MASTER

Range-assignment for broadcasting in mobile ad-hoc networks

Tiwari, A.

Award date:
2016

Master's Thesis

Eindhoven
August 22, 2016

Range-assignment for broadcasting in mobile ad-hoc networks

Ankur Tiwari
Software Science
Master in Computer Science and Engineering
Eindhoven University of Technology

Supervisor : **Professor Mark de Berg**
Mentor : **Professor Kevin Buchin**
Algorithms Group
Eindhoven University of Technology

Acknowledgements

I would like to take this opportunity to extend my sincere gratitude and appreciation to all the people who have helped me and extended their support during this Master Thesis. Without any one of them, my accomplishments would not be possible.

First of all, I would like to thank my parents for all the efforts they have put to provide me with the best possible education, as per my choice and the freedom, to pursue my ambitions.

I am grateful to my supervisor Prof. Mark de Berg to provide me with an opportunity to do my thesis under his supervision and making me a part of Algorithms Group. This thesis could not be completed without his helpful comments and guidance. Special thanks to professor Kevin Buchin for being very co-operative and helpful in providing knowledgeable insights throughout the project. I feel very proud to be a student of my professors Mark de Berg and Kevin Buchin. I would also like to thank Prof. Bart M.P. Jansen for taking the time to always clarifying my doubts and providing me with an interesting learning experience.

Nevertheless, I want to thank my sister, Gunjan Tiwari, and my friends who have supported me in the much longer journey over the past two years and for all the cherished experiences.

Ankur Tiwari
Eindhoven, August 22, 2016

Contents

Chapter 1

Introduction

A mobile ad-hoc network (also known as MANET) is a wireless network containing self-configuring mobile sensor devices connected wirelessly. It is an infrastructure-less network with no stationary base station or central device to keep track of the activities of sensor devices. The devices can change location arbitrarily, and configure itself on the fly. The devices in the network are connected via WiFi or any other wireless connection (cellular or satellite). The main purpose of the devices in the network is to forward data traffic towards the destination devices. As the sensor devices are mobile and they can easily re-configure their properties, the devices switch to different transmission ranges to keep other devices within range for successful data transmission. Therefore, the resulting topology in the network is highly dynamic with respect to time.

Mobile ad-hoc networks have a wide range of practical applications. There are different types of these networks depending on its application domains, such as VANETs (Vehicular Ad Hoc Networks), SPANs (Smart Phone Ad Hoc Networks), military and tactical MANETs. VANETs, for example, are mainly used to collect data from vehicles. This type of networks have roadside equipment collecting data from the mobile vehicles. The data collected in this network is used for traffic analysis, implementing road-safety measures, and huge range of other applications. The data is also communicated (V2V communication [2]) to other vehicles in the network to prevent accidents by estimating distance between two vehicles by sending their location details to each other. VANETs are also heavily researched in the field of driver-less cars to drive safely by collecting data from near-range vehicles. This technology uses moving cars and roadside base stations as nodes to create a mobile network. The nodes communicate their information to other nodes directly, and via several other nodes.

Another type of mobile ad-hoc network is used in military applications. Con-

sider a war affected zone and a military search operation. Each military personnel carries wireless communication device and they spread in the zone. The devices are used to communicate with other devices by forwarding messages. Depending on the requirement, it could be a 2-way communication or a broadcast operation from a single source device.

The latest innovations in self-driving cars, smart and cheap mobile devices, communicating cars on the road, etc. are making mobile ad-hoc networks an important research topic. A highly dynamic and autonomous nature of the sensor devices give rise to many algorithmic problems related to topology control. Topology control is very important in these networks for variety of reasons such as to provide energy-efficient operations and physical security mainly in the case of vehicular network. The network topology is very dynamic and typically multihop, and it could contain uni-directional or bi-directional links. A network may need to maintain one or more of the following operations using a network topology: broadcast, strong-connectivity, network packet collision-prevention, contention within the network.

In this thesis, we study the topology problem of maintaining successful broadcast operation within a mobile ad-hoc network. A broadcast requires a source node S to send data packets to destination nodes, either directly (1-hop) or via other nodes (multihop). The variant of broadcast operation, that is studied in this report, is concerned with sending broadcast data to every other nodes as destination in the network. The sensor devices in the network are considered as *nodes* with varying locations, and a node A can send a message to another node B if the node B is inside the circular range of the node A. This is denoted with an directed edge from the node A to the node B. It gives rise to a graph which is called *communication graph*. The problem of topology control in the network requires to maintain a connected communication graph in the network with certain properties. For successful broadcast the communication graph should contain a directed spanning tree rooted at the source node S. The directed spanning tree is referred as *broadcast tree* in this study. The nodes in the network are under mobility, which requires the nodes to maintain their range to maintain the broadcast tree over time. Our goal is to adapt a broadcast tree over time while keeping the energy consumption low. As shown in the figure 1.1, the nodes extend their ranges in order to maintain the broadcast tree within the network.

We propose an algorithm for the topology problem discussed in the previous paragraph, which is referred to as *VORONOI_RA*. The algorithm uses geometric data structure, Voronoi diagram, to calculate range-assignment function that assigns range to the nodes in the network to maintain the broadcast tree while trying

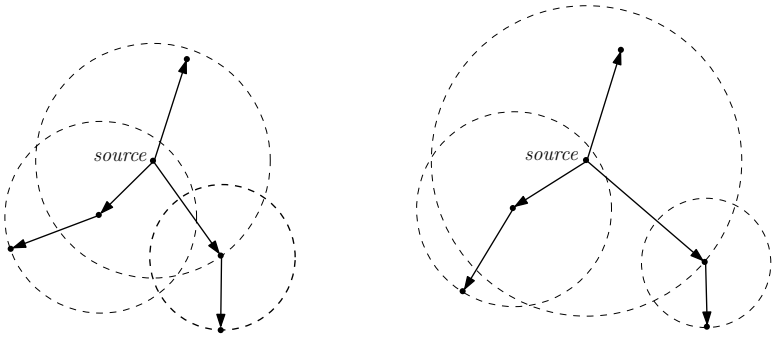


Figure 1.1: (*left*) the initial position of the nodes and (*right*) the position of the nodes after movement

to increase the energy efficiency of the network. We also present different variants of the algorithm using different heuristics. The results of the *VORONOI_RA* algorithm is compared with the existing approach to maintaining broadcasting under nodes mobility. We define three performance measures: MAX, AVG, and TOTAL for maximum, average, and total energy consumption respectively in the network. The proposed algorithm performs better than the existing algorithms in terms of MAX performance measure. For AVG and TOTAL, the algorithm performs efficiently, but the performance decreases for few settings of the network that are discussed in details later in the thesis.

Before discussing the algorithms, we present a formal description of the algorithmic problem. A wireless network contains a set of nodes $\mathcal{N}$. The nodes in the network are moving, and the location of a node u at time t is denoted with $u(t)$. A range assignment function $r : \mathcal{N} \rightarrow \mathbb{R}^+$ assigns a range-assignment function $r(u,t)$ to a node $u \in \mathcal{N}$ which is proportional to the energy consumption at the node u at time t. The energy consumption $p(u,t)$ at any node u at time t is considered to be $C \cdot r(u,t)^\beta$ where $\beta \geq 1$ is a distance power gradient and C is a constant. With this, we can say that a node u will consume $C \cdot r(u,t)^\beta$ amount of energy to maintain a range of $r(u,t)$ at time t. When the network is static or time does not matter in the context, we use $r(u)$ and $p(u)$ for range and power of the node u respectively. The overall energy consumption in the network at time t is:

$$p(\mathcal{N},t) = \sum_{u \in \mathcal{N}} p(u,t) = C \cdot \sum_{u \in \mathcal{N}} r(u,t)^\beta$$

As mentioned earlier that the topology of a network can be represented using a communication graph $G = (\mathcal{N}, E)$ which can be a directed or undirected graph. In a directed communication graph, E is the set of directed edges and an edge$(u,v) \in E$ exists iff $r(u)$ is at least the euclidean distance $d(u,v)$ between the nodes, i.e. $E = \{(u,v) : r(u) \geq d(u,v)\}$. In an undirected communication graph, the edge set E contains undirected edges, and an edge $(u,v) \in E$ exists when $r(u)$ and $r(v)$

both are at least the euclidean distance between the two nodes, i.e. $E = \{(u, v) : r(u) \geq d(u, v)\ and\ r(v) \geq d(u, v)\}$; otherwise the edge (u, v) does not exist.

In this thesis, we consider the directed communication graph which should contain a directed spanning tree (*broadcast tree*) rooted at the source node $s \in \mathcal{N}$ for successful broadcasting. The broadcast tree $T_B(t)$ is maintained in the communication graph over time. The range assigned to the nodes by the range-assignment function maintains the broadcast tree over time while keeping the energy consumption low.

Chapter 2

Background

Mobile ad-hoc networks are required to be highly reliable to be successfully used in different applications. For example, a VANET must be reliable to prevent road accidents, and to predict the traffic conditions accurately. For military operations, the nodes in the network must remain connected to allow successful communication under critical situations. MANETs are heavily researched in the latest technologies of driver-less cars. Range-assignment problems in a MANET are very important in order to maintain the reliability of the network. Range-assignment problems have been studied in a large number of research works. In this chapter, we discuss existing research works in the field of range-assignment problem in sensor networks.

The algorithm proposed by Li *et al.* [3] is a distributed algorithm and maintains a local directed MST at each node under stationary setting. In a stationary setting, the nodes have no movement in the network. Furthermore, the energy consumption at the nodes is adjusted to get bidirectional edges in the output communication graph. Under limited mobility, it uses a probabilistic model to decide when a node might move in or move out of the range of another node, and the topology is fixed accordingly. Rodoplu and Meng [7] proposed a distributed position-based algorithm that maintains a strongly connected directed communication graph, and the optimization criterion is to minimize the maximum power consumption at any node in the network. The algorithm periodically executes the proposed algorithm to fix topology under mobility. Ramanathn and Rosales-Hain [6] introduce two centralized algorithms for maintaining connectivity, and bi-connectivity. The author proposed two distributed heuristics to support connectivity under mobility. The optimization criterion used by the authors of the research papers [3], [7], and [6] is to minimize total power consumption in the network. The authors of the research papers [10], [4] present algorithms to maintain a movement-connected network under strict assumptions on the speed of the nodes and direction of the motion. Zhao *et al.* [10] present a centralized algorithm to maintain connectivity

within a unit time interval under nodes mobility in straight line and constant velocity. Li *et al.* [4] relaxes the assumptions used in [10] and allow variable speed; they present both centralized, and distributed algorithms to maintain movement-connectivity. The power optimization criteria is not discussed by the authors of the research papers [10] and [4]. Wu and Dai [9] propose a method where the nodes are under mobility with no constraints on speed and direction. The algorithm tries to remove inconsistent views as shown in figure 2.3 and uses one of the distributed algorithm (LMST or relay-region based) presented in other papers to create a communication graph. Finally the power at the nodes is extended to get a *buffer* region such that their direct neighbors in the graph remain connected under mobility.

Li, Hou and Sha [3] proposed a minimum spanning tree based distributed topology control algorithm, called local minimum spanning tree (LMST). The effective topology control algorithm preserves network connectivity with bidirectional edges and minimal possible energy consumption. LMST is a localized approach that depends only on the information collected within one hop. The authors denote network topology under maximum transmission range d_{max} as $G = (V, E)$ where V is the set of nodes and $E = \{(u, v) : d(u, v) \leq d_{max}\ and\ u, v \in V\}$. It defines a Visible Neighborhood set $NV_u(G)$ as $\{v \in V(G) : d(u, v) \leq d_{max}\}$ for each node $u \in V$, and $G_u = (V_u, E_u)$ is the induced graph of G where $V_u = NV_u(G)$. The algorithm runs in several phases:

1 *Information Collection*: Nodes broadcast *HELLO* message with max power, and each node u makes a list of their Visible Neighborhood $NV_u(G)$.

2 *Topology Construction*: Every node builds its local MST $T_u = (V(T_u), E(T_u))$ of G_u using neighborhood information $NV_u(G)$ and uses transmission power ($C \cdot d^{\alpha}$, C is a constant) between nodes as edge weights. It uses a *weight function* to assign distinct weights to every edge to get a unique MST. This function is useful when two or more edges have equal weights such that the weights remain the same but it gives a higher priority to one of the edges based on its higher index. A node u is said to be the neighbor of a node v if and only if there is a directed edge $(u, v) \in E(T_u))$ and this is denoted with $u \rightarrow v$, and the neighbor set of u is $\{v : u \rightarrow v\}$. Note that this relation is not symmetric as shown in figure 2.1. In the figure, the node u selects the node v as a direct neighbor in its LMST while the node v has an indirect path to the node u. A connected topology is derived under LMST of nodes which is a directed graph $G_0 = (V_0, E_0)$ where $V_0 = V$ and $E_0 = \{(u, v) : u \rightarrow v\ and\ u, v \in V(G)\}$.

3 *Determination of Transmission Power*: Each node can determine from the *HELLO* messages the transmission power it requires to reach its neighbors. Radius of a node u is the maximum euclidean distance between u and its farthest

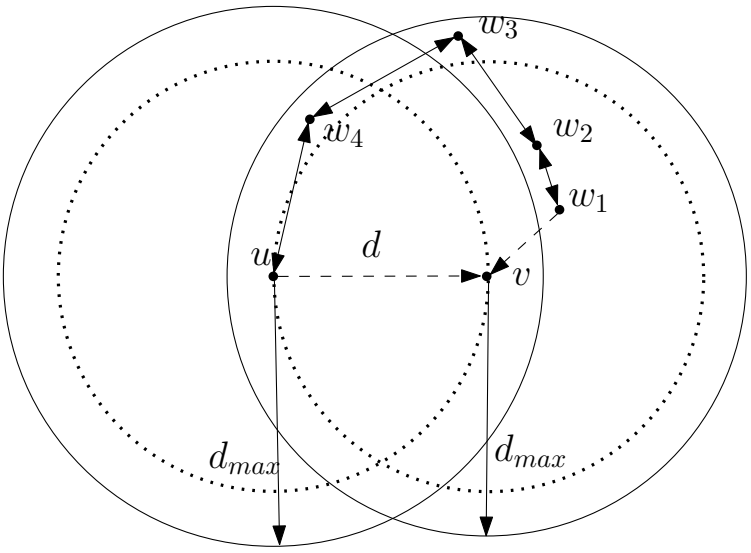


Figure 2.1: LMST of two nodes u and v

neighbor in $N(u)$. Note, $N(u)$ is the set of immediate neighbors of node u, while $NV_u(G)$ is the set of one hop neighbors when maximum energy is used at node u. The transmission power of the node is simply $c \cdot r^{\alpha}$.

4 *Construction of Topology With Only Bidirectional Edges*: Some links in G_0 are unidirectional as shown in figure 2.1. In this phase, they term two topologies G_0^+ and G_0^-, wherein G_0^+ all unidirectional edges are converted into bidirectional edges and in G_0^- all unidirectional edges are deleted.

The main results from the research paper is that the graph G_0 preserves connectivity of graph $G = (V, E)$ where $E = \{(u, v) : d(u, v) \leq d_{max}\ and\ u, v \in V\}$. And the bidirectional graphs G_0^+ and G_0^- preserves the connectivity of G_0. All of the graphs obtained under LMST are having a degree bound of 6, i.e. $deg(u) \leq 6$ for all $u \in V(G_0)$. The topology control algorithm under LMST is energy efficient as per the comparisons made with other algorithms like R&M [7]. LMST attempts to reduce contention by having bounded degree of nodes, and maximizes spatial reuse and network capacity by constructing local MST. In case of limited mobility, a probabilistic model is used where the probability that a node w moves into (or moves out) of the range of a node u is used to repair LMST. The downside of this approach is that the constructed topologies (communication graphs G_0, G_0^+, G_0^-) are not bi-connected, i.e. in case a node fails then the graph might get disconnected. This can usually happen in case of mobility when a node goes out of the range of any other nodes.

Rodoplu and Meng [7] present a distributed position-based algorithm to create

a minimum energy network. For stationary network, the author introduces the notion of *relay region*. For a node u that intends to transmit to node v, the node v is said to be in relay region of a third node w if the node u will consume less power in transmitting to node v via node w instead of transmitting directly. It also presents the notion of *enclosing set E(u)* of a node u which is a set of all the nodes that do not lie in the relay regions that are formed by node u with other nodes in its range. An *enclosure graph* is a graph where an edge in the edge set is a directed link between nodes u and v such that node v can be reached directly from node u. The algorithm runs in two phases and maintains strong connectivity in the final topology such that there is a directed path between any two nodes $u, v \in G$.

Phase 1 *Search for Enclosure*: Each node i executes a local search to find the enclosure set. It examines the neighboring nodes that a node can reach using maximum transmission power and keeps only those that do not lie in relay regions of any previously found nodes.

Phase 2 *Cost Distribution*: Each node i runs distributed Bellman-Ford shortest path algorithm upon the enclosure graph using power consumption as the cost metric. $C_{u,v}$ is defined as the cost of sending a packet from a node u to a node v. A node u picks the link corresponding to the minimum cost neighbor and $Cost(u) = \min_{v \in N(u)} C_{u,v}$

A strongly connected and minimum power topology for stationary network is presented in their research work. In the case of mobility, the nodes periodically execute the distributed algorithm to self-configure. The nodes usually go to sleep mode to prevent power consumption. When the nodes wake-up then they re-compute the costs to fix the minimum-energy topology. Another assumption is made on the velocities of the nodes which uniformly lies in the interval (v_{min}, v_{max}) where v_{min} and v_{max} are minimum and maximum allowed velocity.

Ramanathan and Rosales-Hain [6] presented two centralized algorithms, *CONNECT* and *BICONN-AUGMENT*, to maintain connectivity and bi-connectivity respectively in the network. The *CONNECT* problem is that given a wireless network with set of nodes $\mathcal{N}$, and a power function $p(u, t)$, they present algorithm that finds a per-node minimal transmission power assignment $p(u)$ where $u \in \mathcal{N}$, such that the induced communication graph is connected, and $\max_{u \in \mathcal{N}} p(u)$ is minimum. *CONNECT* is a greedy algorithm that iteratively merges connected components unless only one connected component is left. It selects edges between nodes in increasing order of their euclidean distance which is similar to a MST algorithm. The paper describes a post-processing algorithm as a subroutine to *CONNECT* which further minimizes per node transmission power. *BICONN-AUGMENT* extends the *CONNECT* algorithm to find per node minimal power increase $\delta(u)$ such that the induced graph is biconnected, and $\max_{u \in \mathcal{N}} p(u) + \delta(u)$ is a minimum. This is again

a greedy algorithm that finds all biconnected components in the graph obtained from the *CONNECT* algorithm using depth-first-search. Then, the node pairs are selected in increasing order of euclidean distance, and joined if they belong to different biconnected component. This gives per node minimal transmission power while maintaining bi-connectivity. The worst-case running time of the presented algorithms is $O(n^2 \log n)$.

Ramanathan *et al.* also present two distributed heuristics for topology control in the case of mobile networks: Local Information No Topology (LINT) and Local Information Link-State Topology (LILT). In LINT, each node is configured with parameters- the *desired* node degree d, a high threshold on the node degree d_h, and a low threshold d_l. Every node periodically checks the number of active neighbors. The node increases its transmission power if the degree is less than d_l, and decreases if its greater than d_h. The magnitude of power change depends on current degree d_c and the desired degree d. LILT further improves LINT by overriding the high threshold when the topology change indicated by routing updates results in undesirable connectivity. LILT is also extended to maintain bi-connectivity in the induced graph.

Zhao, Lloyd and Ravi [10] proposed a centralized algorithm for topology control in a mobile network and uses an undirected graph to represent the network. The algorithm is based on time slicing approach, i.e. the lifetime of the mobile network is sliced into unit time intervals. It is assumed that the nodes move at constant velocity and direction within an unit interval of time. It introduces the concept of *distance function* and *threshold function* based on vector calculations. The distance function $d_{AB}(t)$ between nodes A and B gives the distance between two moving nodes at any time t. The threshold function $\pi(A,B)(t) = d^{\alpha}_{AB}(t)$ decides possible connectivity between nodes A and B. The threshold between A and B in a time slot $[t_g, t_h]$ is defined as $\pi(A,B)[t_g,t_h] = MAX(\pi(A,B)(t_g), \pi(A,B)(t_h))$. The authors present a decision version and an optimization version of the algorithm. The decision version of the algorithm simply checks the connectivity of the *threshold* graph where an edge weight is $\pi(A,B)[t_g,t_h]$ in a time slot $[t_g,t_h]$. The optimization version of the algorithm finds the minimum power p_{min} under which network $\mathcal{N}$ is movement-connected. This version defines p_{min} as the maximum power required for movement-connectivity in a unit time intervals $[t_h,t_g]$. The optimization version runs in worst case time $O(n^4(\log(n))^2)$.

The algorithm presented in this research paper maintains a movement connected topology under strict assumptions of constant velocity and direction within a unit time interval. The approach minimizes the power consumption while maintaining connectivity, but the power consumption is not minimum as a uniform power (maximum) is assigned to the nodes within a time interval $[t_g,t_h]$.

Aiming to solve the problem in above paper by Zhao *et al.* [10], Li *et al.* [4] presented a centralized and a distributed topology control algorithms for Variant Rate Mobile Networks (VRMN). The nodes can move with variant velocities in the model presented in this paper within a unit time interval (nodes move in constant velocities in [10]). Here, they adopt the same model where power of a node u is $p(u) = C \cdot r^{\beta}$, where C is a constant and β typically lies in $[2, 4]$. In a time interval, the distance between two nodes is the maximum distance between their endpoints and it is denoted with $len(u, v)$. As shown in figure 2.2, the distance between nodes in a time interval is taken as euclidean distance between $sp(j)$ and $ep(i)$, where ep is the endpoint and sp is the starting point of a node movement.
The centralized version of the algorithm works in three phases:

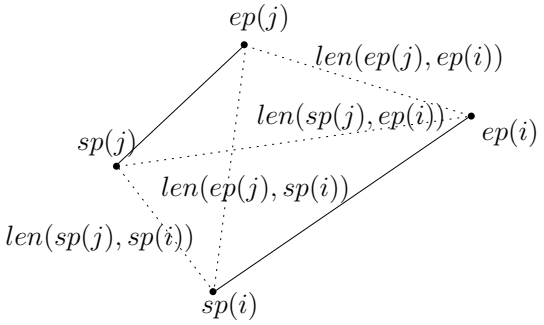


Figure 2.2: LMST of two nodes u and v

- compute the initial neighborhood list $N(u)$ of one hop neighbors for each node u using maximum power p_{max} at each node.
- Each node u then broadcasts its neighborhood list $NL(u)$ to the neighbors in the list. If for some node $v \in NL(u)$, there is a shorter path via another node $k \in NL(u)$ then the entry of node v is deleted from the neighborhood list of the node u by the node k. The resulting neighborhood list of the node u will contain the nodes which can be reached directly from u and that is the shortest path.
- Last step is to use breadth-first-search to check if the resulting topology is connected. If not, it simply assigns maximum power to each node in the network $\mathcal{N}$. Otherwise, the power assigned to a node is the power required to transmit a packet to the farthest node in its final neighborhood list.

The distributed version has similar steps, except that the neighbor list is created by the *hello* messages received from the neighboring nodes which is sent at maximum power p_{max}. Each node u broadcasts its list $NL(u)$ to its neighboring nodes, and only the nodes in the list ($v \in NL(u)$) will accept the list because some nodes, not in the list, may receive the list as they have come later in the range due to

movement. Each node u sends a *probe* message to the nodes in the list $NL(u)$ to check for relay nodes, and accordingly deletes any nodes in the list that has a relay node w in between. Similar to the centralized version, it uses a breadth first search to check connectivity, and assign power as done in centralized version. Both, the centralized and distributed versions run in $O(n^3)$ time, and maintain a movement-connected network under nodes mobility with variable speeds.

Jie Wu and Fei Dai [9] give an algorithm to maintain connectivity using localized topology control methods like MST-based topology control. The approach also tried to remove inconsistent views where a node is not detected as a logical neighbor because of asynchronous *hello* messages and the node movement (as shown in figure 2.3). In their approach, the LMST method is used to create topology which also maintains consistent view under both synchronous and asynchronous *hello* messages. The concept of buffer zone is used which extends the transmission range of the nodes to prevent connectivity under mobility. The extended transmission range depends on the moving speed of the mobile nodes.
The approach used in this research work maintains a movement-connected topol-

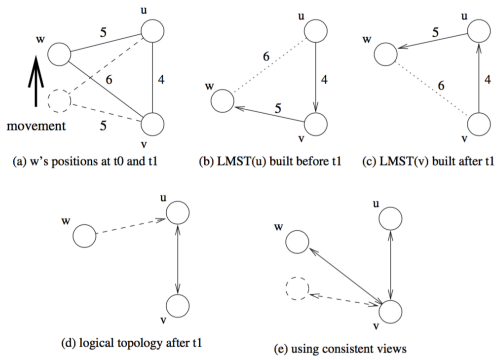


Figure 2.3: Inconsistent view in a 3-node network (taken from [9])

ogy by extending many movement-insensitive algorithms, like MST-based, Relay region based. There are other research papers by Wu *et al.* [8] and Lou *et al.* [5] towards broadcast reliability which extend the research work done by Wu and Dai [9] to maintain broadcasting under nodes movement.

Roeloffzen *et al.* present kinetic data structures in a black-box model. In this model, the location $p(t)$ of an object p becomes available at regular time steps. The position of the objects are retrieved at different time steps t_i. Also, it makes a *displacement assumption* which is the maximum displacement of any moving object at any time step t_i and it is defined as

$$d_{max} \leq mindist_k(P(t_i))$$
$$dist(p(t_i), p(i_{i+1})) \leq d_{max}, \forall p \in P$$

where $mindist_k(P(t_i))$ is defined as the minimum distance of a point $p \in P$ from its k^{th} neighbor. Intuitively there are no more than k points within a distance of d_{max} from each other. Another important assumption in this work is the *distribution assumption* which indicates how "nicely" the objects are distributed. *k-spread* $\Delta_k(P)$ of a set of points P is used as a distribution parameter, and it is defined as

$$\Delta_k(P) = diam(P)/mindist_k(P)$$

where $diam(P)$ is the largest distance between any two points in the point set P. This could be used in the cases when a small movement in fewer points may bring large number of changes. The k-spread can be used to bound the number of points within a region if the $diam(P)$ is small.
Roeloffzen *et al.* [1] describe a framework in the black-box model that could be used to maintain a structure on moving objects that are defined only by a subset of the objects, called the *defining sets*. A subset $Q \subseteq P$ is a defining set of the structure $\mathcal{S}(P)$ if $\mathcal{S}(P) = \mathcal{S}(Q)$ and for any strict subset $Q' \subset Q$ gives $\mathcal{S}(P) \neq \mathcal{S}(Q')$. As an example, if P is a set of points and $\mathcal{S}(P)$ is the axis-aligned bounding box $bb(P)$. Then the defining set Q of $\mathcal{S}(P)$ is at most four points on the each edge of the bounding box $bb(P)$. An object is called *active* if its a part of the defining subset Q, otherwise *inactive*. A simple way to compute defining sets is to define a bound on the number of time steps when an object will become a part of the set Q which depends on the structure $\mathcal{S}(P)$. The key feature of this framework is that it ignores the *inactive* objects when updating the structure $\mathcal{S}(P)$. The *inactivity function* is defined to calculate the time stamps when an object is going to be active again. The framework helps in recomputing the structure of interest $\mathcal{S}(Q)$ which leads to a sublinear update algorithms. The thesis describes a general algorithm which is used to maintain Convex Hull, 2-Center, Delaunay Triangulation, and used in collision detection on moving set of objects P. It is important, in the framework, to define the *inactivity function* which guarantees a correct defining set. A brief description on the approach of maintaining a Convex Hull and a Delaunay Triangulation using the presented framework is explained below.

- Convex hull: the defining set for this structure is the set of convex hull vertices. The set is typically much smaller than the original set of points. The author presents an algorithm that runs in $O(k\Delta_k \log^2 n)$ amortized time per time step. Two variants of the algorithm are presented. One uses a floor function to get rid of one *log* factor from the running time, but it needs to know $mindist_k(t)$ at every time step t. The other variant considers that the

point set does not have a bounded spread, and displacement assumption is sufficient for this variant.

- Delaunay Triangulation: In the thesis work, they present two simple algorithms to update a triangulation at every time step by moving points from location $P(t)$ to location $P(t+1)$. One of the algorithms inserts points $P(t+1)$ and deletes $P(t)$. While the other moves the points form $P(t)$ to $P(t+1)$ in a straight line and maintains triangulation by edge flips. The work in this thesis also provides a bound on the number of edge flips and time spent per edge flip. Displacement assumption and k-spread is used just to analyze the running time.

As described this work presents efficient algorithms to maintain a data structure in kinetic setting under strong displacement assumption combined with k-spread. Many practical applications may satisfy the assumption, but in many cases it can fail if the displacement of objects is large. In the case of large displacement, the presented model cannot guarantee correctness.
The existing framework can be modified to deal with large displacement by removing its dependency on the displacement assumption. This can make it more general which could be used in many practical applications very efficiently and with high robustness.

Chapter 3

Range-assignment algorithm

In this chapter, we propose a centralized range-assignment algorithm, referred as *VORONOI_RA*, to maintain a successful broadcasting in mobile ad-hoc network. The exact problem setting is discussed before we dive into the details of the algorithm. The network setting describes the properties and behavior of the network and the sensor nodes. In this setting, the locations of the nodes are available at different times t_i where $i \in \mathbb{N}^+$. A *time interval* is defined to be the time gap between two consecutive times when the locations are available. A time interval is denoted as $[t_i, t_{i+1}]$ where t_i and t_{i+1} are two consecutive times with available locations of the nodes. A broadcast tree $T_B(t_i)$ is calculated using a Minimum Spanning Tree (MST) algorithm at time t_i. The tree $T_B(t_i)$ is used in determining the initial range of the nodes in the network during the time interval $[t_i, t_{i+1}]$. The initial range $r(u, t_i)$ of a node $u \in \mathcal{N}$ is the Euclidean distance $d(u(t_i), w(t_i))$ between the node u and its farthest child node w. The proposed algorithm assigns a range-assignment function r to every node u in the network to determine its range $r(u, t)$ at time t in a given time interval. Every node in the network is assigned a range such that a successful broadcast is maintained over time while keeping an energy efficient network. The energy efficiency is analyzed in terms of following performance measures:

1 *MAX* : the maximum power consumption $\max_{u \in \mathcal{N}} C \cdot r(u, t)^\beta$ in the network at time t.

2 *TOTAL* : the total power consumption $p(\mathcal{N})$ $(:= \sum_{u \in \mathcal{N}} r(u, t)^\beta)$ at time t.

The performance measures are referred as *MAX*, and *TOTAL* in rest of the thesis.

The chapter is divided into two sections. Section 3.1 discusses the existing algorithm which is taken as a reference to devise an efficient and reliable algorithm. Section 3.2 presents the *VORONOI_RA* algorithm to provide an effective solution to our range-assignment problem.

3.1 Background

Wu and Dai [9] discuss an approach to extend the range of the nodes with a *buffer range* to maintain a movement-connected network for successful network-related operation, such as broadcast. A *buffer range* is an additional range added to the initial range of a node at time t_i such that it keeps all of its child nodes within its range in a given time interval $[t_i, t_{i+1}]$. As shown in figure 3.1, the range of a node u is extended with a buffer range to keep the node w in its range over the given time interval. The *buffer range* in this case is simply taken as (*buffer range*=)

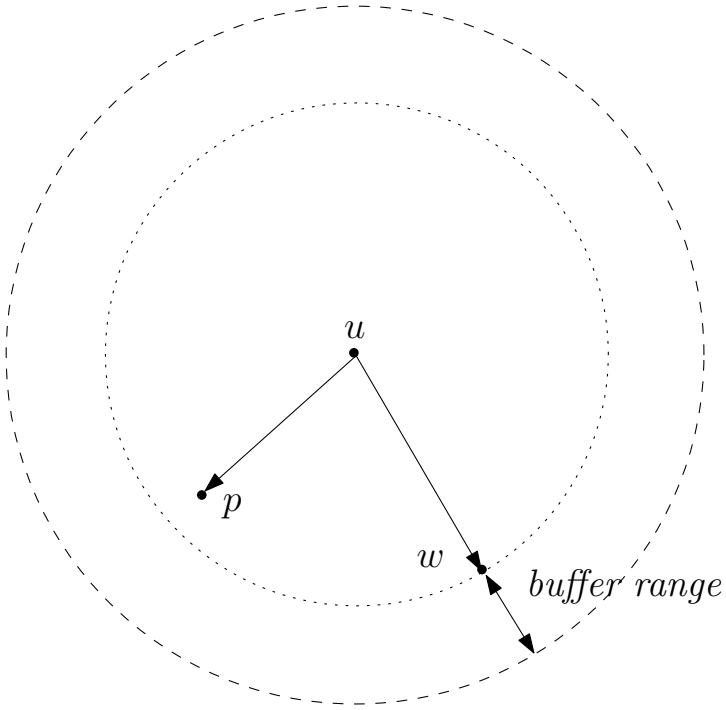


Figure 3.1: shows the buffer range of the node u where the node w is the farthest child node

$2 \cdot V_{max} \cdot (t_{i+1} - t_i)$ which is the maximum increase in Euclidean distance between the node u and its farthest child node w in time interval $[t_i, t_{i+1}]$. The assignment of an extended range allows the node u to keep all of its child nodes in range. The directed edges of the broadcast tree $T_B(t_i)$ are maintained by extending the range of the nodes (with at least one child node). Hence, a broadcast tree is maintained throughout the given time interval, and it will allow successful broadcasting within the network. This algorithm is referred as *DIRECT_RA* in rest of the thesis.

The main drawback of this algorithm is that it directly assigns an extended range to the nodes in the network, whereas the nodes might not move at all or

move a little distance apart. The *DIRECT_RA* algorithm for range-assignment is clearly not optimal, and we could do better: a simple modification to this approach could result in a network with better energy-efficiency, and we discuss this modified algorithm which is also referred as *INTERPOLATED_RA*.

The algorithm *INTERPOLATED_RA* is proposed for the first time in this thesis. The idea is to calculate the range of a node, at time t, by taking into account the maximum possible distance between the node itself and its farthest child node. A broadcast tree $T_B(t_i)$ is created at time t_i when the location of the nodes is available. Consider a node u and its child node w as shown in figure 3.1. The maximum distance $d(u(t), w(t))$ between two nodes u and w is possible when the nodes move in opposite directions. This distance, at time t, is given by the equation below:

$$d(u(t), w(t)) = d(u(t_i), w(t_i)) + 2 \cdot V_{max} \cdot (t - t_i)$$

where V_{max} is the upper bound to the speed of nodes in the network $\mathcal{N}$.

Recall that the node w is the farthest child node of u in broadcast tree $T_B(t_i)$. We claim that the node u can keep all its child nodes in its range iff $r(u,t) = d(u(t), w(t))$ at time t.

To prove this claim, consider another child node p of the node u with $d(u(t_i), p(t_i)) \leq d(u(t_i), w(t_i))$ at time t_i. From the equation of the maximum distance between any two nodes at time t, we can see that $d(u(t), q(t)) \leq d(u(t), w(t))$ always holds in given time interval. This proves that the node u keeps its child nodes in its range iff $r(u,t) = d(u(t), w(t))$. Hence, we set

$$r(u,t) = d(u(t), w(t)) = r(u,t_i) + 2 \cdot V_{max} \cdot (t - t_i)$$

Similarly, other nodes in the network will cover their child nodes if the range is calculated in this manner for every node. The algorithm *INTERPOLATED_RA* uses above range assignment function to assign range to the nodes in the network. Throughout the time interval, the nodes keep their child nodes in their range, and hence, preserving the directed edges of the broadcast tree $T_B(t_i)$. Consequently, the broadcast tree is maintained throughout the time interval $[t_i, t_{i+1}]$ in the communication graph for successful broadcasting in the network.

Modified Algorithm

The algorithm *INTERPOLATED_RA* creates a minimum spanning tree $T_B(t_i)$ in step 1. The tree supports successful broadcasting of a message from the source

node at time t_i. The algorithm calculates a range function in step 5 for every non-leaf node u, such that the edges of the tree $T_B(t_i)$ is sustained by extending the range of the nodes u over the time interval. As the algorithm maintains the edges

Algorithm 1 *INTERPOLATED_RA*($\mathcal{N}$, V_{max}, $[t_i, t_{i+1}]$)

1: $T_B(t_i) \leftarrow MST(\mathcal{N})$
2: **for** $u \in T_B(t_i)$ **do**
3: **if** u is not a leaf node in $T_B(t_i)$ **then**
4: Let w be the farthest child node of u at time t_i
5: $r(u,t) \leftarrow d(u(t_i), v(t_i)) + 2 \cdot V \cdot (t - t_i)$
6: $p(u,t) \leftarrow C \cdot r(u,t)^{\beta}$
7: **end if**
8: **end for**

that were present in $T_B(t_i)$ at any time $t \in [t_i, t_{i+1}]$, the broadcast tree is preserved in the communication graph over time. Hence, the resultant topology allows a successful broadcasting over the given time interval. Figure 3.2 shows the increase

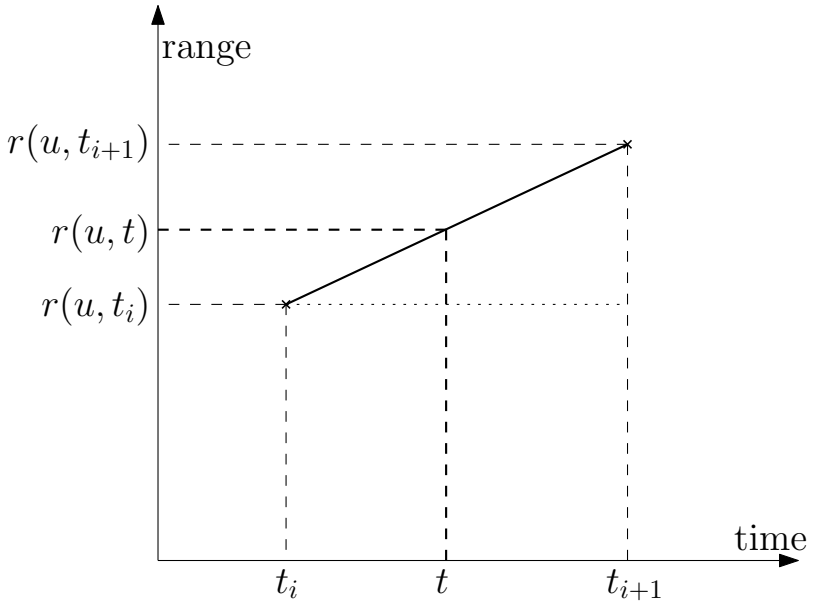


Figure 3.2: shows example of range-assignment function to assign range to a node u.

in the range of a node u in the network. The increase in range $r(u,t)$ with time t is linear as the range of a node u is directly proportional to time t.

3.2 Proposed Algorithm

In this section, the *VORONOI_RA* algorithm is proposed. There are few terms used in the algorithm which are discussed before presenting the algorithm in section 3.2.3. We discuss the movement model in section 3.2.1, which describes the movement of nodes in a network. We also discuss the concepts of *ordering* and *responsible nodes* which are used in the algorithm *VORONOI_RA*.

3.2.1 Movement Model

A movement model describes the movement of the nodes in a network. It takes into consideration the current velocity of the nodes and decides the possible next positions in the 2D plane according to time that passed. In the experimentation, the movement of the nodes depends on two factors. These factors play an important role to mimic the movement pattern of sensor nodes in a network used in a practical application. For example, vehicles in a Vehicular Mobile Network mostly move in a straight or a slightly curved path. Thus, the randomness in their motion is very limited. While in an example of a mobile ad-hoc network in a military application, the nodes (transmitters) can have high randomness in their motion because the person carrying the device can turn in any random direction. These movement patterns are addressed with the two factors of movement model. These factors are:

- *direction factor* α : it decides the deviation of a node's path in time interval $[t_i, t_{i+1}]$ from the direction of previous path in interval $[t_{i-1}, t_i]$. The node can deviate from its previous direction by an angle of $\alpha/2$ on either side as shown in the figure 3.3. The figure shows that the node u can be present anywhere in the region bounded by the dashed arc centered at location $u(t_i)$ within time interval $[t_i, t_{i+1}]$. At time t_{i+1}, the location of the node u will be available, and it will lie inside the region shown in the figure 3.3. At some time $t \in (t_i, t_{i+1})$, the next location could be calculated uniformly, or by using a Gaussian distribution such that there is higher probability to remain close to the previous direction.

- *speed factor* ϵ : it decides the change in speed of the nodes in the network. The new speed $V(u, t_i)$ at time t_i of a node u in the network will lie in the range $[\max(V(u, t_{i-1}) - \epsilon, 0), \min(V(u, t_{i-1}) + \epsilon, V_{max})]$. The speed of the nodes is assumed to be constant throughout the time interval until next time interval starts.

These factors may affect the performance of the algorithm in keeping an energy efficient mobile ad-hoc network. The results from the experimentation will be analyzed to examine the effectiveness of the algorithm with respect to above factors.

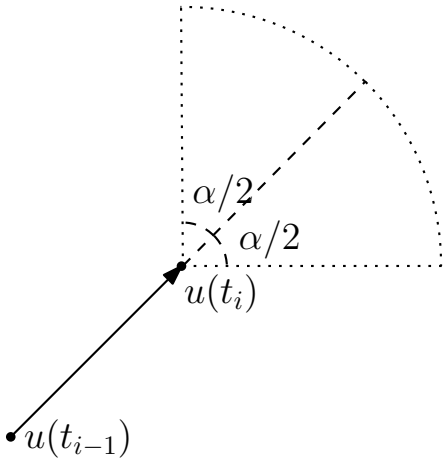


Figure 3.3: shows the *direction factor* α, and the allowed region of movement in current interval $[t_i, t_{i+1}]$.

3.2.2 Ordering and Responsible nodes

An ordering of the nodes is an assignment of an integer to each node $u \in \mathcal{N}$, which we denote by $order(u)$. The nodes are processed in the provided ordering. With processing a node w (except the source node), we mean to assign a range-assignment function to the responsible-nodes in set $\mathcal{R}_w$ of the node u, such that the node u is always in the range of one of the nodes in the network. The source node is assigned an order 0 as it will be the initiator of the broadcast operation and requires no other node to keep the source node in its range. The ordering mainly helps in preventing a node to cover another node already covering it; in other words, it prevents cycles in the broadcast tree. The presented algorithm *VORONOI_RA* makes use of responsible-nodes when processing a node. The set $\mathcal{R}_w$ is the set of responsible-nodes for a node $w \in \mathcal{N}$, so the nodes in the set $\mathcal{R}_w$ are responsible to cover the node w at any time in the interval $[t_i, t_{i+1}]$. A node u covering another node w means that the node w lies within the transmission range of the node u. The performance of the algorithm concerning energy efficiency may change with different heuristics of choosing a set of responsible nodes.

In this thesis, we consider the *BFS+distance* ordering. According to this ordering, for each pair of nodes $u, w \in \mathcal{N}$, $order(u) < order(w)$ iff

- $depth(u) < depth(w)$ in the broadcast tree $T_B(t_i)$, or
- the distance $d(u(t_i), parent[u](t_i)) \leq d(w(t_i), parent[w](t_i))$ and the nodes u, w are present on same level in the tree $T_B(t_i)$, where $parent[u]$ is the parent node of u in the tree $T_B(t_i)$.

We can also think of another ordering technique which is *distance from the source node, i.e.* for each pair of nodes u, $w \in \mathcal{N}$, $order(u) < order(w)$ iff $d(s(t_i), u(t_i)) \leq d(s(t_i), w(t_i))$ where s is the source node. In the current implementation, this ordering is not used, but it would be interesting in future to also analyze the algorithm under this ordering.

We suggest two different heuristics that are used to decide a set $\mathcal{R}_w$ of responsible nodes for a node w in the network:

a. *LOWER_ORDER* : the set $\mathcal{R}_w \subset \mathcal{N}$ of responsible nodes consists of all nodes u such that $order(u) < order(w)$, for each $u \in \mathcal{R}_w$.

b. *2_NEAREST* : the set $\mathcal{R}_w \subset \mathcal{N}$ consists of the two nearest neighbor nodes to the node w such that $order(u) < order(w)$ where $u \in \mathcal{R}_w$.

Section 3.2.3 proposes the *VORONOI_RA* algorithm to maintain an energy efficient mobile ad-hoc network while maintaining the broadcasting. There are two variants of the algorithm based to the heuristics of choosing a set of responsible nodes for a node in the network. One of the variants is referred as *LOWER_ORDER_RA* if heuristic *LOWER_ORDER* is considered from above list. The other variant is called *2_NEAREST_RA* if the heuristic *2_NEAREST* is used.

3.2.3 *VORONOI_RA* Algorithm

The algorithm *INTERPOLATED_RA* assigns a range function $r(u, t)$ for each node $u \in \mathcal{N}$ to extend its range to keep its farthest node within range. We want to propose another algorithm that could maintain the broadcast tree but the parent node is not "*fixed*". With "*fixed*", we mean that a node could be covered by different nodes during the time interval $[t_i, t_{i+1}]$ for the purpose of further optimizing the energy efficiency. Several nodes can be responsible to take over a node w depending on their closeness to the node. Figure 3.4 shows a situation where the node q might extend its range to keep the node w in its range. These nodes are included in the set of responsible-nodes $\mathcal{R}_w$ as discussed in section 3.2.2. The nodes in set $\mathcal{R}_w$ are responsible to keep the node w in their range if the node w enters their respective Voronoi cell, as explained next.

The algorithm presented in this chapter is referred as *VORONOI_RA* algorithm. The main idea behind this algorithm is to use a geometrical data structure, Voronoi diagram. The Voronoi diagram is used to determine the range of the nodes in set $\mathcal{R}_w$ to cover the node w throughout a given time interval. To this end, we construct the Voronoi diagram of the nodes in $\mathcal{R}_w$. Let $\mathtt{R}_u$ denote the Voronoi cell of a node $u \in \mathcal{R}_w$. Then we make the node u responsible for covering w when $w \in \mathtt{R}_u$. At time t_i, the node w can either exist inside or outside of the Voronoi cell $\mathtt{R}_u$ of the

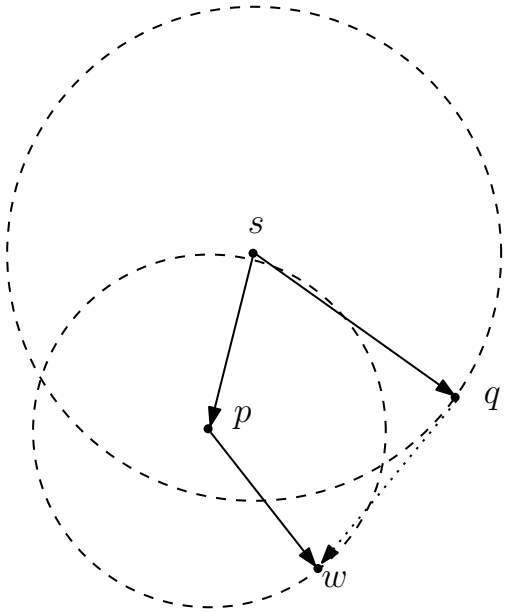


Figure 3.4: shows that the node q might take over the node w by extending its range

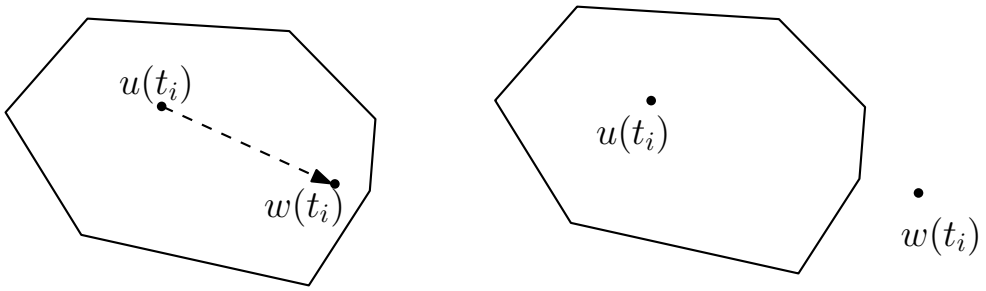


Figure 3.5: shows the voronoi cell $\mathtt{R}_u$ of a node u and two possible locations of the node w at time t_i.

node u. Figure 3.5 shows a node u with its Voronoi cell $\mathtt{R}_u$. The node u gets a initial range when the node w lies in its Voronoi cell at time t_i as shown in the *left* figure 3.5.

The movement model, mainly the *direction factor*, plays an important role in deciding the range of a node over time. Suppose that the direction factor is $\alpha = 360^o$, which means that the nodes are having maximum randomness in their movement. At time $t \in [t_i, t_{i+1}]$, a node w can have a displacement of $disp_w(t) = V_w \cdot (t - t_i)$ units. The location $w(t)$ will be located within a circle $\mathtt{C}_w(t)$ of radius $disp_w$ and center $w(t_i)$. It is also shown in figure 3.6; the circles labeled with $C_w(t^{'})$, $C_w(t^{'})$, and $C_w(t^{'''})$ denote the location of the node w at different times $t^{'}$, $t^{'}$, and $t^{'''}$ respec-

tively.

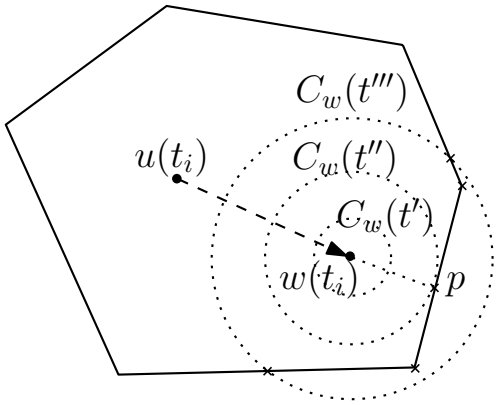


Figure 3.6: shows the circular region $\mathtt{C}_w(t)$ of a node w at different times $t^{'}$, $t^{''}$, and $t^{'''}$, such that $t_i < t^{'} < t^{''} < t^{'''} \leq t_{i+1}$.

In order to cover all the possible locations of the node $w(t)$ at time t, a node $u \in \mathcal{R}_w$ must cover the farthest point in the region $\mathtt{C}_w(t) \cap \mathtt{R}_u$. As the node u is also moving at a speed of V_u, hence, the range of the node u at time t should be

$$r(u,t) = \max_{z \in \mathtt{C}_w(t) \cap \mathtt{R}_u,\, p \in \mathtt{C}_u(t)} d(z,p)$$

We claim that the farthest point z from the location $u(t_i)$ in the region $\mathtt{C}_w(t) \cap \mathtt{R}_u$ is

(i) the farther tangent point p on the circle $\mathtt{C}_w(t)$ which lies on the line joining $w(t_i)$ and $u(t_i)$ as long as the point p lies inside the Voronoi region of the node $u(t_i)$. The circles labeled $C_w(t^{'})$ and $C_w(t^{''})$ in the left diagram in figure 3.6 have the farthest points p inside the Voronoi cell $\mathtt{R}_u$, or

(ii) one of the vertices of the Voronoi cell lying inside the circle $\mathtt{C}_w(t)$, or the intersection points of the circle $\mathtt{C}_w(t)$ and Voronoi cell $\mathtt{R}_u$. These points will lie on the boundary of the Voronoi cell $\mathtt{R}_u$. The circle labeled $C_w(t^{'''})$, for example in figure 3.6 produces these intersection points on the boundary of the Voronoi cell $\mathtt{R}_u$.

The line joining $w(t_i)$ and $u(t_i)$ gives two tangent points on the circle, and one of them is farther than the other from the node location $u(t_i)$. It is trivial to check that the point p is also the farthest point in the region $\mathtt{C}_w(t) \cap \mathtt{R}_u$ from the node u as long as it remains inside the cell $\mathtt{R}_u$. This proves case *(i)* of our above claim.

To prove that the case *(ii)* is also holds, let's consider $C_w(t)$ and the boundary of the Voronoi cell $\mathtt{R}_u$ lying inside $C_w(t)$ in figure 3.7. We prove that there is no

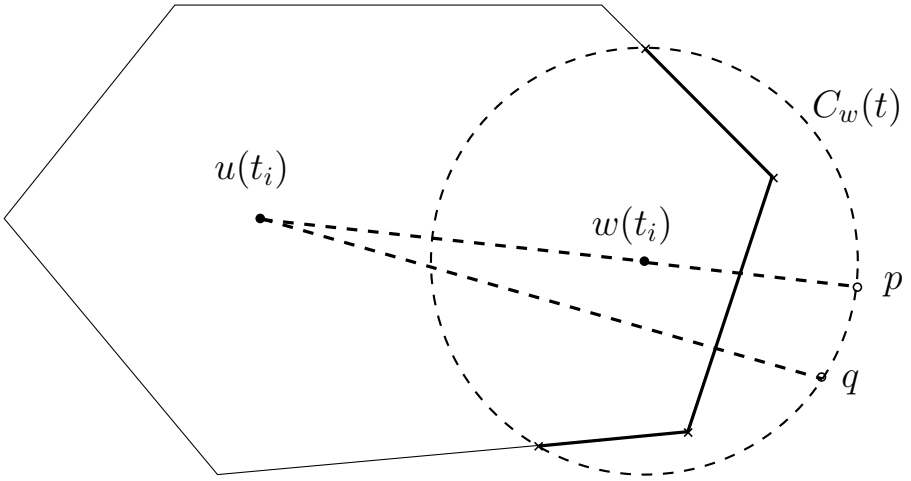


Figure 3.7: the distance between the node u and a point q on circle boundary decreases as q moves away from the point p on the circle boundary.

point on the portion of circle $\mathtt{C}_w(t)$ inside Voronoi cell $\mathtt{R}_u$ that could be the farthest point from the node u. This will prove that one of the points (crossed points in figure 3.7) on the Voronoi $\mathtt{R}_u$ boundary is the farthest point from the node u at time t. Consider the tangent point p as shown in the figure 3.7. The distance between the node $u(t_i)$ and a point q decreases as the point q moves away from the point p on the boundary of the circle $\mathtt{C}_w(t)$. Thus, we can say that any point q on the arc inside the Voronoi region is closer to the node u as compared to the crossed point shown in the figure 3.7. And, one of the crossed points on the boundary of Voronoi cell is at a maximum distance from the node u because the Voronoi edges are line segments. The node u is always at the maximum distance from one of the endpoints (crossed points in the figure) of a line segment. This proves that one of the vertices (inside $C_w(t)$) of Voronoi cell, or the intersection points of $C_w(t)$ and $\mathtt{R}_u$ must be the farthest possible location of the node w at time t.

Hence, there are finite number of points to be checked to find the farthest point z in region $\mathtt{C}_w(t) \cap \mathtt{R}_u$ from the node location $u(t_i)$. Once the point z is found, the range of the node u to cover a node w will be

$$r(u,t) = \max_{p \in \mathtt{C}_u(t)} d(z,p)$$

It is easy to find the point p. It is simply the point in the direction of $u(t_i)$ from z, at a euclidean distance of $V_u \cdot (t - t_i)$ from the node $u(t_i)$.

There is one final step that should be taken care of. Each node w except the source node, assigns a range $r_w(u,t)$ to a node u in its set $\mathcal{R}_w$ of responsible-nodes.

Thus, a node u may be assigned different range values by different nodes to keep them in range for a successful broadcast. Let $\mathcal{R}_u^*$ be the set of nodes for which node u was responsible for keeping them in its range at some time t. The final range $r(u,t)$ of the node u is

$$r(u,t) = \max_{w \in \mathcal{R}_u^*} r_w(u,t)$$

Until now, we have described an approach to find the range $r(u,t)$ of a node u such that it keeps every node $w \in \mathcal{R}_u^*$ in its range. Now this approach is used to assign range to any node $u \in \mathcal{N}$ for a successful broadcast operation. Algorithm *VORONOI_RA* summarizes our approach for range assignment to the nodes in the network $\mathcal{N}$ at time $t \in [t_i, t_{i+1}]$. Step 1, the algorithm creates a minimum spanning

Algorithm 2 *VORONOI_RA*($\mathcal{N}$, V, $[t_i, t_{i+1}]$)

1: $T_B(t_i) \leftarrow MST(\mathcal{N})$
2: provide *BFS+distance* ordering to the nodes
3: **for** $w \in T_B(t_i)$ in increasing order **do**
4: $\quad \mathcal{R}_w \leftarrow$ set of responsible-nodes to cover the node w
5: $\quad$ calculate Voronoi diagram of the nodes in the set $\mathcal{R}_w$
6: $\quad$ **for** $u \in \mathcal{R}_w$ **do**
7: $\quad\quad r_w(u,t) = \max_{z \in \mathtt{C}_w(t) \cap \mathtt{R}_u,\, p \in \mathtt{C}_u(t)} d(z,p)$
8: $\quad$ **end for**
9: **end for**
10: **for** $u \in \mathcal{N}$ **do**
11: $\quad r(u,t) = \max_{w \in \mathcal{R}_u^*} r_w(u,t)$
12: **end for**

tree $T_B(t_i)$ of the nodes in the network. The complexity of creating a minimum spanning tree is $O(n^2)$ if we first create a complete graph with $O(n^2)$ edges, and then run an algorithm like Prim's or Krushkal's for creating the minimum spanning tree (MST). There is an $O(n \log n)$ algorithm to achieve the same by creating a Delaunay triangulation of the nodes first. A MST of nodes is always a sub-graph of the Delaunay triangulation of the nodes. The time complexity of triangulation is $O(n \log n)$, and it gives a graph containing $O(n)$ edges. Thus, an algorithm like Krushkal's or Prim's will take $O(n)$ time to give the MST $T_B(t_i)$ of the nodes in the network $\mathcal{N}$ at time t_i.

The nodes are provided with ordering in step 2, which decides the order in which nodes are processed. Step 3-9, the algorithm iterates for each node w to find its set $\mathcal{R}_w$ of responsible nodes, and assigns a range to each node u in the set $\mathcal{R}_w$

such that the node u keeps the node w in its range at time t. The range assigned to cover the node w is simply the maximum possible distance between $u(t)$ and the farthest point z in the region $\mathtt{C}_w(t) \cap \mathtt{R}_u$. It is already explained before to find such point z. Step 10-12, the algorithm finds the final range $r(u,t)$ of the node u.

A node u is considered to be present in the set $\mathcal{R}_w$, if $order(u) < order(w)$. Thus, every node w, except the source node, must have a directed edge from one of the nodes with a lower order in a given time interval $[t_i, t_{i+1}]$. It also means that we have a directed path from the source to every node in the set $\mathcal{N}$ of the sensor nodes. Hence, a broadcast tree is always maintained in the communication graph of the network consisting of mobile sensor nodes. This proves that the algorithm *VORONOI_RA* always keeps a successful broadcasting over time.

Effect of direction factor α

The range assignment is discussed for $\alpha = 360^o$. For a value of *direction-factor* $\alpha < 360^o$, there are more cases to be considered when finding the farthest point z in the region $\mathtt{C}_w(t) \cap \mathtt{R}_u$, where $\mathtt{C}_w(t)$ is not a circle anymore but a reachability region as shown in figure 3.3. There are different possibilities of the farthest point z in this case of $\mathtt{C}_w(t)$. The point z could either lie on $\mathtt{C}_w(t)$, or it is the center of the arc. Figure 3.8 shows all of the different cases. The cases *1*, *2*, and *3* in figure 3.8 shows that the farthest point lies on the arc, *i.e.* a, b, or p, while, case *4* shows that the center $w(t_i)$ could be the farthest point.

If the point z lies on the $\mathtt{C}_w(t)$, then it would be one of the endpoints (a and b) of the arc, or the tangent point p. The tangent point is always the point at maximum distance from a node u if it lies on the arc. Otherwise, one of the endpoints is the farthest because one of them (either a or b) is the closest point to p on the arc boundary. We already discussed that the distance of a points q from another point decreases as the point q goes away from the tangent point p on the circular boundary. We also need to consider the intersection of Voronoi cell $\mathtt{R}_u$ with the arc $\mathtt{C}_u(t)$. It is similar to the case we discussed when $\alpha = 360^o$. Again, with finite number of checks, the point z can be found in the region $\mathtt{C}_w(t) \cap \mathtt{R}_u$. The range of the node u is

$$r(u,t) = \max_{p \in \mathtt{C}_u(t)} d(z,p)$$

where $\mathtt{C}_u(t)$ is the reachability region of the movement of the node u.

Variants of *VORONOI_RA* algorithm

Recall that our algorithm uses the concept of responsible nodes. The set $\mathcal{R}_w$ of responsible nodes for a node w contains the nodes that are responsible for keeping

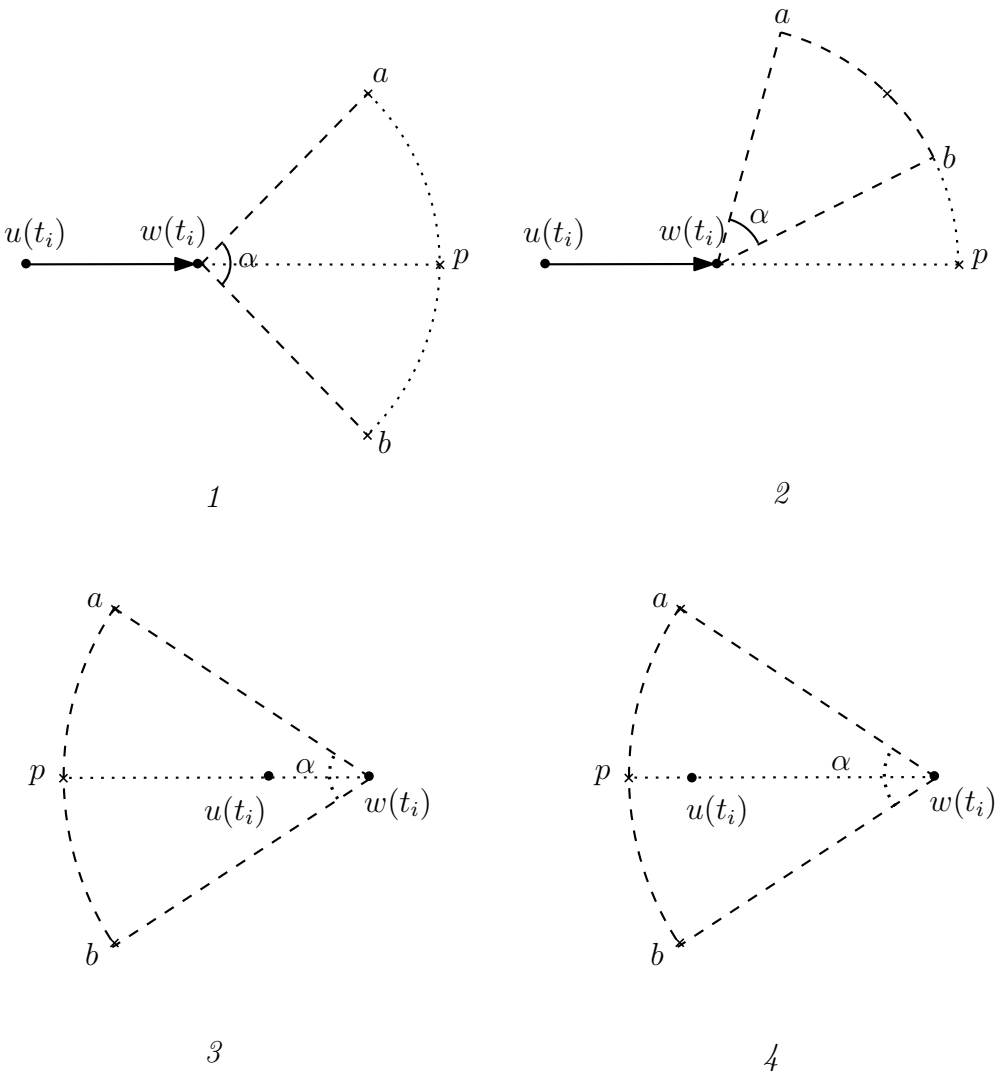


Figure 3.8: how direction factor affects the farthest point to the responsible-node u of a node w.

it in their range at time t in given interval. In this thesis, we mainly consider two different heuristics for choosing the responsible nodes as discussed in section 3.2.2. We propose following two variants of the *VORONOI_RA* algorithm.

- *LOWER_ORDER_RA* : this variant uses the heuristic *LOWER_ORDER* for finding the set of responsible nodes for a node w in step 4 of the algorithm.
- *2_NEAREST_RA* : this variant uses the heuristic *2_NEAREST* for finding the set of responsible nodes for a node w in step 4 of the algorithm.

These variants are implemented, and the results are compared with the results of

the algorithms *DIRECT_RA* and *INTERPOLATED_RA* in results section 4.2 of the next chapter.

Chapter 4

Experimental Evaluation

This chapter discusses the experimental setup that is developed to test our algorithms performance on different networks. Section 4.1 of this chapter discusses the implementation of the experimental setup, which is developed to test and compare the performance of the algorithms on different mobile ad-hoc networks. Furthermore, section 4.2 includes the results of the algorithms. The graphs of different performance measures are included in the results section, and observed patterns are explained.

4.1 Experimental Setup

The previous chapter discussed different algorithms to maintain a broadcast tree in a mobile ad-hoc network. Below, we analyze the performance of the algorithm in terms of the proposed performance measures: MAX, AVG, and TOTAL. An experimental setup is developed in C++ and Qt to analyze the performance concerning energy efficiency in the network. The experimental setup consists of different components to take the required inputs, execute the algorithms, and display the results on a graphical user interface (GUI). The following list explains the components included in the setup:

- InputComponent : this component is used to get the inputs that describe the network properties. For example: the number of nodes, parameters for movement model, length of the time intervals, etc.
- RenderComponent : this component is used to paint nodes and its details on the GUI to analyze the results of the algorithms by observing the changing state of the network over time.
- SimulationComponent : this component simulates nodes movement. It is responsible to run the algorithms, assign ranges to the nodes, and send real-time data to the GraphComponent. Each algorithm is run on the input data

from InputComponent, and the range-assignment function is computer for each node in the network.

- GraphWindow : it consists of three GraphComponent which are plotted with the data received from the SimulationComponent. The data received is maximum, average, and total energy consumption over time.
- GraphComponent : this component displays graph of performance measure over time. As mentioned earlier, there are three performance measures, and a graph is plotted for each of the measures with time:
  - MAX : maximum energy consumption at time t, *i.e.* $C \times \max_{u \in \mathcal{N}} r(u,t)^{\beta}$
  - TOTAL : total power consumption at time t, *i.e.* $p(\mathcal{N})$ $(:= \sum_{u \in \mathcal{N}} r(u,t)^{\beta})$

Many parameters characterize the properties and the behavior of the nodes in a mobile ad-hoc network. The parameters are received as an input from the InputComponent described above. Following is the list of the parameters that we use to characterize a network:

1 Number of nodes, #*nodes*, in the network.

2 Length of a time interval : a time interval is $[t_i, t_{i+1}]$, where t_i and t_{i+1} are two consecutive times when the location of the nodes is available. The length of a time interval is the time duration between t_i and t_{i+1}. We assume that each time interval is of equal length in our implementation of the experimental setup.

3 Maximum allowed speed V_{max}: it is the upper bound on the maximum allowed speed of the sensor nodes in the network.

4 Direction factor α: the value of direction factor lies in $[0^o, 360^o]$. An increased value of direction factor increases the randomness in the mobility of the nodes.

5 Speed factor ϵ: it is another parameter that describes the movement model of the nodes in the network. This factor is used to estimate the speed of the nodes in current time interval. The maximum allowed speed V_{max} is an upper bound to the speed.

Once the network parameters are provided to the experimental setup, the nodes are generated with random locations at time t_0. Later on, the location depends on the movement model parameters (α and ϵ) at time t_i for $i > 0$. Above inputs are provided to the algorithms to assign range to the nodes for successful broadcasting. The output of the algorithms is analyzed for different networks. The networks, in the experimentation, differ according to the values of above input parameters. We

want to analyze the efficiency of the algorithms on various sensor networks. The algorithms are run multiple times on a network with different sets of randomly generated location of the nodes. The results are observed carefully to discuss the energy efficiency of the networks under different algorithms discussed in chapter 3.

4.2 Results and Analysis

The algorithm from the research paper by Wu and Dai [9] is taken as a starting point of our experimental analysis. The algorithm is referred as *DIRECT_RA*. As the algorithm is inefficient, we proposed a modification to the algorithm; and the modified algorithm is called *INTERPOLATED_RA*. The algorithm assigns a range function to each node u which decides the range $r(u,t)$ of the node at time t in a given interval. The range function is

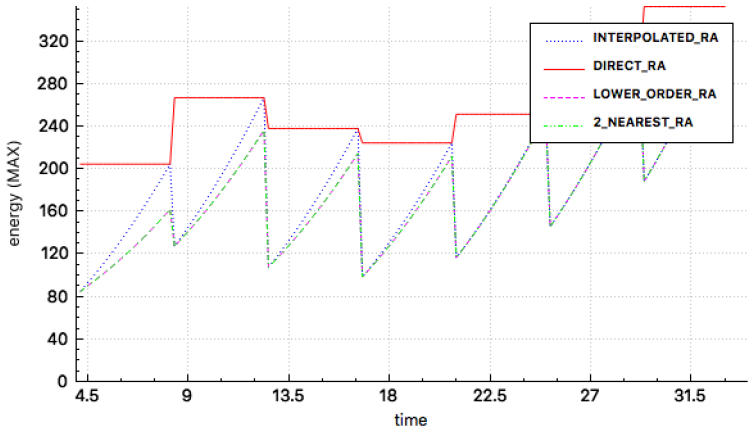
$$r(u,t) = r(u,t_i) + 2 \cdot V_{max} \cdot (t - t_i)$$

which interpolates between the initial range $r(u,t_i)$ and $r(u,t_{i+1})$. Recall that, at any time t, the range $r(u,t)$ is the maximum possible euclidean distance between the node u and its farthest child node in the broadcast tree $T_B(t_i)$.

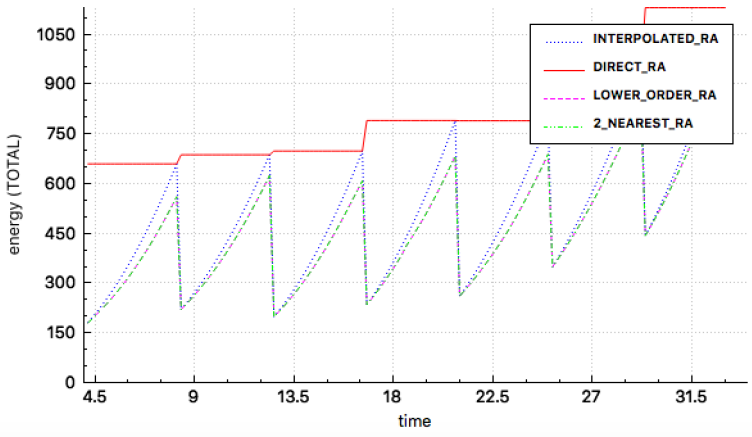
Our proposed algorithm, *VORONOI_RA*, uses Voronoi diagrams to maintain broadcasting in a mobile sensor network. The algorithm uses the concepts of movement model, ordering, and responsible nodes as explained in chapter 3.

The results are obtained as graphs between each of the performance measures and time. The interval length is taken as 4 seconds in the experimentation. Thus, the location of the nodes is available at time $t = 0, 4, 8, \ldots$ and so on. Other parameters - #nodes, α, and maximum speed V_{max} - are changed to test the algorithms on different networks.

Figure 4.1 shows the graphs for a network with 10 sensor nodes. The direction factor α is kept low as 20^o, and maximum allowed speed is 20 units. The experimentation is performed on a 1200×1200 2D plane. It can be seen from the graphs that the variants of *VORONOI_RA* algorithm (*LOWER_ORDER_RA*, *2_NEAREST_RA*) are more energy efficient than other algorithms (*DIRECT_RA*, *INTERPOLATED_RA*). The *VORONOI_RA* algorithm performs better in terms of all three performance measures (*MAX*, *TOTAL*). At the start t_i of every interval, the performance measures have equal values for the algorithms, except the *DIRECT_RA* algorithm. This behavior is seen due to assignment of range as the distance between a node and its farthest child node at time t_i. Later, between the times t_i and t_{i+1}, the energy consumption decreases in case of the variants

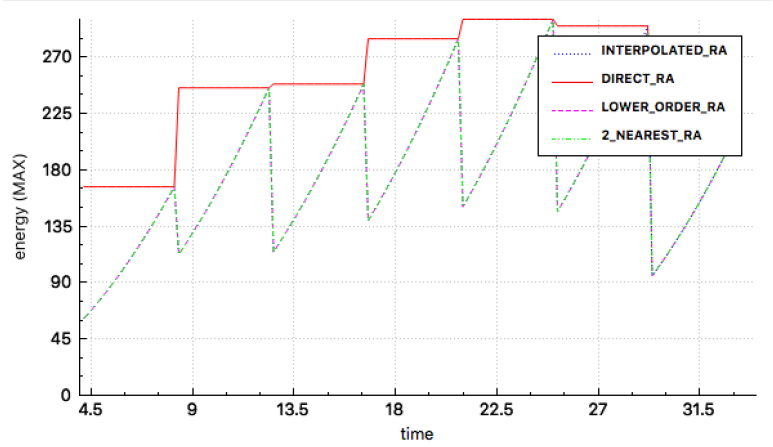


(a) maximum energy consumption over time

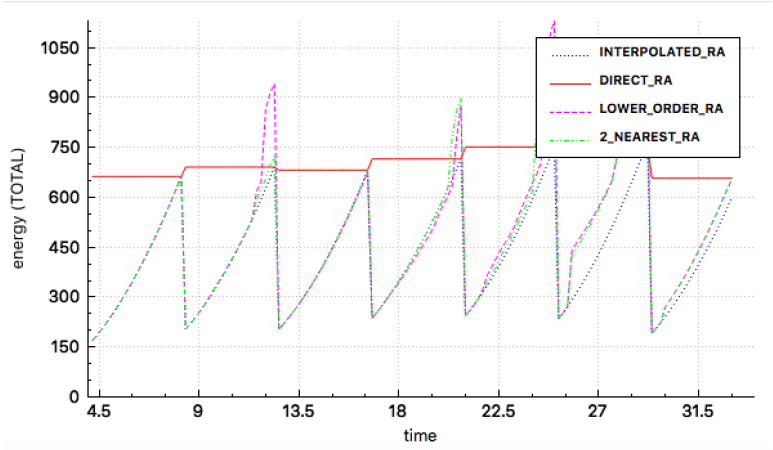


(b) total energy consumption over time

Figure 4.1: Above graphs show the values of performance measures over time t for the network with following network setting: $\#nodes = 10$, interval length is 4 seconds, $\alpha = 20^o$, and $V_{max} = 20$.



(a) maximum energy consumption over time

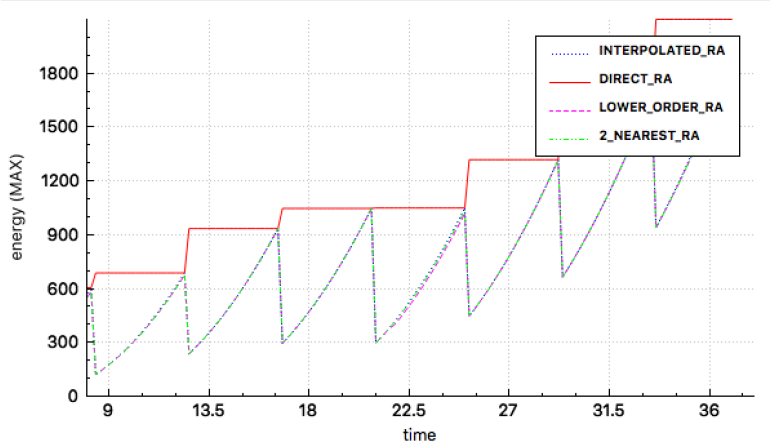


(b) total energy consumption over time

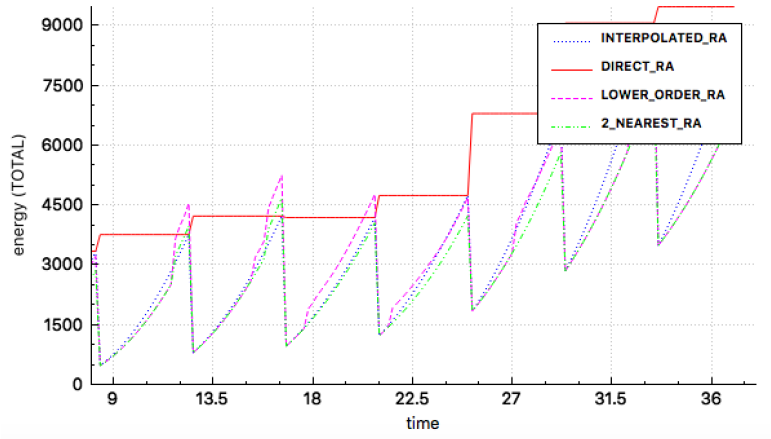
Figure 4.2: Above graphs show the values of performance measures over time t for the network with following network setting: $\#nodes = 10$, interval length is 4 seconds, $\alpha = 300^o$, and $V_{max} = 20$.

LOWER_ORDER_RA, *2_NEAREST_RA* of the proposed algorithm. Under *INTERPOLATED_RA* algorithm, a node keeps extending its range to keep its farthest node in its range. While in the variants of *VORONOI_RA*, a node can only extend its range up to the farthest vertex of its Voronoi cell.

The energy consumption patterns are shown in the graphs of figure 4.2 for another network containing 10 sensor nodes. The direction factor α is increased to 300^o, and it increases the randomness in the movement of the nodes in the network. The increase in randomness in the movement reduces the performance of the algorithm *LOWER_ORDER_RA* and *2_NEAREST_RA*, which can be confirmed by the sudden increase in the TOTAL energy consumption graph. The MAX energy



(a) Graph between maximum energy consumption and time



(b) Graph between total energy consumption and time

Figure 4.3: Above graphs show the values of performance measures with time t for the network with following network setting: $\#nodes = 10$, interval length is 4 seconds, $\alpha = 20^o$, and $V_{max} = 60$.

consumption remains nearly the same as in the *INTERPOLATED_RA* algorithm, because the maximum range of a node cannot exceed the distance of the node from the farthest vertex v of its Voronoi cell. In the algorithm, *INTERPOLATED_RA*, a node u keeps extending its range to cover its farthest child node. This also explains the lower values of MAX energy consumption in graph 4.1a under the variants of the *VORONOI_RA* algorithm.

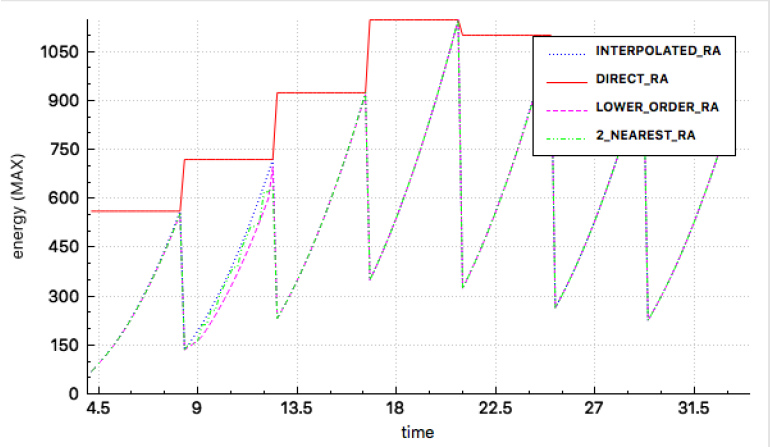
We observe a sudden increase in the TOTAL energy consumption in the graph 4.2b. The increase is mostly towards the end of any interval. This is due to multiple responsible nodes trying to take over a node by extending their range. Due to a large increase in randomness ($\alpha = 300$) in the movement, it becomes more

likely that a node may fall in Voronoi cells of different responsible nodes. Also, the variant *2_NEAREST_RA* has less increase than the variant *LOWER_ORDER_RA* because only two responsible nodes are involved in *2_NEAREST_RA*.

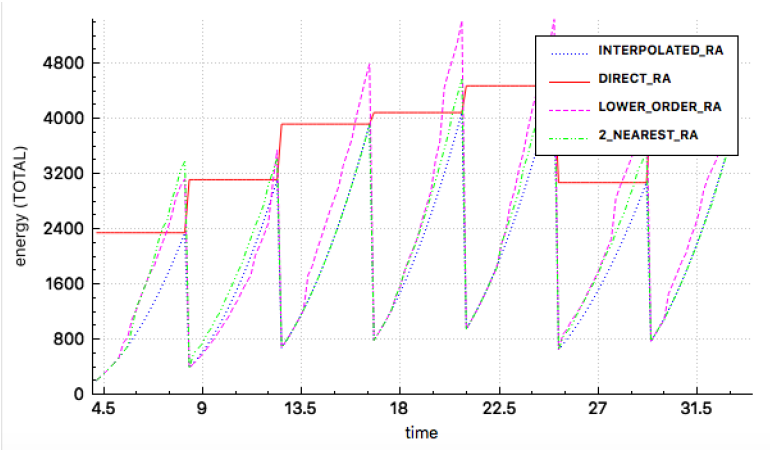
Figure 4.3 shows the graphs obtained when the algorithms are run on a network with *10* nodes, $\alpha = 20$, and the maximum allowed speed is increased to *60*. As seen in the graphs 4.3a, 4.3b, the variants (*LOWER_ORDER_RA*, *2_NEAREST_RA*) perform better than existing algorithms in terms of MAX performance measure. It can be reasoned in a similar way as done for the graphs in figure 4.1. Due to increased speed, a node have higher chance of quickly moving into one of the multiple Voronoi cells. Thus, the performance decreases in terms on *TOTAL* energy consumption. This is the reason we see peaks in the graph 4.4b of *TOTAL* energy consumption.

Figure 4.4 shows results for a network with both increased maximum speed ($V_{max} = 60$) and the direction factor $\alpha = 300^o$. The patterns seen in the graphs, *MAX* and *TOTAL*, are similar to the ones for the network with increased direction factor $\alpha = 300^o$ and low speed ($V_{max} = 20$). The AVG and TOTAL energy consumption for the variants of *VORONOI_RA* algorithm are higher than the energy consumption under the algorithm *INTERPOLATED_RA*. An increase in energy consumption is seen at the end of the intervals $[t_i.t_{i+1}]$ due to multiple nodes trying to keep a node in their range by extending their ranges.

The *MAX*, *TOTAL* graphs are obtained for different mobile ad-hoc networks. Each network differs from one another according to the values of network setting parameters discussed in the previous section 4.1. For each network, the graphs are generated many times on different sets $\mathcal{N}$ of nodes with randomly generated initial location. Similar patterns are seen in the graphs for each randomly generated set $\mathcal{N}$ of the network. Consider a network with following network settings: Number of nodes is 10, interval length is 4 seconds, direction factor α is 20^o, and maximum allowed speed in 20. This is considered as one network, and the algorithms are run at least 10 times on different sets $\mathcal{N}$ of randomly generated nodes (*i.e.* the initial locations of the nodes are different in every execution of the algorithms for this network). We have included the graphs for a single run of the algorithms for a particular network. Furthermore, the values of network parameters are changed, and the graphs are obtained for a network with different network properties. The algorithms are run many times for each network. The graphs are included and analyzed for one of the several executions of the algorithms for every network. The graphs that are not included in the thesis have shown similar patterns which are explained with the help of the results included in this thesis.



(a) Graph between maximum energy consumption and time



(b) Graph between total energy consumption and time

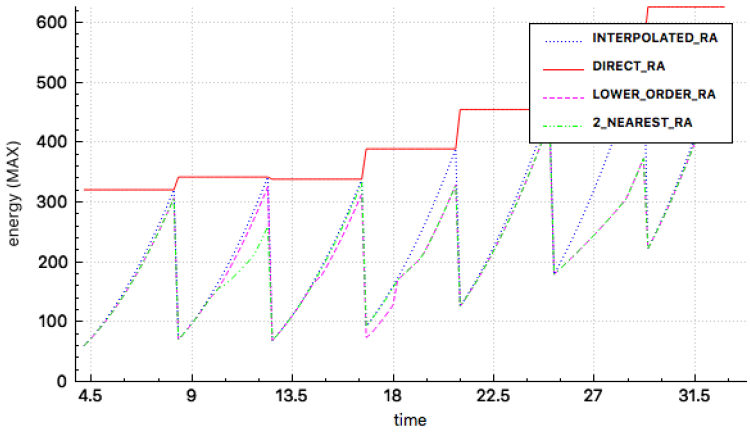
Figure 4.4: Above graphs show the values of performance measures with time t for the network with following network setting: $\#nodes = 10$, interval length is 4 seconds, $\alpha = 300^o$, and $V_{max} = 60$.

To summarize, the results of the variants of the *VORONOI_RA* algorithm are mainly affected by the direction factor α. It is seen that a network with a lower value of α is energy efficient compared to the algorithms, *DIRECT_RA* and *INTERPOLATED_RA*. The existing algorithms try to extend a node's range to the maximum possible distance between the node itself and its farthest child node; while in the variants of the *VORONOI_RA* algorithm, the range of a node is extended to the farthest point z in the region $\mathtt{C}_w(t) \cap \mathtt{R}_u$. And the distance of a node to the point z is always lower or equal to the distance from its farthest child node at time t.

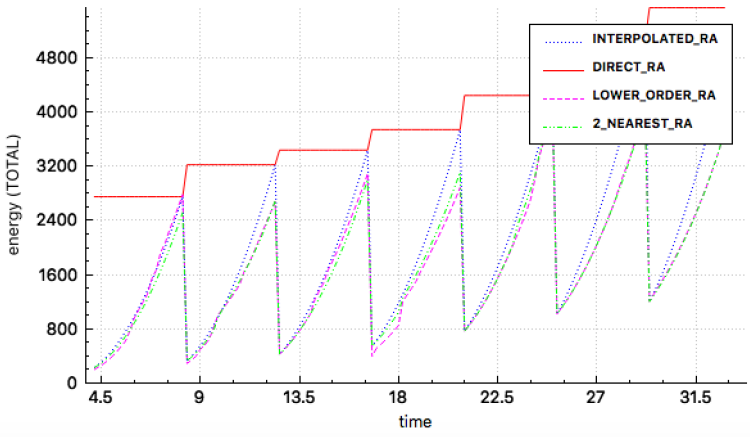
As seen in the MAX graphs, a network remains energy efficient under the variants of the *VORONOI_RA* algorithm in terms of maximum energy consumption at any time t. As explained already, this happens because a node's maximum range depends on the farthest vertex of its Voronoi cell. Thus, it cannot exceed that distance, while the algorithms, *DIRECT_RA* and *INTERPOLATED_RA*, can keep extending the range of a node to keep its farthest child node in its range.

In case of the TOTAL graphs for different networks, the energy efficiency depends on the direction factor α. In the case of lower values of α, there is less possibility of a node to go into Voronoi cells of many nodes. Thus, as seen in graphs 4.1b and 4.3b, the total energy consumption in the networks remain lower under the variants of *VORONOI_RA* algorithm. The reason for the decrease in efficiency is explained earlier, which is due to multiple responsible nodes trying to keep a node in their range at some later time t in most of the time intervals $[t_i, t_{i+1}]$. For example, a sudden increase in average and total energy consumption can be seen at time 15, and 20 seconds in graphs 4.2b, and 4.4b.

The results are also obtained for networks with a higher number of the nodes (#*nodes* $= 20$). Figure 4.5 and 4.6 show graphs for a network with *20* sensor nodes. The length of the time interval is kept as 4 seconds, and the maximum allowed speed is kept at 40. While, figure 4.5 shows results of a network with direction factor as $\alpha = 20$, and figure 4.6 shows graphs in case of the higher value of direction factor $\alpha = 300$. Due to a larger number of nodes, energy consumption seemed to be higher at some points in the graphs 4.5b with $\alpha = 20$ for the *VORONOI_RA* algorithm. Whereas the energy efficiency remains better under *VORONOI_RA* algorithm in terms of maximum (*MAX*) energy consumption. For the network with increased number of nodes and high value to direction factor α, the energy efficiency decreases in terms of total (*TOTAL*) energy consumption due to reasons discussed before.

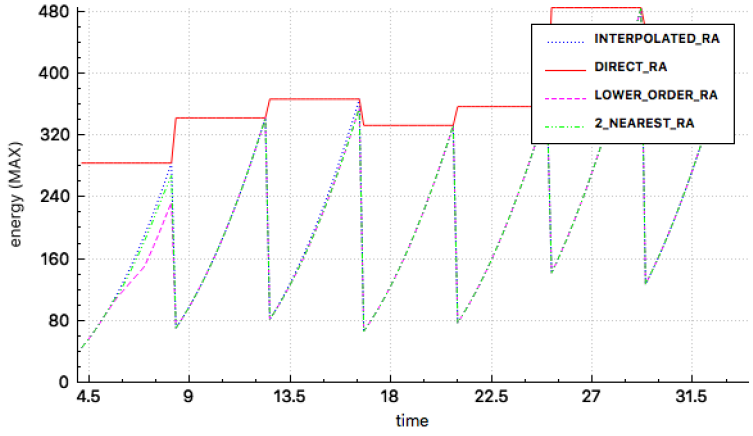


(a) maximum energy consumption over time

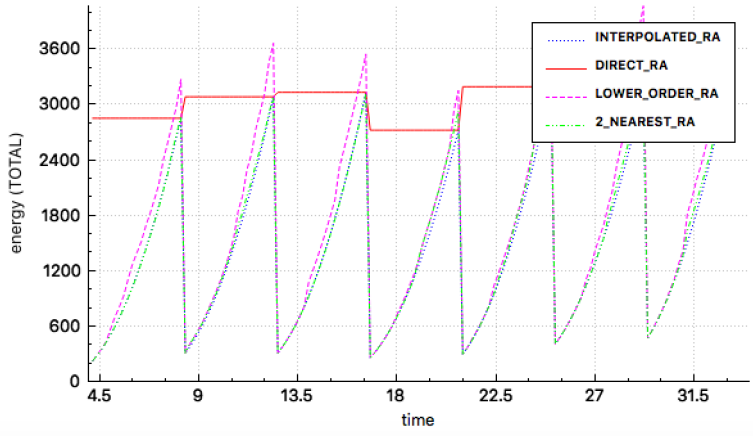


(b) total energy consumption over time

Figure 4.5: Above graphs show the values of performance measures with time t for the network with following network setting: $\#nodes = 20$, interval length is 4 seconds, $\alpha = 20^o$, and $V_{max} = 40$.



(a) maximum energy consumption over time



(b) total energy consumption over time

Figure 4.6: Above graphs show the values of performance measures with time t for the network with following network setting: $\#nodes = 20$, interval length is 4 seconds, $\alpha = 300^o$, and $V_{max} = 40$.

Chapter 5

Conclusion

In this chapter, we review the work done by presenting a small introduction to the problem, and making conclusions from the results that are obtained through careful experimentation. Later in this chapter, we present an idea for further improvements.

5.1 Summary

This thesis studies the important algorithmic problem of range-assignment for successful broadcasting in mobile ad-hoc network. We propose a novel algorithm for this problem with two variants: LOWER_ORDER_RA and 2_NEAREST_RA. The algorithm makes use the geometric data structure called Voronoi Diagram which is not studies before in existing literature. The algorithm executes and assigns a range to the nodes in the mobile network over time for successful broadcasting. An experimental setup is developed in Qt and C++ to analyse the performance of the proposed algorithm. We then analyzed the performance concerning MAX (maximum) and TOTAL (total) energy consumption in the network under different algorithms, both existing and proposed.

The results of out proposed algorithm regarding different performance measures are positively convincing. Our initial aim was to increase the energy efficiency of the network while maintaining the broadcasting, and it has been very frequently observed through the results of the proposed variants of the broadcast algorithm. The algorithm proved to be efficient in the case of MAX performance measure in all the different network parameters. Thus, the maximum energy consumption was kept lower or sometimes equal in comparison to the existing algorithm INTERPOLATED_RA. With respect to the performance measure, TOTAL, the algorithm seemed to be performing efficiently according to one of the primary factor that describes the movement model. It is referred as the direction factor and

denoted with α which describes the randomness in the movement of the nodes. For small values of the direction factor α, the algorithm performed efficiently on almost every network differing in their parameters. For larger values of the direction factor, the average and total energy consumption suddenly increased above the values obtained under the INTERPOLATED_RA algorithm. The increase in energy consumption mainly occurred at a later time in given time interval. Thus, the algorithm proved to be a little less efficient in case of higher values of direction factor.

To conclude, the overall performance of the broadcast algorithm proved better than the existing ones. The algorithm successfully maintained broadcasting in a given mobile ad-hoc network. The energy efficiency increased with decreasing the randomness in the movement of the nodes in the network. A small value of direction factor (low randomness in the movement) is more intuitive in real applications of a mobile ad-hoc network than the large value. Consider the example of driver-less cars, or cell phone devices; the movement of them is not always random, but they try to move in a path with a little deviation. In such applications, our algorithm proves to be a dependable solution for range assignment for a reliable broadcasting over time.

5.2 Future Work

The thesis has addressed the range-assignment problem and proposed an algorithm *VORONOI_RA*. There is still scope for further considerations. We have used a Voronoi diagram to effectively maintain broadcasting in a mobile ad-hoc network. The algorithm performed well in maintaining an energy efficient network for low values of the direction factor. The proposed algorithm is suitable for a network with constrained randomness in the movement of the nodes.

We have also tried to use a weighted Voronoi diagram (Power Diagram) to increase the performance of the algorithm in a network with high randomness in nodes movement. The main idea is that when computing the Voronoi diagram of a set $\mathcal{R}_w$ of responsible nodes, we should take into account the current ranges of the nodes in $\mathcal{R}_w$. Due to some difficulties in the implementation of the algorithm that uses Power diagram, we could not include the algorithm in this thesis. A weighted Voronoi diagram could be a possible solution to get an energy efficient network in a movement model where our current algorithm fails to perform well.

Bibliography

[1] de Berg, Roeloffzen, and Speckmann. "Kinetic convex hulls, Delaunay triangulations and connectivity structures in the black-box model". In: *Journal of Computational Geometry* 3 (2012), pp. 222–249.

[2] Jakub Jakubiak and Yevgeni Koucheryavy. "State of the art and research challenges for VANETs". In: *5th IEEE Consumer Communications and Networking Conference*. IEEE. 2008, pp. 912–916.

[3] Li, Hou, and Sha. "Design and analysis of an MST-based topology control algorithm". In: *Wireless Communications, IEEE Transactions on* 4 (2005), pp. 1195–1206.

[4] Li, Huang, and Xiao. "Energy efficient topology control algorithms for variant rate mobile sensor networks". In: *Mobile Ad-hoc and Sensor Networks. The 4th International Conference on*. IEEE. 2008, pp. 23–30.

[5] Lou and Wu. "Toward broadcast reliability in mobile ad hoc networks with double coverage". In: *Mobile Computing, IEEE Transactions on* 6.2 (2007), pp. 148–163.

[6] Ramanathan and Rosales-Hain. "Topology control of multihop wireless networks using transmit power adjustment". In: *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings.* Vol. 2. IEEE. 2000, pp. 404–413.

[7] Rodoplu and Meng. "Minimum energy mobile wireless networks". In: *IEEE Journal on Selected Areas in Communications* 17 (1999), pp. 1333–1344.

[8] Wu and Dai. "Mobility management and its applications in efficient broadcasting in mobile ad hoc networks". In: *Wireless Commu- nunications and Mobile Computing (WCMC)*. Vol. 1. IEEE. 2004, pp. 7–21.

[9] Wu and Dai. "Mobility-sensitive topology control in mobile ad hoc networks". In: *Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International*. IEEE. 2004, pp. 522–535.

[10] Zhao, Lloyd, and Ravi. "Topology Control for Constant Rate Mobile Networks". In: *Global Telecommunications Conference, 2006. GLOBECOM'06. IEEE*. IEEE. 2006, pp. 1–6.