path by creating balanced single-hop clusters.

There has also been significant research in both multi-channel networking [21], [3], [10] and network coding [16], [13], [17] within the LLN context.

In terms of supporting mobile data collection, one strategy is to use controlled mobility. Work in this area includes Kansal et al. [12], who demonstrated that using mobile data collection can increase network lifetime and utility in the embedded sensor nodes. A second strategy considered multiple mobile sinks as in Jea et al. [11] who extended [12] by employing multiple mobile collectors to reduce data latency. They divided a convex area into sub-areas in which a load balancing algorithm is implemented to assign each mobile collector (MULE) to evenly sub-area. Nevertheless, the scalability becomes an issue because increasing the size of the convex area and increasing the size of the sensor nodes degrades the performance with multiple sinks.

Gao et al. [6] designed efficient member assignments to the sub sinks using a Genetic Algorithm (GA) in order to enhance the power consumption and the data delivery to the mobile data collector, which traverses the network in fixed trajectory with multi-hop communication. Each sub-sink buffers the received data from member nodes using shortest path routing trees. On the other hand, Liang et al. [15] suggested an approximation algorithm for an NP-Hard optimization problem called Capacitated Minimum Forest (CMF) in hierarchical heterogeneous WSN architecture. The mobile sink traverses through planned trajectory and collects data from the gateways where they then temporarily aggregate data from sensors through multi-hop routing.

By considering the changes in mobile sink trajectories, Smeets et al. [19] implemented a mobile platform called *Trainsense* to support mobile WSNs applications. The modeled train inheres some of its features – such as controllers and detectors – to the motes for different mobility trajectories. Nevertheless, none of these related works have considered fixed route and uncontrollable mobile data collectors with one-hop routing in unstable environments.

By considering this mobility model, we propose two heuristic methods to reduce the uploading latency by applying the advantage of network coding and multi-channel coding in LLN networks [2]. We use the *MuCode* protocol [1], which allows head nodes the possibility to overhear packets synchronously from their member nodes, while also eavesdropping on packets from other nodes in order to increase the chances of recovering lost data. Each head node implements network coding to the eavesdropped packets along with the already stored packets, and that combination creates encoded packets. Then the head node stores them in the transmit queue, which consists of both encoded and plain packets. Finally, each head node concurrently sends all its data to the sink through different channels in order to avoid collision with one condition, that the mobile sink has enough multi-radios to receive packets simultaneously.

III. System Architecture

Our application class is supported by general purpose LLNs that use resource constrained wireless sensor nodes. The purpose of these nodes is to periodically sense and process data and then report this data back to a base station or base stations. In our case, the mobile collector is used to achieve this goal. The method for reporting this data back to the appropriate base station is via a mobile data collection node. The overarching performance objectives in this environment are to reduce the latency of data collection, to ensure that all available data is transferred to the mobile node when it is in contact.

As explained in Sections 1 and 2, a significant challenge in achieving the above objectives is to maximize throughput while coping with high bit error rates. To meet this challenge, we use both network coding and multi-channel LLN transmission policies.

Network coding in a lossy wireless system relies on the ability of nodes to overhear communications from neighbors. These eavesdropped packets can then be coded with other packets to improve network reliability by allowing the same information to be sent to MS via different paths. We have designed MuTrans to take advantage of eavesdropping transmission overhearing. This eavesdropping is performed by cluster head nodes. The receivers can decode packets intended for them by performing mathematical operations based upon information they already have. It is also used to allow receivers to recover a number of packets even when some of those packets are lost due to environment instability. From a mathematical perspective, network coding is performed by techniques such as Random Linear Network Coding (RLNC) [9], [8], where packets are defined by elements over a Galois field. The original packets are recovered by using operations such as the Gaussian elimination. The Random part of RLNC means the bits are combined after they are multiplied by a random coefficient.

In our architecture, we assume that the route of the mobile collector (MuCar) is known, and the head nodes are determined using the LBC-DDU algorithm, an algorithm that is described in Section 2. We now precisely define *polling sector* as the range in which the MuCar can communicate with head nodes and receive data while it is on the move. The polling sector length at head node H_i can be measured by the distance between the starting point, where MuCar contacts with H_i, and the ending point before it disconnects with H_i. The location of each polling sector is known once the head nodes are selected.

Members transmit their data to the nearby head nodes where the data is stored. Each member node picks one head node to send their data to. We are assuming a non-splitting data flow system, which means that the data from a node has to be sent to its dedicated neighbored node without dividing the data into multiple receivers. Once MS initiates a contact with the head nodes, they send their aggregated data along with their own data to MS.

LLN nodes possess radios that have single transceivers and are capable of supporting multi-channel communication protocols [18]. The sink, on the other hand, has multiple radios that allow it to receive data from head nodes concurrently through different channels. This multiple radio setup is used to prevent packet collisions and reduce uploading latency. By enabling multi-channel networking, head nodes in a neighborhood can send packets simultaneously without causing collisions, as long as they use different channels (λ_i). The gap between any two frequencies is large enough so that the interference is eliminated, meaning that channel λ_i does not interfere with λ_j, where $i \neq j$. Successful use of multi-channel communication requires that senders and receivers agree upon which channels to use at which time, and their agreement requires a channel scheduling policy. The advantage of this approach is that any pair of nodes that use different channels will not interfere with each other, thus avoiding the hidden terminal problem.

Transmitting over multiple channels decreases latency and increases throughput. In mobile systems, the head nodes must use different frequencies by a deliberate policy of synchronized channel switching that occurs while uploading data to the multi-radio mobile sink [2]. The channel switching is needed when the number of head nodes at the uploading position are larger than the number of available radios at the mobile sink.

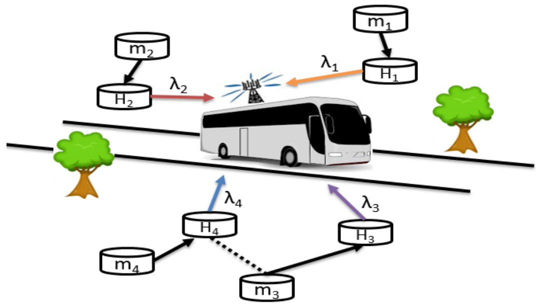

Fig. 1: MuTrans System Model

Figure 1 illustrates the MuTrans system architecture. The basic idea is that there are two types of nodes – head nodes and member nodes. The Figure shows the member node m_3 sending its packets to its head node (solid line) H_3 , while another head node H_4 overhears the packets (dotted line) and uses network coding to enhance network reliability.

Recall that the definition of a polling sector is the range within which the MuCar can communicate with head nodes and receive data while on the move. The polling sector length at head node H_i can be measured by the distance between the starting point where MuCar contacts with H_i and the ending point before it disconnects with H_i. Notice that the location of each polling sector is known once the head nodes are selected. Once the mobile sink (MS) enters the polling sector, the head nodes transmit their stored data through different assigned channels λ_i.

The member node assignment to each head node becomes critical in this situation. This assignment can increase the data load at the head node, which increases the energy consumption while also increasing the uploading latency to the MS. Since the MS has uncontrollable mobility, the uploading deadline is firm and unbreakable at each polling sector. Another important consideration is when the number of concurrent uploading head nodes exceeds the number of radios available at the MS. Therefore, fair and balanced scheduling mechanisms have to be designed to overcome this problem.

IV. System Model

Based off of the system architecture described in Section 3, we now show how to model the application network. We describe the latency analysis in the worst-case scenario where all the packets are successfully received to their designated head nodes. We first define the inter-session σ as the total time the MS travels the network while receiving the data from all of the head nodes and download them to the base station in a dwell position. Without loss of generality, we consider the MS to perform periodic data collections, so, for every σ time unit, the mobile makes a "round trip."

Let the distance PS_j represent the polling sector j and the distance the MS travels once it starts communicating with H_i until it terminates that communication. We define Δ_j as the contact duration time between MS and the head nodes at PSj. The upload time $\omega_j(\Delta_j)$ is the time MS needs to upload data from a polling point j at contact duration Δ_j. We assume the MS has $\Re$ radios, and those radios allow it to receive data simultaneously from most $\Re$ head nodes.

Let $\Psi_i(\sigma)$ represent the data that needs to be uploaded to the MS from head node H_i at inter-session σ. $\Psi_i(\sigma)$ can be formulated as:

$$\Psi_i(\sigma) = \gamma_i \sigma + \sum_{z=1}^{k_i} \gamma_{z,i} \sigma \tag{1}$$

where γ_i is the data rate generated by the head node H_i and $\gamma_{z,i}$ is the data rate generated by a member node $m_{z,i}$ that is assigned to the head node H_i, while k_i is the number of member nodes assigned to H_i.

By assuming the packet size is fixed, using Equation (1) , the number of packets that the head node H_i has at inter-session time σ can be found by the following equation:

$$G_i(\sigma) = \left\lceil \frac{\Psi_i(\sigma)}{packet\_size} \right\rceil \tag{2}$$

Subsequently, the uploading latency for head node H_i at inter-session σ is equal to multiplying $G_i(\sigma)$ by τ, where τ is the packet transmitting latency time.

Let Γ_j represent the number of head nodes that the MS can cover while receiving data at polling sector j. Again, without a loss of generality, and to simplify the notation, we assume that if H_i and $H_{i'}$ share the same polling sector PS_j, then they both have the same starting and ending contact time while uploading their data to the MS. Consequently, if the number of head nodes are more than the number of radios available at MS, then all these nodes cannot upload to the MS simultaneously, because there will be transmitting interference between some head nodes that use the same channel. Therefore, we introduce the concept of *uploading scheduling* where each number of at most $\Re$ head nodes are allowed to transmit concurrently to the MS, while the remaining are scheduled for next transmissions in the same way as the previous schedule. The number of different uploading scheduling that the MS needs at PS_j is formulated as:

$$\Pi_j = \left\lceil \frac{\Gamma_j}{\Re} \right\rceil \tag{3}$$

This equation shows that if $\Gamma_j \leq \Re$, then only one schedule is needed to transmit a packet per head node at PS_j. The challenge is to reduce the uploading latency. We observe that each schedule latency is bounded by the maximum number of packets a head node has per schedule. Hence, the upload time $\omega_j(\sigma)$ takes to the MS at polling sector j for inter-session time σ can be formulated as:

$$\omega_j(\Delta_j) = \sum_{c=1}^{\Pi_j} G_*^c(\sigma) \ . \ \tau \ \leq \Delta_j \tag{4}$$

where $G_*^c(\sigma)$ is the selected maximum number of packets that a head node has that belong to the schedule c in inter-session σ at PS_j using Equation (2). This uploading latency should not exceed the contact interval time Δ_j or else not all data will be received during that trip. Unlike traditional transmitting with a single transceiver at the MS, in our multi-channel scheme the uploading latency among the head nodes is only considered by the node that has the maximum total number of transmitted packet among the head nodes. The reason is that head nodes transmit concurrently to the MS using different channels that have been assigned by the MS prior the uploading process.

Assuming that the communication duration between the MS and all of the head nodes from all of the polling sectors have the same contact period Δ, the *total uploading latency* $\Omega(\sigma)$ at inter-session time σ can be formulated as:

$$\Omega(\sigma) = \sum_{j=1}^{\overline{N}} \omega_j(\Delta_j) \ \leq \overline{N} \ \Delta \tag{5}$$

where $\overline{N}$ is the total number of polling sectors on the network.

V. MuTrans Protocol

The purpose of the $MuTrans$ protocol is twofold: First, to dynamically assign each member node to a head node. Second, to, as fairly as possible (in the sense of overall reducing total upload latency), assign uploading schedules to head nodes. For each of these problems, we present polynomial time heuristics. The heuristics are meant to be run with full knowledge of system parameters, and therefore can be run by the MS. The protocol can be run continuously as the mobile traverses the system or can be run whenever system parameters change, such as head node or member node assignment, changes in data rates, or changes in mobility patterns as in LBC-DDU.

A. Data Load Balancing

The Data Load Balance (*DLB*) - Algorithm 1 - heuristically balances data loads among the head nodes in order to reduce the uploading duration in the mobile data collector. We define $C(m_i)$ as the set of candidate head nodes that the member node m_i can assign its data to. Those candidates are one-hop distanced to m_i. Initially, each member and head node calculates its collected data $\Psi_{m_i}(\sigma)$ and $\Psi_{H_i}(\sigma)$ respectively from Equation (1) during inter-session time σ.

We say that a schedule's balance is measured by the quantity $|G^x_{max} - G^x_{min}|$, and that a schedule that is "balanced" has the property $\min |G^x_{max} - G^x_{min}|$.

In steps 4-5, the set Q is defined as having all of the member nodes at the current polling sector so that it is sorted based on the number of candidate head nodes per member nodes in ascending order and then based on the local data in descending order. In this way, the least candidates' member node chooses first in order to give more precise and better options to the member nodes that have more candidate head nodes. If we do the opposite, then a member node with high candidates may choose a head node, which may be the only option for another member node – therefore increasing the load to that head node. This can be avoided if that choice was processed later.

Algorithm 1: Data Load Balance (DLB)

```
1:  Initialization: each member node $m_i$ defines
    $C(m_i) = \{H_i | \forall H_i \in NB(m_i)\}$;
2:  Initialization: each member node $m_i$ calculates
    $\Psi_{m_i}(\sigma) = \gamma_{m_i}.\sigma$;
3:  Initialization: each head node $H_i$ calculates
    $\Psi_{H_i}(\sigma) = \gamma_{H_i}.\sigma$;
4:  $Q = \{m_i | \forall i \in \{1, \ldots, \mu_j\}\}$;
5:  Sort $m_i$ in $Q$ based on the size of $C(m_i)$ in ascending
    order, then based on $\Psi_{m_i}(\sigma)$ in descending order;
6:  while $Q$ is not empty do
7:      $m \leftarrow EXTRACT(Q)$;
8:      $m$ chooses head node $H_i$ s.t. $H_i \in C(m)$ and $\Psi_{H_i}(\sigma)$
        is the lowest;
9:      $\Psi_{H_i}(\sigma) \leftarrow \Psi_{H_i}(\sigma) + \Psi_{m_i}(\sigma)$;
10: end while
11: Output: each member node is assigned to a head node
    with data load balanced;
```

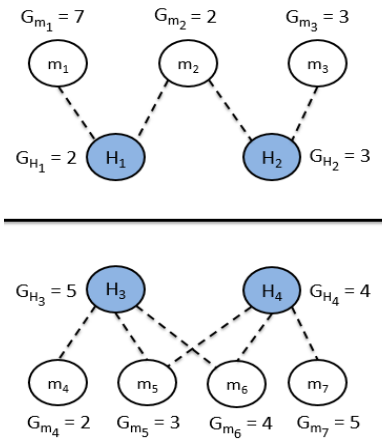

Fig. 2: An example of the candidate head nodes per member node. Each number represents the data load.

In steps 6-10, iteratively, we pick the first member node and choose the head node with lowest data load and update its data load by including the member node's data load. This iteration is processed until the set Q is empty.

Figure 2 represents an example of the *DLB* algorithm where the dark circles are the head nodes, while the white circles are the member nodes. The dotted line shows the candidate head node that will be assigned by the member node, while the solid line represents the path of the MS. Each node calculates its G value during σ duration using the Equation (2). Because head nodes are not assigned to any member node yet, the head nodes calculate their G values using only their local data.

In this example, the set Q is equal to $\{m_1, m_7, m_3, m_4, m_6, m_5, m_2\}$, which is sorted based on the member nodes with the least options to choose a head node and with the highest number of packets. With regarding this sequence, each member node selects the head node with smallest data load. Figure 3 shows the final assignment with load balanced among the head nodes.

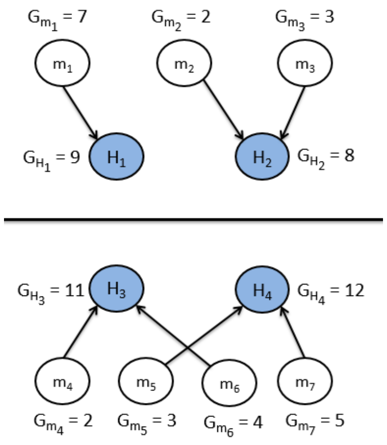

Fig. 3: The member assignments after using *DLB* method.

Algorithm 2: Dynamic Round-Robin Scheduling (DRRS)

```
1:  Input: head node set $HS_j$;
2:  $S \leftarrow HS_j$;
3:  while $S$ is not empty do
4:      Sort $S$ based on $G_i$ in descending order;
5:      Divide $S$ into $g = \left\lceil \frac{|S|}{\Re} \right\rceil$ schedules;
6:      Select the $g$ head nodes where each has minimum $G$
        per schedule $x$ ($G^x_{min}$);
7:      In each schedule $x$, head nodes concurrently transmit
        their data $G^x_{min}$ times;
8:      For each head node $H_i$ at schedule $x$,
        $G^x_i \leftarrow G^x_i - G^x_{min}$;
9:      Remove all head nodes with $G_i = 0$ from set $S$;
10: end while
11: Output: fairness dynamic schedules that utilize
    concurrent heads uploading;
```

The complexity time for our heuristic algorithm is $\mathcal{O}(\mu_j$ x $(\Gamma_j + \log \mu_j))$, where μ_j is the number of member nodes, while Γ_j is the number of head nodes.

B. Utilized Fair Scheduling

A synchronized schedule that is based on the number of packets in the head nodes has to be maintained to reduce the uploading latency. Given Γ_j, and given head nodes that want to upload their data to the mobile data collector at a given polling sector PS_j, we define Algorithm 2 , which fairly and dynamically utilizes the scheduling of the concurrent head nodes that are uploading. Let HS_j be the head node set at the polling sector PS_j.
Steps 1-2 takes the HS_j set and assigns it to set S. The iteration of the algorithm considers the remaining non-sent packets in the set S. At first, the set is sorted based on the number of packets each head node has in descending order. Then, it divides these head nodes by the number of available

channels that the mobile data collector has, which creates g schedules of head nodes. Each schedule x has a head node with a minimum number of packets (called G^x_{min}), which is the indicator for how many simultaneous transmitting should be done during this iteration. We use Weighted Round Robin scheduling, in the sense that $G^1_{min} \geq G^2_{min} \geq \ldots \geq G^g_{min}$, which leads to the number of concurrent transmitting per iteration in schedule 1, which is always greater than or equal to schedule 2, and so on until schedule g. After the transmitting process is complete in this iteration, each head node at schedule x deducts G^x_{min} from its total packet, then S removes all the head nodes that have no remaining packets to send. The algorithm repeats the previous steps until the head nodes transmit all packets.

Before we define fairness, certain terminology must be introduced. Let x be a schedule, and let $|x|$ be the number of concurrent heads uploading at schedule x. A schedule x, which has the "maxed concurrent heads uploading" when it satisfies the property: $\min |\Re - |x||$.The fairness uploading scheduling is weighted by the G_{min} on each schedule. For each iteration, the weighted round-robin scheduling is taking place with the queuing ratio $\frac{G^1_{min}}{G^g_{min}} : \frac{G^2_{min}}{G^g_{min}} : \ldots : 1$ where for instance, in the first iteration, schedule 1 transmits G^1_{min}/G^g_{min} packets while schedule g transmits only single packet in round-robin fashion. This process is repeated until the end of the first iteration.

Figure 4 depicts an example of *DRRS* scheduling with two radios available at MS. In the first iteration, the set S = $\{H_4, H_3, H_1, H_2\}$ where it is sorted based on the largest number of packets. Then, the set is divided into 2 schedules: schedule 1 = $\{H_4, H_3\}$ with $G^1_{min} = 11$ while schedule 2 = $\{H_1, H_2\}$ with $G^2_{min} = 8$. Now, in the transmitting phase, each head node on the same schedule transmits packets simultaneously using different assigned channels. However, each schedule has its own time slot that does not overlap with another schedule. The round robin scheduling is implemented to emphasize fairness among the schedules, where schedule 1 transmits 11 packets, while schedule 2 transmits eight packets during the first iteration.

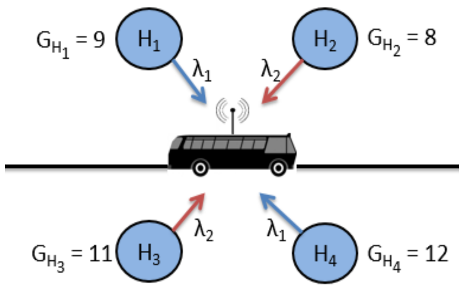

Fig. 4: *DRRS* example using multi-channel with $\Re = 2$.

In the second iteration, the set S will be $\{H_4, H_1\}$, which produces one schedule with $G^1_{min} = 1$, where each head node has only one remaining packet. The total time slots needed to transmit 40 packets from 4 head nodes is 20, which is exactly the same as the optimal solution, and that is not always the case. The running time complexity of *DRRS* algorithm is $\mathcal{O}(\Gamma_j^2 \log \Gamma_j)$.

VI. Evaluation and Results

Using Cooja, an LLN simulation tool, we evaluated $MuTrans$ against several other approaches. The purpose of this evaluation was to judge the effectiveness of our proposed heuristic. Since there do not exist other heuristic algorithms that are applicable in our predictable multi-channel environment, we defined two simple heuristics to solve our two subproblems: Random Member Assignment RMA and Static Round Robin Scheduling $SRRS$. RMA is an algorithm that randomly assigns member nodes to their nearby head nodes without considering data balancing, whereas $SRRS$ is a round robin scheduling that fixes the head node location to its original schedule.

These two baseline heuristics enabled us to define three new protocols. The first combination is RMA + $SRRS$, where the head node selection is randomly assigned by the member nodes, while the concurrent uploading schedules are statically assigned to the head nodes with a single iteration. The second combination is RMA + $DRRS$, where the head nodes are selected randomly by the member nodes, while the concurrent uploading schedules are dynamically assigned to the head nodes with multiple iterations as in algorithm 2. The last combination is DLB + $SRRS$. Here the head nodes are assigned by the member nodes based on algorithm 1, which balances the data rate load at the head nodes; however, the concurrent uploads are statically scheduled to these head nodes. All of these three new protocols use multi-channel network coding.

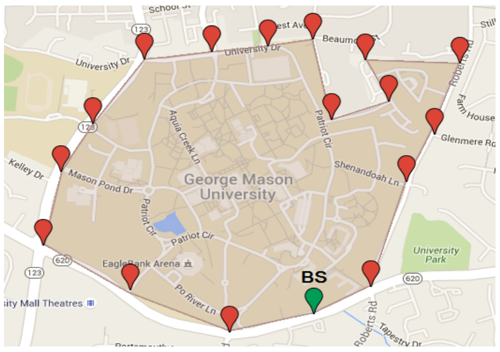

Fig. 5: George Mason University map

To represent a realistic uncontrollable but predictable environment, we used a topological representation of George Mason University, as shown in the Figure 5. We used the Contiki operating system to implement the protocols, where the red/light drop pins are the polling sectors in which head nodes upload their data to the MS, while the green/dark drop pin represent the Base Station in which MS has to dwell at and send the collected data to. Cooja, the Contiki simulation tool, emulates the network nodes and its hardware platform. We chose the TMote Sky node which transmits 250 kbit/s using MSP430 microprocessor and CC2420 radio. The platform works in 10 KB of random access memory and 48 KB of program flash. The radio medium is configured to the Unit Disk Graph Medium Distance Loss with 50m transmission and interference ranges. In our experiments, we assumed that each node sends a packet of size 50 bytes at time slot t where the difference between two consecutive time slots is 100 milliseconds per transmission.

The MS is traversing the GMU campus in a fixed route with a distance of 5.3 km and a speed of 55 km/h. We installed 16 polling sectors, where the number of head nodes is 6 per polling sector. The total number of deployed nodes are 192 with Δ = 4.6 seconds contact duration between the MS and the head nodes. Each member node i has its own data transmitting rate γ_i in order to test the data balance algorithm. These member nodes are randomly scattered and are 1-hopped away from the head nodes. We evaluated our protocols with a different number of radios $\Re$ available at the MS. These experiments are conducted with a variety of wireless Packet Error Rates (PERs) using the *MuCode* network coding protocol [2].

A. Packet Delivery Rate (PDR)

The first experiment is to judge the Packet Delivery Rate. Figure 6 depicts the average packet delivery rate versus PER using two radios in MS. In a reliable communication (i.e. PER = 0%), the PDRs are maximized, and all of the four techniques are equal. At PER = 10%, however, DLB + $DRRS$ becomes the only method that maintains full PDR, with a difference up to 8.7%. Even when PER gets higher, our protocol is superior to the other techniques by, at least, 40% , 3%, and 11% in PDR, which can be compared to RMA + $SRRS$, RMA + $DRRS$, and DLB + $SRRS$.

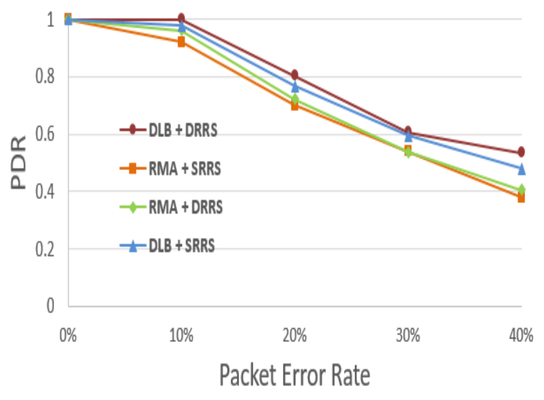

Fig. 6: Average packet delivery rates with $\Re$ = 2.

The reason for this result is that DLB increases the chance to get more overheard encoded packets from neighbors, which eventually leverages the use of network coding to decode the lost packets. Furthermore, $DRRS$ increases the total number of packets to be transmitted, compared to $SRRS$ for the same contact duration, which raises both the plain and encoded transmitted packets. It can be noticed that DLB significantly plays a greater role in packet delivery consistency when compared to $DRRS$.

B. Network Throughput

Figure 7 shows the average network throughput with different PERs. The four protocols shown with $\Re = 2$ are the same techniques from the previous results. At PER = 0%, the throughput in $DLB + DRRS$ outperforms the other method with $\Re = 2$ from 3.7% to 12%. The reason for this superiority is that $DRRS$ allows more packets to be transmitted; however, RMA creates unbalanced data loading among the head nodes, which affects the uploading latency.Meanwhile, DLB effectively uses this lemma in order to reduce the uploading latency, which eventually increases the network throughput with a fixed contact time.

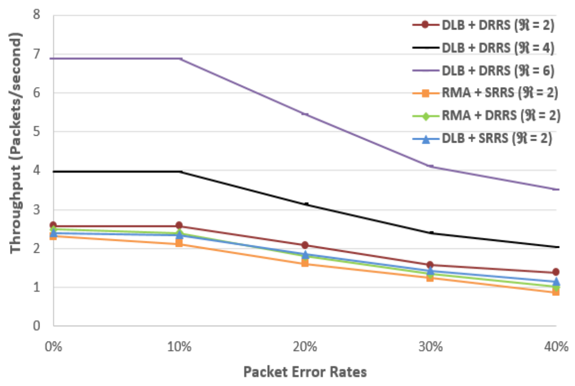

Fig. 7: Average network throughput versus PER with different $\Re$.

On the other hand, $DLB + DRRS$ $(\Re = 6)$ outperforms the same model when $\Re = 2$ and $\Re = 4$ in relation to network throughput in different packet error rates up to 166% and 73%, while at PER = 40%, the gap is shrunk slightly to 154% and 72%. This throughput's advantage is because $DRRS$ increases the concurrent data uploading by increasing the number of radios available in MS in order to be equal to the number of head nodes at that polling sector, which increases the total number of packets efficiently.

C. Trips Required for Data Delivery

One of our motivations is to quantify the number of trips required to complete data delivery. To assess this, we ignore the impact of buffer overflow and simply count the number of required trips. Figure 8 depicts the average number of trips that MS $(\Re = 2)$ has to accomplish in order to get the data needed by the system that has 40% of PER, with different total data packets originated by all the nodes on the network.

With a small amount of data, even with the high packet error rate, all of the four protocols can get all of the data that is originated by the nodes. There are two reasons for this successful transmission: First, network coding is implemented in all of them, which helps recover lost data packets. The

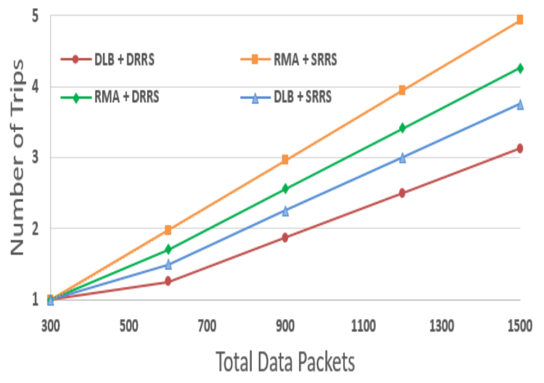

Fig. 8: The average number of trips MS has to take to collects all the data with PER = 40% and $\Re = 2$.

second reason for the completed transmission is that the contact duration meets the deadline for uploading the needed data packets without the need for retransmissions.

Once the data packets increase to 900, the MS has to take another trip in order to get the remaining lost packets that needed to be retransmitted. Since the contact duration time is always fixed in such an application, the over amount data packets, which are not considered by the system administrator, will miss the deadline. Hence not only are the retransmission of packets needed in high packet loss rates, but also the MS has to take more trips in order to receive all of the data packets, as seen in the previous figure. Nevertheless, DLB + $DRRS$ outperforms $RMA + SRRS$, $RMA + DRRS$, and $DLB + SRRS$ by up to 57%, 36%, and 20% in data upload latency respectively.

VII. Conclusion

This paper presented *MuTrans*, a network-coded multi-channel protocol for uncontrollable but predictable mobile data transport. Network coding has been used to significantly enhance the reliability and throughput for low power and lossy networks. Multi-channel networking, on the other hand, can essentially reduce the uploading latency and improve the network throughput. *MuTrans* balances the non-reachable node assignments to the local head nodes. It uses synchronized dynamic round robin scheduling for uploading data to the mobile data collector. We presented two heuristic algorithms Data Load Balance DLB and Dynamic Round-Robin Scheduling $DRRS$ and evaluate them against different heuristic techniques. The results indicated that *MuTrans* outperforms the other methods in packet delivery rates, throughput, and latency.

Acknowledgment

This work is supported by NSF under grants CNS-1116122 and CNS-1205453.

References

[1] M. Abdulaziz and R. Simon. Mobile data collection using multi-channel network coding in wireless sensor networks. In *Local Computer Networks (LCN), 2015 IEEE 40th Conference on*, pages 205–208. IEEE, 2015.

[2] M. Abdulaziz and R. Simon. Multi-channel network coding in tree-based wireless sensor networks. In *Computing, Networking and Communications (ICNC), 2015 International Conference on*, pages 924–930. IEEE, 2015.

[3] S. Chieochan and E. Hossain. Channel assignment for throughput optimization in multichannel multiradio wireless mesh networks using network coding. *Mobile Computing, IEEE Transactions on*, 12(1):118–135, Jan 2013.

[4] M. Di Francesco, S. K. Das, and G. Anastasi. Data collection in wireless sensor networks with mobile elements: A survey. *ACM Trans. Sen. Netw.*, 8(1):7:1–7:31, Aug. 2011.

[5] C. Fragouli, J.-Y. Le Boudec, and J. Widmer. Network coding: an instant primer. *ACM SIGCOMM Computer Communication Review*, 36(1):63–68, 2006.

[6] S. Gao, H. Zhang, and S. K. Das. Efficient data collection in wireless sensor networks with path-constrained mobile sinks. *Mobile Computing, IEEE Transactions on*, 10(4):592–608, 2011.

[7] Y. Gu, F. Ren, Y. Ji, and J. Li. The evolution of sink mobility management in wireless sensor networks: A survey. 2015.

[8] T. Ho, R. Koetter, M. Medard, D. R. Karger, and M. Effros. The benefits of coding over routing in a randomized setting. 2003.

[9] T. Ho, M. Médard, R. Koetter, D. R. Karger, M. Effros, J. Shi, and B. Leong. A random linear network coding approach to multicast. *Information Theory, IEEE Transactions on*, 52(10):4413–4430, 2006.

[10] O. D. Incel, L. van Hoesel, P. Jansen, and P. Havinga. Mc-lmac: A multi-channel mac protocol for wireless sensor networks. *Ad Hoc Networks*, 9(1):73–94, 2011.

[11] D. Jea, A. Somasundara, and M. Srivastava. Multiple controlled mobile elements (data mules) for data collection in sensor networks. In *Distributed computing in sensor systems*, pages 244–257. Springer, 2005.

[12] A. Kansal, A. A. Somasundara, D. D. Jea, M. B. Srivastava, and D. Estrin. Intelligent fluid infrastructure for embedded networks. In *Proceedings of the 2nd international conference on Mobile systems, applications, and services*, pages 111–124. ACM, 2004.

[13] L. Keller, E. Atsan, K. Argyraki, and C. Fragouli. Sensecode: Network coding for reliable sensor networks. *ACM Trans. Sen. Netw.*, 9(2):25:1–25:20, Apr. 2013.

[14] Y. Kim, H. Shin, and H. Cha. Y-mac: An energy-efficient multi-channel mac protocol for dense wireless sensor networks. In *Proceedings of the 7th international conference on Information processing in sensor networks*, pages 53–63. IEEE Computer Society, 2008.

[15] W. Liang, P. Schweitzer, and Z. Xu. Approximation algorithms for capacitated minimum forest problems in wireless sensor networks with a mobile sink. *Computers, IEEE Transactions on*, 62(10):1932–1944, 2013.

[16] P. Ostovari, J. Wu, and A. Khreishah. Network coding techniques for wireless and sensor networks. In *The Art of Wireless Sensor Networks*, pages 129–162. Springer, 2014.

[17] R. R. Rout and S. K. Ghosh. Enhancement of lifetime using duty cycle and network coding in wireless sensor networks. *Wireless Communications, IEEE Transactions on*, 12(2):656–667, 2013.

[18] M. Sha, G. Hackmann, and C. Lu. Real-world empirical studies on multi-channel reliability and spectrum usage for home-area sensor networks. *Network and Service Management, IEEE Transactions on*, 10(1):56–69, 2013.

[19] H. Smeets, C.-Y. Shih, M. Zuniga, T. Hagemeier, and P. J. Marrón. Trainsense: a novel infrastructure to support mobility in wireless sensor networks. In *Wireless Sensor Networks*, pages 18–33. Springer, 2013.

[20] T. Watteyne, A. Molinaro, M. G. Richichi, and M. Dohler. From manet to ietf roll standardization: A paradigm shift in wsn routing protocols. *Communications Surveys & Tutorials, IEEE*, 13(4):688–707, 2011.

[21] Y. Wu, J. Stankovic, T. He, and S. Lin. Realistic and efficient multi-channel communications in wireless sensor networks. In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, April 2008.

[22] M. Zhao, Y. Yang, and C. Wang. Mobile data gathering with load balanced clustering and dual data uploading in wireless sensor networks. *IEEE Transactions on Mobile Computing*, 14(4):770–785, 2014.