2017

Analysis of multi-resolution data aggregation using push-assisted random walks in mobile ad-hoc network (MANET)

Sowmya Srinivasapura Devaraja

Analysis of Multi-Resolution Data Aggregation using Push-assisted Random Walks in Mobile Ad-hoc Network(MANET)

Sowmya Srinivasapura Devaraja

Problem Report submitted to the

Benjamin M. Statler College of Engineering and Mineral Resources

at West Virginia University

in partial fulfillment of the requirements for the degree of

Masters of Science

in

Electrical Engineering

Vinod Kulathumani, Ph.D, Chair

Brian Woerner, Ph.D

Matthew Valenti, Ph.D

Lane Department of Computer Science and Electrical Engineering

Morgantown, West Virginia

2017

Keywords: data aggregation, EZ-AG, self-repelling, random walk

ABSTRACT

Analysis of Multi-Resolution Data Aggregation using Push-assisted Random Walks in Mobile Ad-hoc Networks (MANETs)

Sowmya Srinivasapura Devaraja

Data Aggregation in Mobile Ad-hoc Network (MANET) has proven challenging because of changing topology. Structure-based models like tree-based, cluster-based and chain-based have high maintenance cost. In earlier works, different forms of biased random walks have been verified to be effective without need for structure maintenance. The key idea in the protocol was to use one or more tokens that are circulated using biased random walks to effectively compute the data aggregation. One such protocol is EZ-AG that uses "Push-assisted Self-Repelling Random Walks".

A self-repelling random walk of a token on a graph is one in which at each step, the token moves to a neighbor that has been visited least often. While self-repelling random walks visit all nodes in the network much faster than plain random walks, they tend to slow down when most of the nodes are already visited. It's verified that a single step push phase at each node can significantly speed up the aggregation and eliminate the slow down. Results have been verified that EZ-AG achieves aggregation in only O (N) time and messages.

When the network is quite large, obtaining only one aggregate may not be sufficient. It will be more useful to provide distance-sensitive multi-resolution aggregates of data. The contribution in this project is, we have analyzed the "Hierarchical EZ-AG" proposed to provide multi-resolution results. We show that aggregates for nearby regions are obtained at faster rate in comparison to the farther region. The idea is to introduce the tokens in the network at distinct levels, execute EZ-AG protocol and obtain localized data aggregation output at distinct levels. Existing techniques for hierarchical aggregations require O (N $\log^{5.4}$ (N)) messages. Hierarchical EZ-AG outperforms these techniques by aggregating with only O (N log (N)) messages. We evaluate the performance of hierarchical EZ-AG considering message overhead, token messages, number of aggregations at distinct levels, node speed and mobility. Our results are validated using simulations in network simulator, ns-3 for network ranging from 100 to 4000 nodes under different node speeds and mobility models.

Acknowledgements

First and foremost, I would want to thank my committee chair and advisor Dr. Vinod K. Kulathumani for his constant support, providing me an opportunity to work with him and guiding through the research to extend his work on EZ-AG protocol without whose support this day would not be possible. I would also like to extend my warm acknowledgements to Dr.Brian Woerner and Dr. Matthew Valenti for being part of my committee.

I would like to extend my sincere thanks to Dr. Masahiro Nakagawa for providing initial understanding on the system and protocol he formulated along with Dr. Vinod K Kulathumani.

I would like to thank all my friends from the bottom of my heart for their suggestions and helping me overcome all my tough times here in Morgantown.

Last but certainly not least, I would like to thank my parents – Mr. S N Devaraja, Mrs. K N Susheela and sister – Mrs. Kavya S D. Without their unconditional love, encouragement and selfless sacrifice this day would not be possible. My sincere thanks to Dr. Mudappa Rangappa who has been besides me throughout my endeavors.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

Mobile Ad-hoc Network (MANET) is a collection of two or more devices or nodes that can dynamically form a network. Nodes are low power devices, equipped with a radio and transmitter. They communicate wirelessly with its peers and/or base station using multi-hop traversals without the aid of any centralized administration. Dense and large sensor networks or MANETs are likely to exchange a lot of redundant information with its neighbors or the base station. Also, nodes utilize a large amount of power in transmitting data. This increases the message overhead, transmission power and decreases lifetime of the nodes in the network. Data aggregation methods are useful to subside these impacts and obtain reliable data. Data aggregation is a technique of aggregating the data to eliminate redundant transmissions and provide consolidated data.

Data aggregation can be broadly classified as: duplicate sensitive and duplicate insensitive data aggregation. Duplicate sensitive data aggregation (includes function like SUM and COUNT) are those whose outputs are altered when same input is incorporated more than once. Duplicate insensitive data aggregation (includes functions like MIN and MAX) are those whose output is not dependent on the repetitive inputs.

For instance, consider {p1, p2} = {5, 4}

SUM (p1, p2) = 9 is not equal to SUM (p1, p1, p2) = 14; (Duplicate sensitive)

MAX (p1, p2) = 5 is equal to MAX (p1, p2, p2) = 5; (Duplicate insensitive)

For static sensor networks, the data aggregation protocols have been designed to obtain the data routing by forming a structure/backbone. The data is routed along the established path to obtain data aggregates. The various structure-based models are tree-based, cluster-based and

chain-based. However, if the underlying structure is changing constantly, these protocols will have scalability issues and high maintenance cost. Applying these models to MANETs is not a reliable option as the topology changes frequently. The link changes per node per second over different speed is indicated in table 1.1. It's clear from the table that in the network, speed plays a vital role on how often the link between nodes break.

Table 1.1 Mapping between speed and link change per node per second

Size/Speed	3 m/s	9 m/s	15 m/s	21 m/s
100	1	5	7	9
300	2	7	10	14
500	3	8	12	16
1000	3	9	14	18
2000	3.8	10	16	20
4000	4	12	18	23

Many approaches have been proposed to obtain data aggregate without creating the structure. The various techniques are flooding, gossiping and using random walks. In flooding technique, node transmits the message/packet to its neighbors which is in turn forwarded/broadcasted to its neighbors either till the destination is reached or maximum hops are reached. This is called as blind flooding. If the number of neighbor increases, it directly affects the probability of collisions which in turn affects the performance and suffers from broadcast-storm problem. Alongside, because of the broadcast messages in this method, the message overhead cost would be very high as $O(N^2)$.

In gossiping technique, the receiving node will send the packet to a randomly picked neighbor, which further picks another random neighbor and the process continues till the entire network's data is aggregated. The general types of gossip techniques are neighborhood gossips and spatial gossips. Unlike neighborhood gossiping, spatial gossips can choose any other node in the network and gossip the data. This scheme uses O(N*polylog(N)) messages to finish the aggregation.

In the earlier work, the different forms of biased random walks has proven to be effective amongst the other structure-free techniques. The key idea in these protocols is to use one or more tokens that are circulated using biased random walks to effectively compute the data aggregations. One such protocol is EZ-AG that uses “Push-assisted Self-Repelling Random Walks”. This technique uses only O (N) messages and time to complete the data aggregation.

A self-repelling random walk of a token on a graph is one in which at each step, the token is handed to a neighbor node that has been visited least often. Figure 1.1 shows the self-repelling random walk. The light circle indicates the unvisited node and dark circle indicates the visited node and the number of visits has been highlighted within the circle. The node carrying Token-1, is surrounded by nodes visited 0, 1 and 2 number of times. While transferring the token to continue random walk, we see that priority will be given to neighbor node visited least number of times.

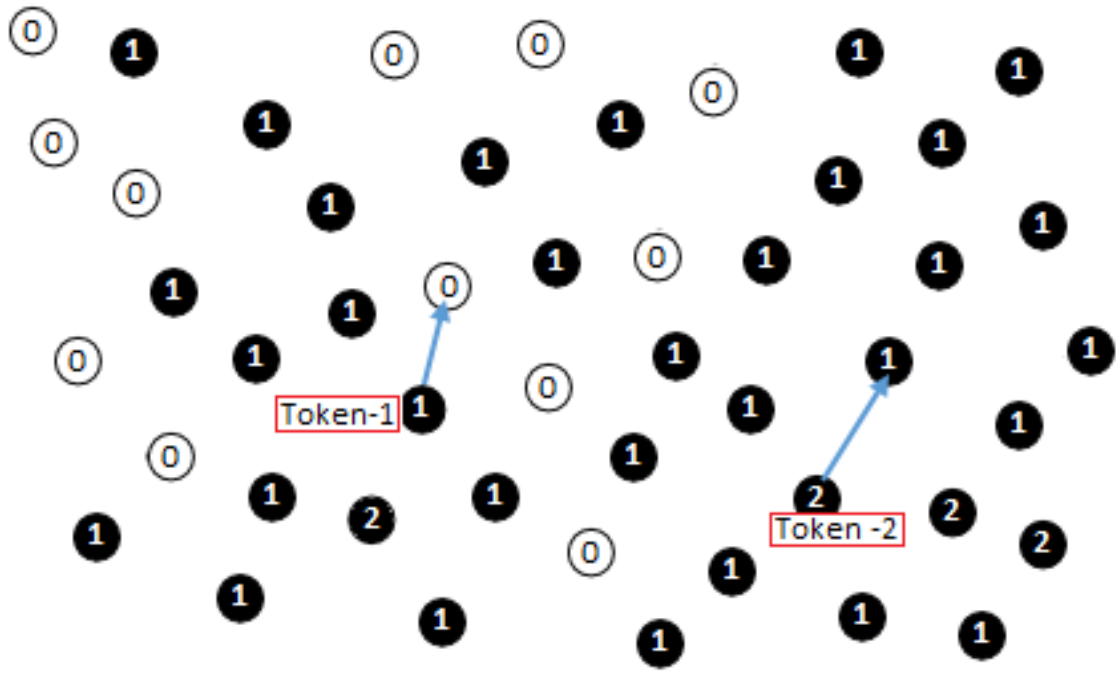

Figure 1.1 Token Transfers in Self Repelling Random Walk

Self-repelling random walks visit all nodes in the network much faster than plain random walks, but they tend to slow down when most of the nodes are already visited because the token

gets trapped between visited nodes. In this condition, token takes multiple hops to visit an unvisited neighbor. It's verified in [1] that a single-step push phase at each node before starting the self-repelling random walk can significantly speed up the aggregation and eliminate the slow down. Results have been verified that EZ-AG achieves aggregation in only O (N) time and messages. When the network is quite large, obtaining only one aggregate may not be sufficient. It will be more useful to provide distance-sensitive multi-resolution aggregates of data i.e., data aggregates for nearby regions are obtained at faster rate in comparison to the farther region. The extension to push-assisted self-repelling EZ-AG, "Hierarchical EZ-AG" has been proposed in [1].

The objective of this work is to implement and analyze the performance of push-assisted self-repelling hierarchical EZ-AG to obtain multi-resolution distance-sensitive data aggregation for duplicate-insensitive data.

1.2 Summary of Contributions

In this work, in-depth analysis has been done on the concept of hierarchical data aggregation. The idea of hierarchical EZ-AG is to introduce the tokens in the network at distinct levels, execute EZ-AG protocol and obtain localized data aggregation output at distinct levels. The result is nearer neighbors get the aggregate information more often than the farther neighbors.

Existing techniques for hierarchical aggregations require O (N $\log^{5.4}$ (N)) messages. Hierarchical EZ-AG outperforms these techniques by aggregating with only O (N log (N)) messages. Thus, we see poly-logarithmic factor improvement in terms of message overhead.

- We implemented and verified the hierarchical EZ-AG protocol which provides multi-resolution aggregates to each node for duplicate in-sensitive data using only O (N log (N)) messages.
- We analyze the performance of hierarchical EZ-AG using network simulator, ns3 for networks ranging from 100 to 4000 nodes under different node speeds and mobility models.
- We compare hierarchical EZ-AG message overhead with the spatial gossip technique which provides multi-resolution outputs.

1.3 Outline

The rest of the report is organized as follows. We discuss the prior work on structure-free data aggregation protocol, EZ-AG in chapter 2. We discuss hierarchical EZ-AG protocol in-depth and the implementation details in chapter 3. In chapter 4, we provide details regarding the system model (network model, different mobility models and the evaluation metrics), analyze the hierarchical EZ-AG protocol by presenting the simulation results obtained by ns-3 simulations, and compare with the existing hierarchical protocol results. After analyzing the results, we provide the conclusion in chapter 5.

Chapter 2

Review of Prior Work

In this chapter, we discuss the prior work done by applying random walk process to design an effective structure-free protocol to obtain data aggregates for MANETs.

2.1 Random Walks

Random walk is a process which operates without building an underlying structure. It can be exploited to be useful in data aggregation. The idea of random walks used in data aggregation can be summarized as follows:

1. One or more tokens are created at randomly selected nodes termed as holder nodes. Number of tokens vary based on the requirement.
2. Tokens are circulated to aggregate the data in the network by traversing through the nodes. Random walk strategy is employed to pass the token to the neighbors in the network.
3. When a node is visited for the first time, its data will be aggregated into the token. Therefore, when all the nodes in the network have been visited the data aggregation will be terminated.
4. To ensure there is no duplicate information, nodes keep track of number of times the token has visited.

Random walks work efficiently for changing topology demanding very less information to be saved thus minimizing the memory overhead. However, the simple random walks have longer cover time as the token visits the same node multiple times before it covers all nodes in the network. To overcome the limitations, biased random walks have been employed.

2.2 Biased Random Walks

In order to avoid the slowdown in simple random walks, biasing strategies where employed. One such idea introduced was self-repelling random walk. A self-repelling random

walk of a token on a graph is one in which the token is passed to the nodes which is visited least number of times by the token. This helps in prioritizing the token movement towards the unvisited nodes. While self-repelling random walks visit all nodes in the network much faster than plain random walks, they tend to slow down when most of the nodes are already visited i.e., it takes multiple token transfers to visit the node in the unvisited region. This is shown in figure 2.1. The dark circles and light circle represent the visited nodes and unvisited nodes respectively. The blue oval indicates the island of unvisited nodes. When the token is trapped in the visited region, we show that it takes 3 token transfers to reach unvisited node. This varies with the number of visited and unvisited nodes in the network and where the token is trapped. To address the slow-down of self-repelling random walk, 2 ideas where proposed and verified.

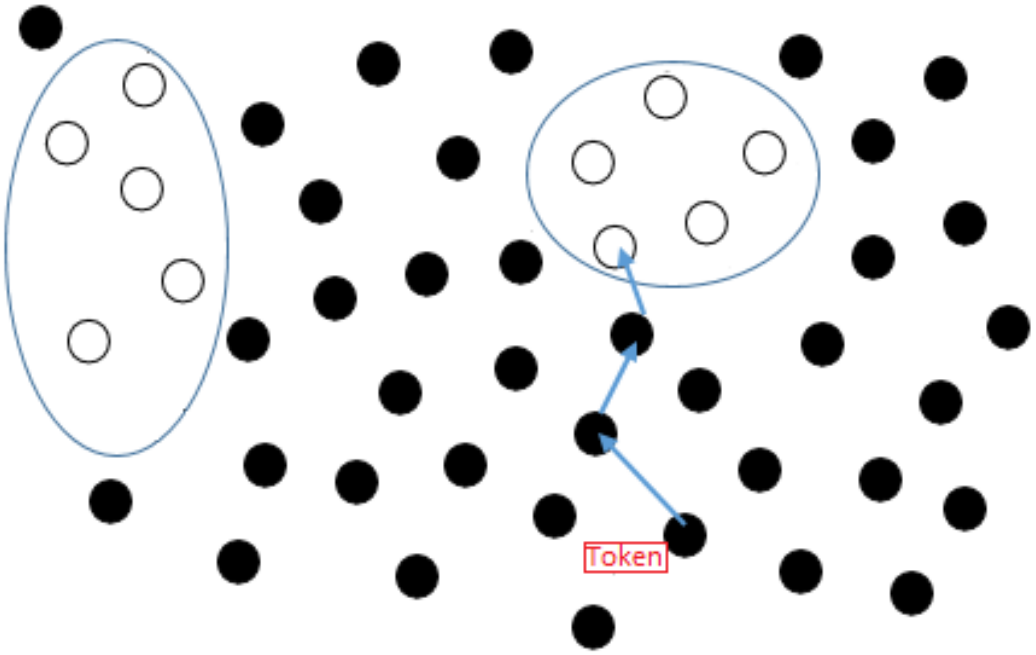

Figure 2.1 Self-repelling Random Walk showing multiple token transfers before visiting unvisited node

2.2.1 Gradient Biased Random Walk

In this method, short multi-hop temporary gradients are introduced from the edge of the visited region to pull the token towards the unvisited nodes. The first step is to establish a path from unvisited nodes to the token holder. The token follows the shortest gradient path to reach the unvisited node.

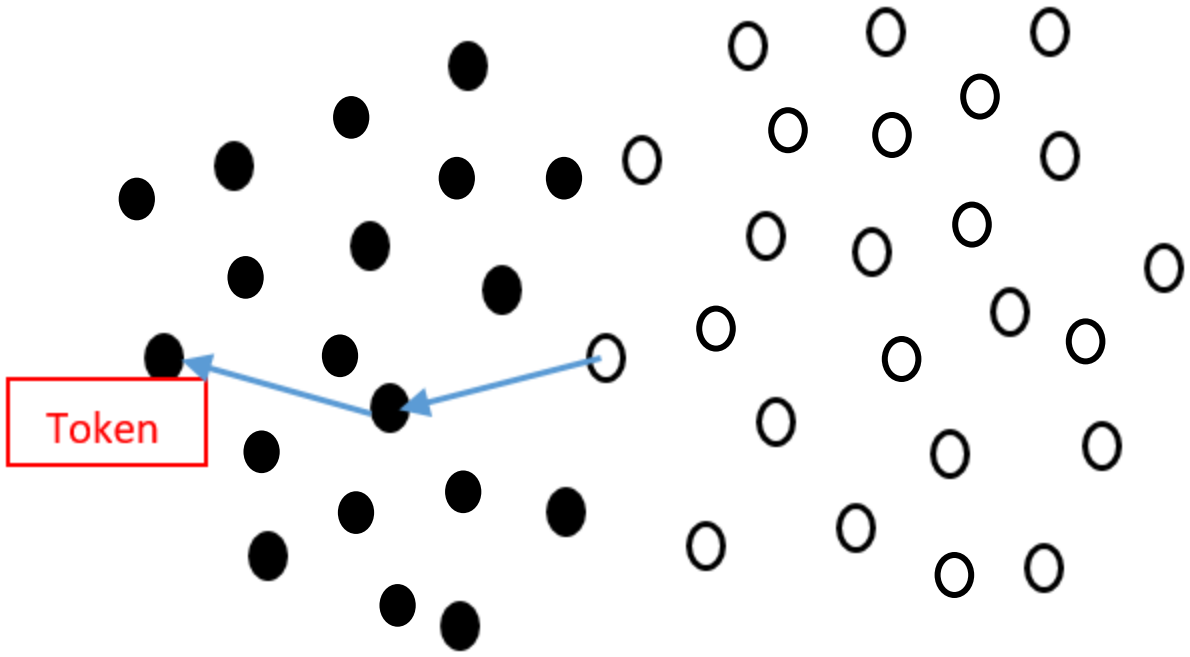

Figure 2.2 Gradient pull towards the token

Figure 2.2 shows the multi-hop gradient message towards the token node. The gradients will be set up periodically to attract the token towards the unvisited nodes. This method introduces a message overhead of O(N log(N)). However, it's verified that the data aggregation will be achieved in O(N) messages and time, the message overhead is compensated by the reduced number of token transfers.

2.2.2 Push-Assisted Random Walk (EZ-AG)

In this method, one-step PUSH phase was introduced before starting self-repelling random walk where information about the one-hop neighbors is learnt. Thus, all nodes carry its neighbor information throughout. Thus, the data aggregation is completed before visiting all the nodes in the network. Summarizing the complete idea of push-assisted, self-repelling random walks (EZ-AG) for a network of 'N' nodes:

1. Node that needs aggregate value of the network floods the interest (aggregate request) in obtaining aggregation to all nodes in the network. This results in N messages. Nodes on receiving the interest message will pushes its data to neighbor nodes.

2. PUSH phase: All nodes in the network will broadcast its state/data to one-hop neighbors (within communication range, R). By the end of PUSH phase, all nodes will have its data along with its neighbor data. This step will also require N messages.
3. The node (initiator/holder node) that sent out aggregation request will generate a token to start the self-repelling random walk. It adds its data into the TOKEN and will broadcast an ANNOUNCE message to other nodes in its communication range to pass the token and continue self-repelling random walk.
4. The neighbor nodes that receives ANNOUNCE message will respond with REQUEST message including number of visits by the TOKEN. Priority is given to the nodes that have lesser TOKEN visits to respond to ANNOUNCE message.
5. Holder nodes receive the REQUEST messages, transfers the token to node with least visits. (ties are broken at random)
6. The TOKEN is transferred to selected node via TRANSFER message. The node that receives the token is now the holder node.

The steps 3, 4, 5 and 6 are repeated till the aggregation of the network is complete. The aggregated value will be broadcasted to all the nodes in the network. It is also shown that the number of token transfers needed to achieve data aggregation reduces with the increase in mobility of the nodes.

The results in [1] show that before self-repelling random walk slows down the aggregate is computed in only O(N) time and messages. This implies that, in terms of message overhead, the EZ-AG protocol outperforms the other existing structure-free data aggregation protocols by a factor of log (N). Table 2.1 shows the comparison of message overhead for data aggregation in structure free models.

Table 2.1 Comparison of message overhead for data aggregation (structure-free models)

Structure-free Data Aggregation Method	**Message Cost**
Flooding	$O(N^2)$
Standard Gossip	$O(N^2)$
Spatial Gossip	O(N polylog(N))
EZ-AG	O(N)

2.2.3 Hierarchical EZ-AG

When a network is large, it is not sufficient to only provide a single aggregate value for entire network. It is more beneficial to provide distance-sensitive, multi-resolution aggregate values for the nodes i.e., a node receives the aggregate information of the closer neighbors more often compared to that of distant neighbors. The idea was proposed and analyzed theoretically in [1] to obtain multi-resolution aggregates in the network in O (N log (N)) messages.

Chapter 3

Hierarchical Aggregation

3.1 Protocol Description

The network area is divided into square cells at distinct levels (0, 1, … , M). The number of cells increase exponentially at each level (shown in Figure 3.1). At the lowest level (level 0), each cell is of side not less than $\frac{R}{\sqrt{2}}$, where R is the communication range for the node, and the diagonal of the cell represents R. Each level-0 cell has θ (log (N)) nodes with high probability. Such graphs are called as geo-dense geometric graphs. Four adjacent cells of level-0 constitute one level-1 cell and with high probability level-1 cell will have (4* θ (log (N))) nodes. Generalizing, four level-i cells combine to form one level-i+1 cell and number of nodes will be 4 times number of nodes in level-i cell. At the highest-level M, we will have only 1 cell covering the entire network. At any instant of time, a node will be part of one cell at each level.

Figure 3.1 shows the hierarchical EZ-AG network sketch. We observe that the cell increases exponentially from highest level (Level-3) to lowest level (Level-0). Node 'X' would receive aggregate from cell 'A' (level-0), cell 'B' (level-1), cell 'C' (level-2) and cell 'D' (level-3). It receives cell A (level-0) aggregates at faster rate in comparison with cell D (level-3) aggregates.

To obtain multi-resolution results, we introduce token at every cell at every level to execute the EZ-AG instance. At any instant, the token of a cell is transferred only to nodes within its cell boundary. Once the aggregation in the cell is complete, it floods the result to the nodes in its cell and adjacent cells. The computation and dissemination of data aggregates at different cells by EZ-AG instance are not time synchronized. Thus, we obtain the aggregates for cells at distinct levels at different time instants. Since the nodes are mobile, the aggregate it receives at any time instant belongs to the cell it is present at that time instant.

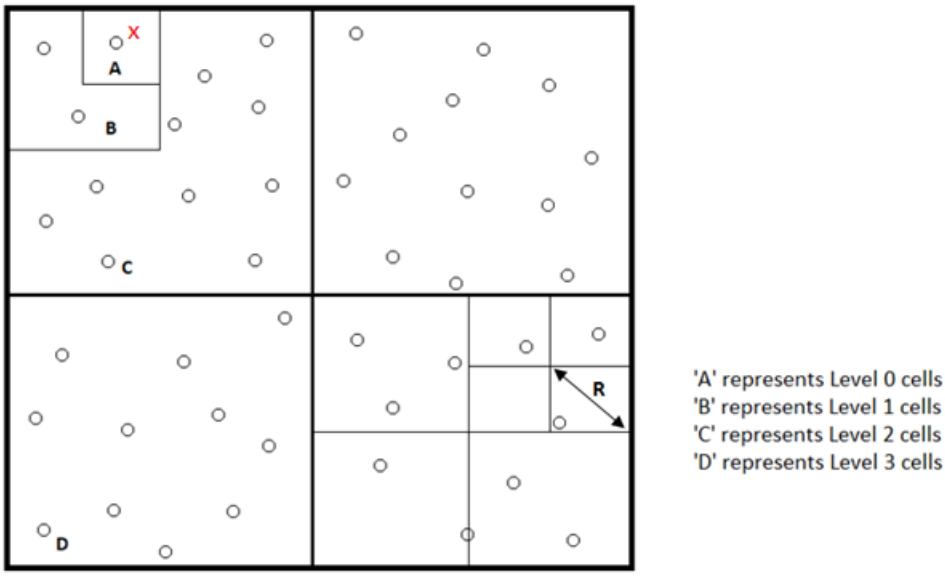

Figure 3.1 Hierarchical EZ-AG for multi-resolution data aggregation

According to the theorems stated in [1], "For Hierarchical EZ-AG, at any level j (highest level indicated as j = 0), the Order and Duplicate Insensitive (ODI) data aggregate can be obtained in $O(4^j\delta)$ time and messages, where δ represents the nodes with high probability in the cell". This implies that the results at lowest level are obtained at exponentially faster and aggregates are disseminated to the nodes in its cell and adjacent neighbors. "Hierarchical EZ-AG can compute an Order and Duplicate Insensitive (ODI) aggregate for all cells at all levels using O(N log(N)) messages". The simulation results obtained as part of this project, supports the theorems proposed.

Mathematical calculation on number of EZ-AG instances

We know that at any level, the number of EZ-AG instances is same as the number of cells. Consider a case as in figure 3.1, where we have 4 levels (0,1,2,3). Level-0 indicates the cells at lowest level and level-3 indicates cells at highest level. Hence, we have 64 instances of EZ-AG at level-0, 16 instances of EZ-AG at level-1, 4 instances of EZ-AG at level-2 and 1 instances of EZ-AG at level-3. Mathematically, we have (64+16+4+1) = 85 instances of EZ-AG running in the network to obtain multi-resolution data aggregates.

3.2 Implementation

Consider 'N' nodes randomly deployed in the network. The steps involved in obtaining multi-resolution data aggregates are as follows:

1. Network Division.
2. Run EZ-AG instances on different cells at distinct levels.
3. Disseminate the result to nodes in adjacent cells.
4. Terminate the EZ-AG execution.

Network Division

The total size of the network is determined by two factors, number of nodes being deployed in the network and the density of the nodes.

$$Size\ of\ network\ \alpha\ \frac{Number\ of\ Nodes}{Density\ of\ Nodes}$$

The network size increases as the number of nodes being deployed increases and decreases as we increase the density of the nodes in the area. We divided the network in hierarchical fashion such that levels in the network increase exponentially. The cells at distinct levels is numbered uniquely and is denoted by *cellID*.

Run EZ-AG instances on different cells

1. Node interested in aggregate information floods the interest message to all nodes in the network. Before the actual data aggregation starts, we have a one-step PUSH phase (to account for the slow-down of the self-repelling random walks) where the nodes push their information to all its current neighbors. EZ-AG instances start the data aggregation using tokens generated at random in all the cells for distinct levels.
2. Node that holds the token is coined as *starter or holder.* It adds its information in the token and continues the random walk by passing token to one of its neighbors.
3. The token is passed only if the following criteria is met
    a. The node is least visited amongst all the nodes requesting the token.

b. The requesting node must be in the same cell as the sender.
(The ties are broken at random)

4. Step 2 and 3 are repeated till the aggregation at a cell is complete, which is intermittently being verified. Once the aggregate is complete for the cell, the results will be disseminated to the neighbors.
5. Steps 2, 3 and 4 are repeated at all the levels till the entire network's data is aggregated at the highest level.

Disseminate the result to adjacent neighbors

The aggregated results at are disseminated (broadcasted) to its 8 adjacent cells at corresponding levels. Figure 3.2 (a) and Figure 3.2 (b) shows the cells to which the aggregate value will be disseminated at lowest two levels. In Figure 3.2 (a) we indicate the broadcast message from a node in cell X will be sent to all the nodes in cell X and all nodes in 8 adjacent cells of cell Y at level 0.

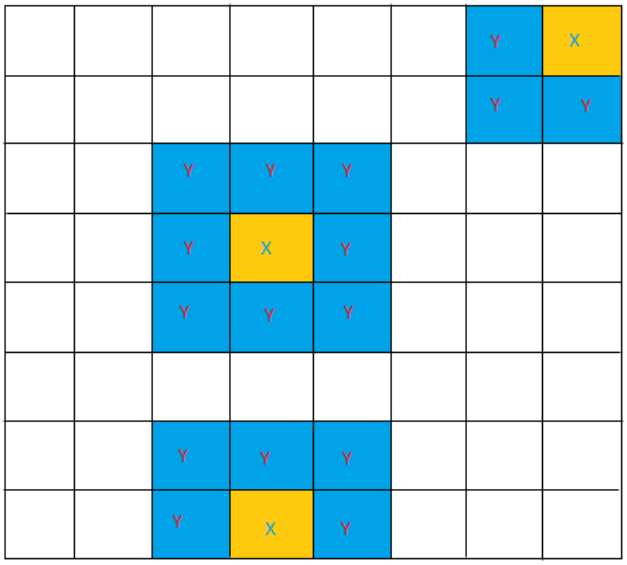

Figure 3.2 (a) Data Aggregation Result Broadcast at lowest level (Level 0)

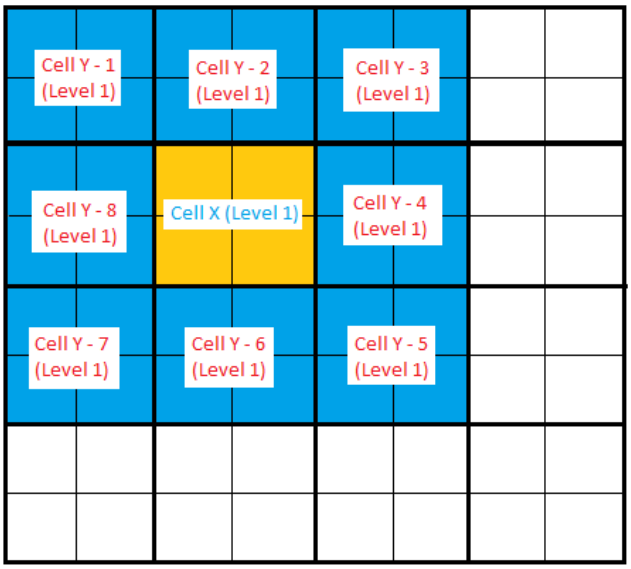

Figure 3.2 (b) Data Aggregation Result Broadcast at Level 1

Terminate the EZ-AG Execution

We terminate the EZ-AG execution when all the data has been aggregated once at the highest level (i.e., the entire network). There are no definite criteria to detect termination. However, as the number of nodes visited by the token increase, the ratio of token transfers to the visited nodes also increase. This ratio can be designed efficiently to detect token termination. Once the termination condition is met, the result at the highest level will be broadcasted to the entire network.

3.3 Aggregation Results at Every Level

Table 3.1 gives information about the number of aggregates expected at distinct levels for the network in figure 3.1. The analytical calculations are explained below.

Table 3.1 Analytical Results of Data Aggregation at distinct levels

Levels	Number of Aggregates
3	1
2	16
1	256
0	4096

Consider the figure 3.1, where the network is divided into four levels (0-3). 'A' represents the smallest level cells and 'D' represents the largest level cells (entire network). We can deduce from the division that we have 1 level-3 cell, 4 level-2 cells, 16 level-1 cells and 64 level-0 cells. As and when the aggregation is completed at each level the results will be disseminated to its 8 neighboring cells as indicated in section 3.2.

Number of disseminations expected at Distinct Levels

In the network, the number of tokens is directly related to the number of EZ-AG instances in that level. At level-3, we have only one cell which implies one EZ-AG instance and so 1 token to aggregate the value. Similarly, we have 4 level-2 tokens, one in each cell at level-2. By the time the level-3 token completes the aggregation one time, each level-2 token finishes the aggregation and dissemination 4 times. Hence, a total of 16 aggregations is expected.

4 tokens * 4 cells = 16 aggregations (expected)

Likewise, we have 16 level-1 tokens, one in each cell at level-1. By the time the highest-level token completes the aggregation one time, each level-1 token finishes the aggregation and dissemination 16 times. Hence, a total of 256 aggregations is expected.

16 tokens * 16 cells = 256 aggregations (expected)

From the aggregation results expected, we understand that the aggregates for cells at lower level is disseminated at exponentially faster rate in comparison to the cells at higher levels.

Chapter 4

Setup and Results of Hierarchical EZ-AG

In this chapter, we present the experimental system model we used and evaluate the results of hierarchical EZ-AG and compare them with EZ-AG (random walks with one-step PUSH phase) and gossip algorithm using simulations from network simulator, ns3.

4.1 System Model

4.1.1 Network Model

We model the topology for MANET's as described: placing 'N' nodes uniformly at random in the square region of side √R. Each node in the network has communication range of R i.e., it can listen and exchange messages with the nodes in the communication range, R. The network density of the deployed nodes is $= \frac{N}{R}$.

Consider the network region being divided into square cells of side $\frac{R}{\sqrt{2}}$ i.e. having the communication range as the diagonal R. Let $R^2 > \frac{2k \log(N)}{\rho}$, it's proved that there exists a constant k > 1 such that each cell has $\theta\ (\log(N))$ nodes at high probability. This implies that the degree of each node is $\theta\ (\log(N))$.

4.1.2 Mobility Model

Mobility in nodes introduces many challenges in deploying the nodes, data forwarding, energy consumption, data aggregation, and scalability and so on. Mobility model gives information about the pattern of movement of the nodes in the network, how the position velocity changes with time. Mobility models are broadly classified as memory-based and memory-less. In memory-based models, the nodes make use of the previous history or move in correlated fashion, or movement of nodes is bounded by obstacles.

Unlike memory-based models, memory-less models don't depend on the previous movement database. They move randomly without any restrictions. The destination, speed and direction are chosen at random without depending on other neighboring nodes. The distinct types of memory-less mobility models are random waypoint, random direction and random walk. Two different mobility models have been considered for our simulations are explained below.

1. *Random Waypoint Mobility Model*

In this model, the nodes randomly select a location in the simulation area as the destination and traverses with constant randomly chosen velocity from $[V_l, V_h]$. V_l represents the minimum velocity the node can move with and V_h represents the maximum velocity limit. The velocity and direction are chosen randomly independent to other nodes. On reaching the destination the node pauses for a duration called as "pause time", T_{pause}. In our case we have maintained the T_{pause} to be 2 seconds. After the pause time next step is taken and proceeds in similar fashion. In Random Waypoint Mobility Model V_l, V_h and T_{pause} affect the mobility of the nodes.

2. *Random walk 2D Mobility Model*

In this model, the node picks a random direction $\theta(t)$ from the 2D space range $(0, 2\pi]$ and a constant random velocity $v(t)$ from $[0, V_{max}]$ to move to the next point. At the end of move, new pair of direction and velocity is determined. We can consider Random walk mobility model as a specific variation of Random Waypoint Mobility model with pause time = 0. If the node reaches the boundary of simulation field, the node will be bounced back in the direction $\theta(t)$ or $\pi - \theta(t)$. This is called as the border/boundary effect.

The motion of the nodes in the network are independent to each other. The important characteristic of this model is that uniformity of the node distribution in the network is preserved over the time.

The key metrics that we consider for verifying hierarchical EZ-AG behavior in MANET's is by checking the number of aggregations obtained at distinct levels by the token before the aggregation of entire network is done once by highest level token. Message overhead (inclusive of token announcements, token requests and token transfers) in the entire network for varying sizes and mobility are verified. We have checked the consistency of the results obtained over 10 different simulations in ns-3 for varying the speeds and network size.

4.2 Performance Evaluation

We have set up a MANET network with nodes varying from 100 to 1000 using different mobility models discussed. The nodes are uniformly deployed in the network within a communication range of $R^2 = \frac{4*\log(N)}{\delta}$. The nodes are geo-densely placed i.e., at high probability each node has 2*log(N) neighbors. Also, the nodes in the network are moving at a speed varying between 3m/s to 21m/s. We have tested the above-mentioned network with varying mobility models: 2-D Random Walk and Random Waypoint. The network is hierarchically divided as described in chapter 4.

In this section, we evaluate the performance of hierarchical EZ-AG considering message overhead, token messages, number of aggregations at distinct levels, node speed and mobility.

4.2.1 Message Overhead

In this section, we present the results for hierarchical EZ-AG message overhead. We have considered total messages in this case. Total messages include the token announcements, token requests and token transfers.

$$Total\ Token\ Messages = (Token\ Announcements + Token\ Requests + Token\ Transfers$$

Token announcements are messages sent out by the token holder nodes to all the nodes in its communication vicinity. Token requests are messages sent to the token holder node by the nodes requesting for the token (i.e., nodes which have received the token announcement

messages). Token transfer are messages resulting from the transfer of token to one of the nodes that requested for the token (that satisfies the criteria mentioned in chapter 4).

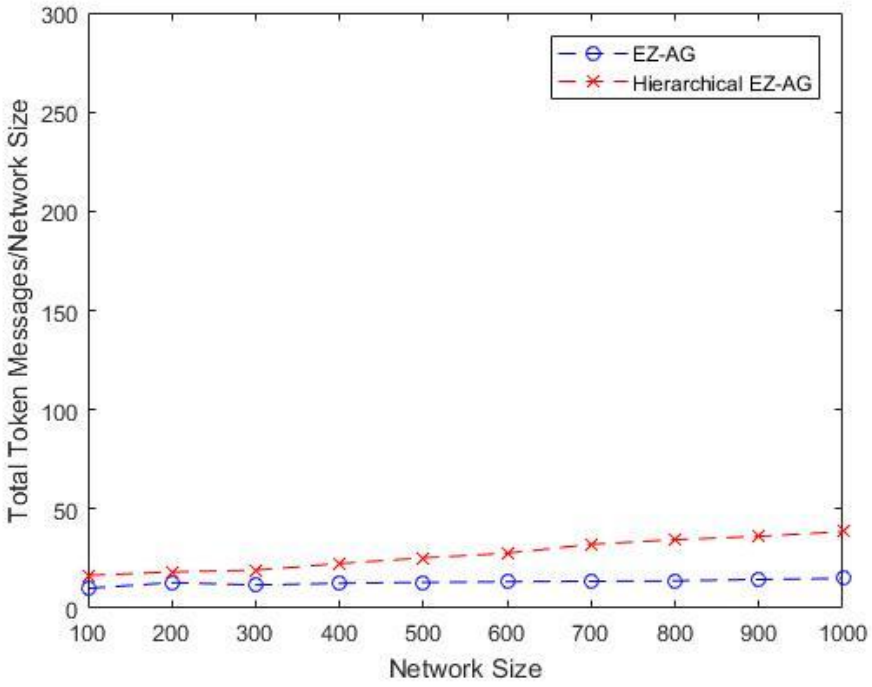

Figure 4.1 Number of messages per node for EZ-AG and Hierarchical EZ-AG

In Figure 4.1 we show total token message overhead of EZ-AG and hierarchical EZ-AG. In hierarchical EZ-AG, we have multiple instances of EZ-AG running at smaller network bound. To highlight log factor overhead we have plotted total token messages normalized by network size on y-axis and network size in x-axis. The results plotted are average obtained by 10 simulations. For EZ-AG, we see that total token messages remain almost flat / constant indicating that they grow linearly with network size indicating O(N) messages required to achieve data aggregation as values are normalized. In Hierarchical EZ-AG, we see total token messages following log(N) pattern indicating N log(N) required to achieve data aggregation.

4.2.1 Variance of Exploration Overhead

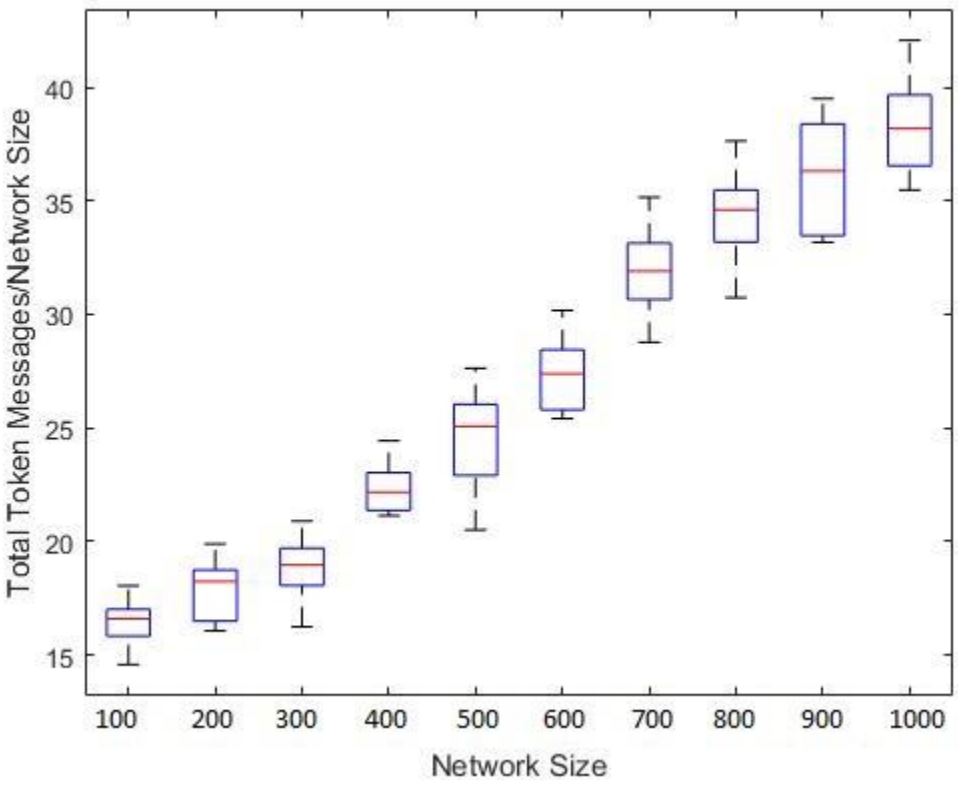

Figure 4.2 Variance in message overhead for Hierarchical EZ-AG

In Figure 4.2 and Figure 4.3 we show the variance in exploration overhead for hierarchical EZ-AG over 10 trials for different network sizes. We can also observe that the values are not highly varying across 10 simulations.

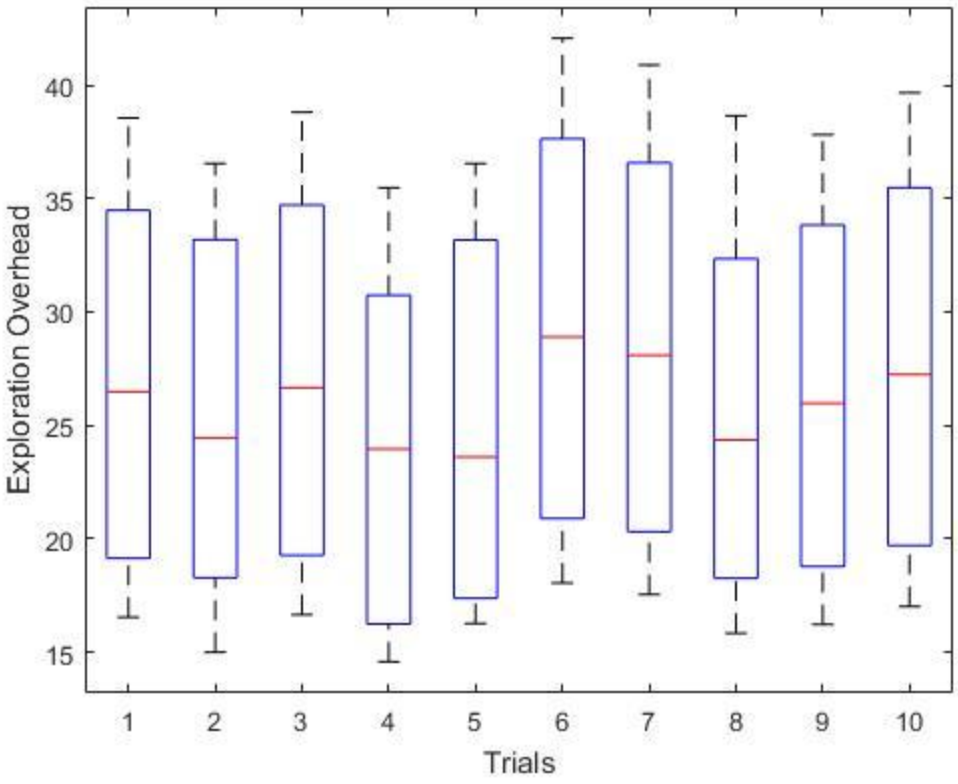

Figure 4.3 Variance in message overhead for Hierarchical EZ-AG over 10 trials

4.2.3 Node Mobility and Speed

In Figure 4.5, we study the impact of node mobility and speed on the exploration overhead. We see that the performance of hierarchical EZ-AG improves as the node mobility increases.

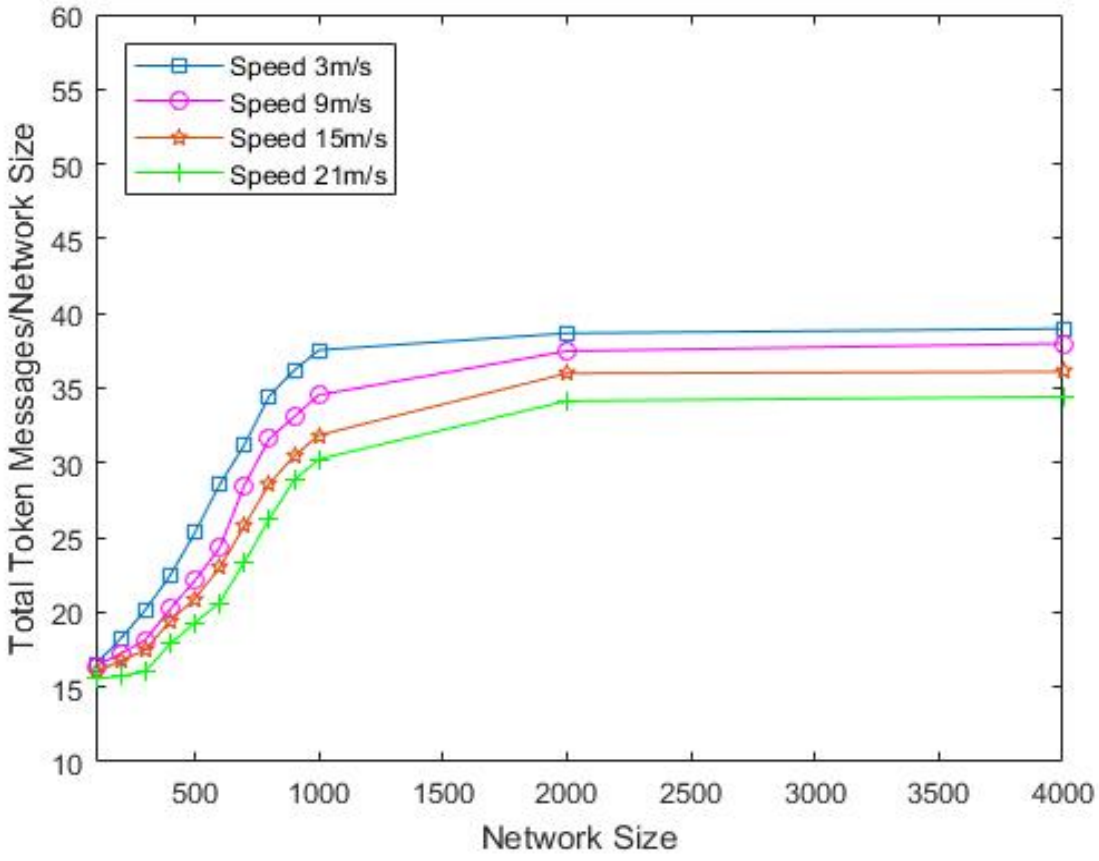

Figure 4.4 Total Messages as a function of Network size for varying speeds

4.2.4 Number of Aggregations at distinct levels

Figure 4.6 shows the number of aggregate values disseminated for a network divided into 2 levels. Level 0 represent the cells at lowest level and level 2 represent the cells at highest level. The cells at every level disseminate the results as and when the aggregation for that cell has been completed and it is not time synchronized.

By the time the level 2 cell completes the data aggregation of the entire network once, on an average we have obtained 235 aggregations at level 0 and 13 aggregations at level 1.

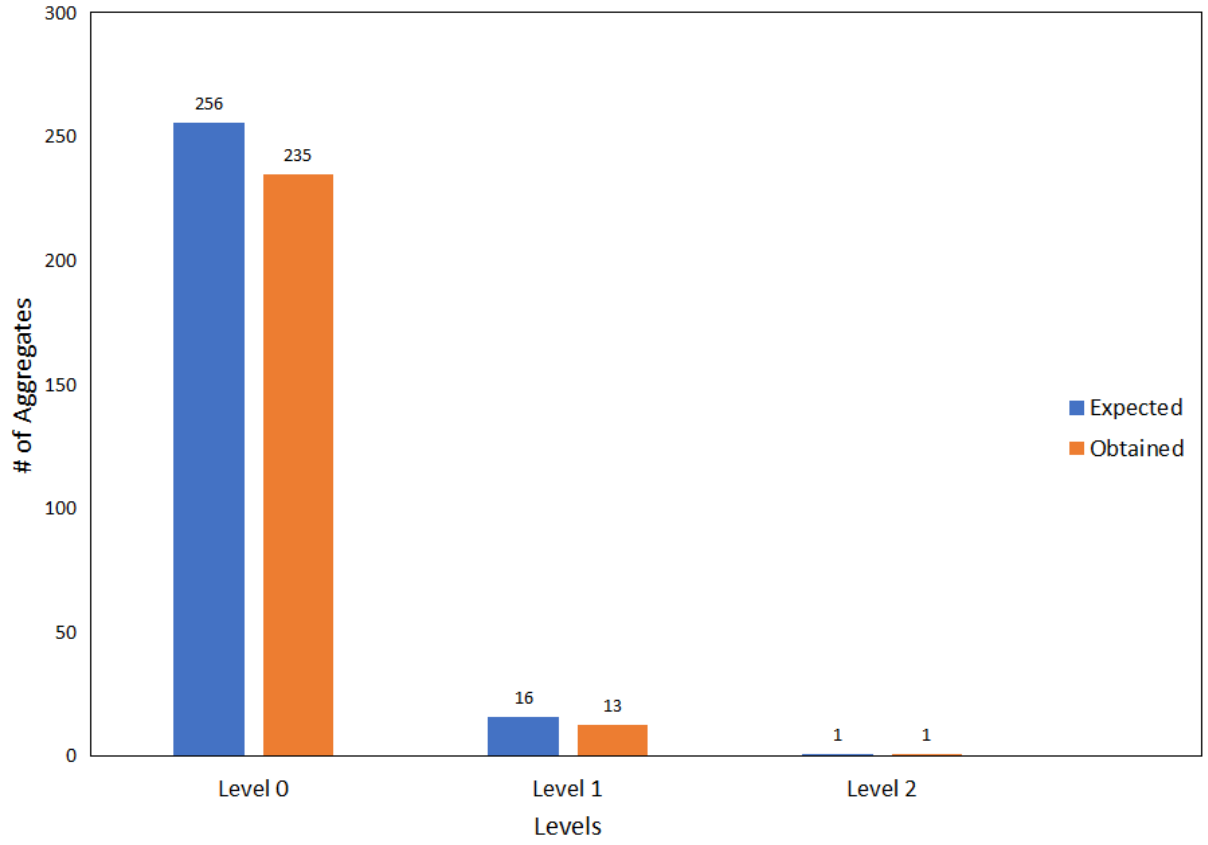

Figure 4.5 Number of Aggregates at distinct level in Hierarchical EZ-AG

4.3 Comparison with Gossip Techniques

Figure 4.7 gives the information of total messages are required per node to complete the data aggregation for EZ-AG, hierarchical EZ-AG and projected message for spatial gossip technique. EZ-AG and Hierarchical EZ-AG has very small message overhead whereas spatial gossip has significantly large message overhead. We see that message overhead grows rapidly as network size increases.

As described in [1], extension of spatial gossip technique requires O ($N \log^{5.4}(N)$) messages to obtain multi-resolution output whereas extension of EZ-AG requires only O ($N \log(N)$).

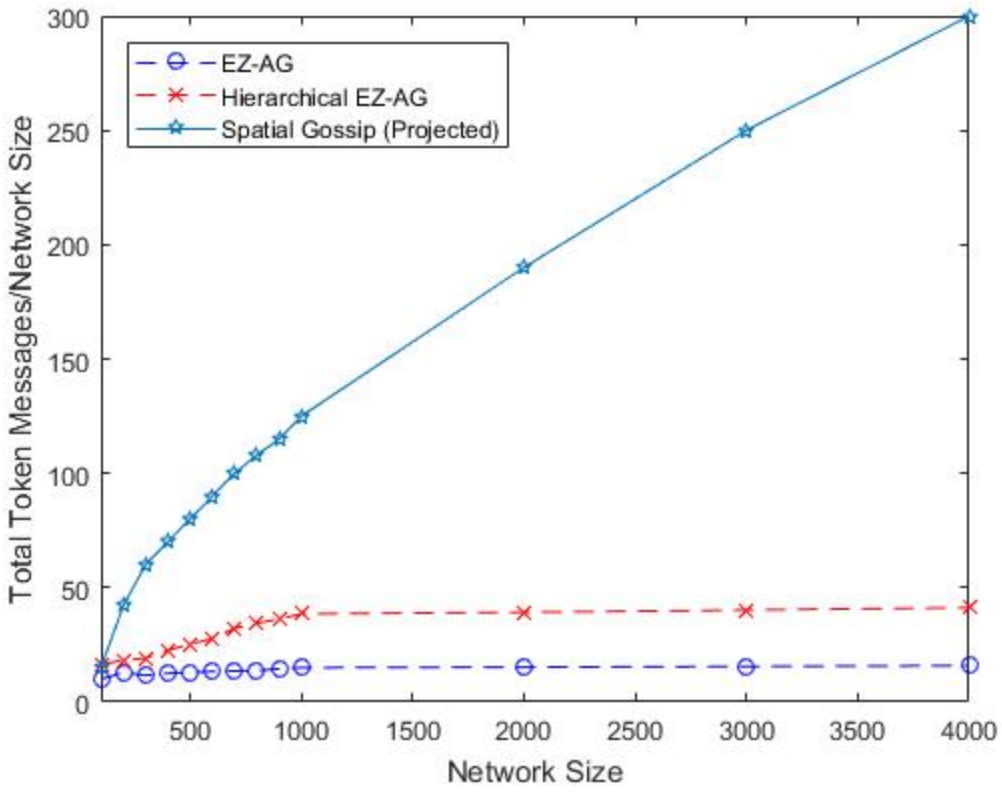

Figure 4.6 Message Overhead comparison of Hierarchical EZ-AG with Spatial Gossip (projected)

Chapter 5

Conclusion

In this report, we have presented the extension of EZ-AG, hierarchical EZ-AG which is used for obtaining multi-resolution data aggregation of duplicate sensitive and duplicate insensitive data in MANETs. We see that hierarchical EZ-AG obtain the aggregates for lower levels cells faster than the aggregates of cells at the higher levels. This is useful in obtaining the local information of the nodes more frequently.

We know that EZ-AG is a light weighted protocol. It makes minimal assumptions of underlying structure and requirements (it does not make any assumptions of the node locations or routing protocols used). These properties of EZ-AG are inherited by hierarchical EZ-AG.

We have showed that our protocol can achieve data aggregation in O (N log(N)) messages. In terms of message overhead, hierarchical EZ-AG outperforms the existing hierarchical technique by the factor of $O(\log^{4.4}N)$. We notice that increasing the mobility to the nodes in the network, the number of messages exchanged for achieving the data aggregation is reduced. We have quantified the performance using ns-3 simulations under different node speeds and mobility models.

References

[1] Vinod Kulathumani, Masahiro Nakagawa, Anish Arora, "EZ-AG: Structure-free data aggregation in MANETs using push-assisted self-repelling random walks", Aug 2017

[2] Vinod Kulathumani, Ken Parker, Mukundan Sridharan, and Anish Arora, “Census: A protocol for visiting all nodes in manets using biased random walks”, 2014.

[3] Noga Alon, Chen Avin, Michal Koucky, Gady Kozma, Zvi Lotker, and Mark R. Tuttle, "Many random walks are faster than one" In 20th Annual symposium on parallel algorithms and architectures, 2008.

[4] A. Clementi, A. Monti, F. Pasquale, and R. Silvestri , "Information spreading in stationary markovian evolving graphs. IEEE Transactions on Parallel and Distributed Systems, vol. 22, no. 9, pp. 1425–1432, Sept 2011.

[5] A.Arora. V.Kulathumani, M.Nakagawa, “Converge characteristics of self-repelling random walks in mobile ad-hoc netowrks. https://www.csee.wvu.edu/ vkkulathumani/srrw.pdf,” .

[6] Chen Avin and Bhaskar Krishnamachari, “The power of choice in random walks: An empirical study,” in Proceedings of the 9th ACM International Symposium on Modeling Analysis and Simulation of Wireless and Mobile Systems, New York, NY, USA, 2006, ACM.

[7] I. F. Akyildiz w. Su Y. Sankarasubramaniam E. Cayirci "A Survey on Sensor Networks", IEEE Commun. 2002

[8] Xiang-Yang Li Ivan Stojmenovic "Broadcasting and topology control in wireless ad hoc networks", Handbook of algorithms for wireless networking and mobile computing, July 2004.

[9] Stephanie Lindsey Cauligi S. Raghavendra "PEGASIS: Power-Efficient Gathering in Sensor Information Systems" Proceedings of the IEEE Aerospace Conference, March 2002.

[10] V. Kulathumani, A. Arora, M. Sridharan, K. Parker, and B. Lemon. On the repair time scaling wall for manets. IEEE Communications Letters, 2016.

[11] Subhajit Pal, Debnath Bhattacharyya, and Tai-hoon Kim, "Chain based hierarchical routing protocol for wireless sensor networks. 2009

[12] Colin Cooper, Alan Frieze, and Tomasz Radzik, “Multiple random walks in random regular graphs,” SIAM J. Discret. Math., Nov. 2009.

[13] Uriel Feige, “A tight upper bound on the cover time for random walks on graphs,” Random Struct. Algorithms, Jan. 1995

[14] Tracy Camp, Jeff Boleng, and Vanessa Davies, “A survey of mobility models for ad hoc network research,” WIRELESS COMMUNICATIONS and MOBILE COMPUTING (WCMC): SPECIAL ISSUE ON MOBILE AD HOC NETWORKING: RESEARCH, TRENDS AND APPLICATIONS, 2002.

[15] David Bertrand Fotue Fotso. "Efficient data aggregation and routing in wireless sensor networks". Networking and Internet Architecture. T´el´ecom ParisTech, 2013.

[16] MJAM Brummelhuis and HJ Hilhorst, “How a random walk covers a finite lattice,” Physica A: Statical Mechanics and its Applications, 1992. Gunti Spandan1, Archana Patel 2, C R Manjunath3 , Nagaraj GS4

[17] Gunti Spandan, Archana Patel, C R Manjunath, Nagaraj G, " Data Aggregation Protocols in Wireless Sensor Networks.", International Journal of Computational Engineering Research, 2010

[18] Kai-Wei Fan, Sha Liu, Prasun Sinha, "Structure-Free Data Aggregation in Sensor Networks", IEEE TRANSACTIONS ON MOBILE COMPUTING, 2007

[19] M. Bala Krishna, Noble Vashishta, "Energy Efficient Data Aggregation Techniques in Wireless Sensor Networks.", 5th International Conference on Computational Intelligence and Communication Networks, 2013

[20] Rik Sarkar, Xianjin Zhu, and Jie Gao, “Hierarchical spatial gossip for multiresolution representations in sensor networks,” ACM Trans. Sen. Netw., Aug. 2011

[21] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, “Randomized gossip algorithms,” Information Theory, IEEE Transactions on, vol. 52, no. 6, pp. 2508–2530, June 2006.

[22] M.G. Rabbat, “On spatial gossip algorithms for average consensus,” in Statistical Signal Processing, 2007. SSP ’07. IEEE/SP 14th Workshop on, Aug 2007.

[23] R. Friedman, D. Gavidia, L. Rodrigues, A. Viana, and S. Voulgaris. Gossiping on manets: The beauty and the beast. SIGOPS Oper. Syst. Rev., October 2007

[24] D. Kempe, J. M. Kleinberg, and A. J. Demers. Spatial gossip and resource location protocols. In ACM Symposium on Theory of Computing, 2001.

[25] A.Arora. V.Kulathumani, M.Nakagawa, “Converge characteristics of self-repelling random walks in mobile ad-hoc netowrks. https://www.csee.wvu.edu/ vkkulathumani/srrw.pdf,” .

[26] S Sasirekha, S Swamynathan2, "A Comparative Study and Analysis of Data Aggregation Techniques in WSN". Indian Journal of Science and Technology, October 2015.

[27] The ns-3 network simulator.,. (n.d.). Online - last accessed 2-June-2016.

Appendix

Network Simulator – 3 (NS3)

NS-3 is an open-source and discrete-event based network simulator useful for research and educational purpose. NS-3 provides models of how data packets work, provides simulation engines for users to conduct experiments. The model sets available in NS-3 is useful to model how Internet protocols works, alongside non-Internet- based systems can also be simulated. NS-3 is not a suitable platform to study real-time and highly controlled systems.

Features in NS-3

1. NS-3 is primarily used on Linux systems, it can also be used for Windows (Cygwin/FreeBSD)
2. NS-3 consists of set of software libraries that can be combined with other external software packages. Data Visualization tools, animation tools can be used. User should work in the command line with C++ and/or python.

ns-3 provides

1. Nodes (used for managing the representations of computing devices in simulations),
2. Channels for communications (CsmaChannel, PointToPointChannel and WifiChannel),
3. Net device useful for communicating with other nodes via channel.
4. Topology helpers useful for creating network/simulation topology i.e installing the nodes in the simulation area, applying mobility models and so on.

Development Environment

Scripting in ns-3 is done in C++ or Python. The ns-3 API's are written in Python. Socket programming API's are also provided in the ns-3 bundle.

Downloading NS-3 using tarball

ns-3 has a complex system and many dependencies on other components. There is online wiki available to aid the process. The source code is available in Mercurial repositories on server http://code.nsnam.org. One can download the tarball release at http://www.nsnam.org/releases .

Tarball is archive format where multiple files are bundled together and compressed. Downloading with tarball is very simple, choose the release download the compressed file and decompress it.

```
$ cd
$ mkdir workspace
$ cd workspace
$ wget http://www.nsnam.org/releases/ns-allinone- 3.17.tar.bz2
$ tar xjf ns-allinone- 3.17.tar.bz2
```

If we change to directory, ns-allinone- 3.17 we can see the following files:

Bake	constants.py	ns-3.17	README
build.py	netanim-3.103	pybindgen-0.16.0.825	util.py

Building with Waf

This is one of the tool useful for building ns-3 and supporting libraries. The working directory must be changes to ns-3 directory. The first command is not mandatory, but it's a good practice to remove out the dependency files generated from the previous builds. The second command is used for re-configuring and dependencies are verified.

```
$ ./waf clean
$ ./waf -d optimized -- enable-examples -- enable-tests configure
```

Some of the ns-3 libraries are not enabled by default, so we should run the below commands to enable the examples and tests.

```
$ ./waf -d debug -- enable-examples -- enable-tests configure
$ ./waf configure -d debug -- enable-sudo -- enable-examples -- enable-tests
$ ./waf configure -d debug -o build/debug -- enable-examples -- enable-tests
```

Testing ns-3

Unit tests of ns-3 are executed by running the following script.

$ ./test.py -c core" script

This script should result in test cases passed output.

92 of 92 tests passed (92 passed, 0 failed, 0 crashed, 0 valgrind errors)

Running a script

We run the scripts in ns-3 under the control of waf which ensures that shared library paths are set correctly and available at run time.

$ ./waf -- run hello-simulator

Once ns-3 is setup we can build our functionality, use the packages provided as and when required.