Design Analysis and Algorithms

Graduate Project Report

Analysis of Sender based, and Receiver based flooding in MANETS

Chaitanya Sai Kumar Talluru

Venkata Bharath Malapati

Computer Science Department

Georgia State University,

Atlanta GA

Course : Design and analysis and algorithms

Instructor : Dr. Alex Zelikovsky

Spring 2020

Motivation:

Flooding is a simple broadcast protocol for delivering a message to all nodes in a network. We will consider the flooding problem in Mobile Adhoc Networks (MANET's). Since, the topology of MANETS is dynamic and flooding requires no prior knowledge of network topology, flooding will be the apt solution for transmitting the message over MANETS. The normal and simple flooding is pure flooding, flooding in which the node transmits the data to its neighbors only when it receives the message for the first time. However, since all the nodes in the network acts as forwarding nodes and as result the pure form of flooding suffers from broadcast storm problem. So, efficient flooding algorithms were demanded to reduce the following problems

a) **Redundant Rebroadcasts:** IF the node decides to broadcast a message which is already broadcasted earlier to its neighbors will create redundancy.
b) **Contention:** This arise due to retransmission between close proximity nodes.
c) **Collisions:** Timing of retransmissions is closely correlated and can result in collisions.

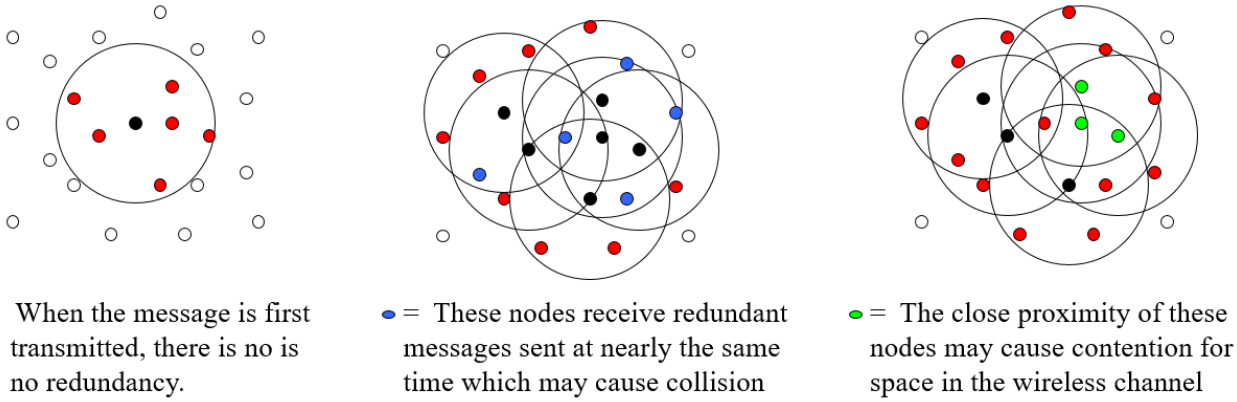


Fig 1: Illustration of broadcast storm problems

Flooding algorithms proposed can be classified into three main categories based on the information that each node contains: 1) No neighboring information 2) one hop neighbors information 3) Two hop neighbor information. The simple form of flooding known as pure flooding falls under 1st category which faces a crucial drawback of broadcast storm. The third category while it can still achieve best results in flooding but maintaining two hop node information will in turn result in huge overhead in the network with high mobility and density.

Predominantly, there are two approaches of achieving flooding: Sender based flooding and Receiver based flooding. In sender based receiver the forwarding node selects the subset of neighbors to be the next hop forwarding node and while forwarding the message the node attaches the list of its forwarding neighbors two the flooding message. On the other hand, the receiver based flooding does not need to attach any such information. The node itself makes the forwarding decision.

There are numerous algorithms described in Sender based, the most prominent are based on Minimum Forwarding Set Problem (MFSP). This problem falls under NP-complete in general graphs. Calinescu et al, studied this problem using unit disks and proposed two algorithms using location information of neighbors. First approach had an approximation ratio of 6 and time complexity of O(nlog n), the other approach achieved an approximation ratio of 3 and O(nlog 2n), n denotes the number of 2-hop neighbors. Liu et al. Proposed O(nlog n) using 1-hop neighbors and Khabbazian proposed O(n) algorithm using 1-hop neighbors.

When it comes to Receiver Based algorithms, the main advantage is that without knowing any prior knowledge of forwarding nodes or topology the algorithm itself computes to which node it must transmit once the node receives the message, unlike in the sender based were forwarding node set is transmitted along with the forwarding message. Since, adhoc networks are most volatile, they are prone to network changes this approach helps in detecting the most efficient and shortest paths with less number of transmissions and redundancy.

Problem Statement:

The Main function of RBF algo's is to find whether to forward the received message to its 1-hop neighboring nodes or not. The nodes for which the message need to be forwarded are computed by algorithm and return a Boolean result. Whenever a node receives a message it does not contain Forwarding set Information, unlike in SBF were the forwarding nodes set information is sent along with the forwarding message. There are numerous researches going on to provide the best possible optimal solution of finding the forwarding nodes with less possible time, few among them were Scalable Broadcast Algorithm (SBA) by peng using 2-hop and 1-hop topology, Khabbazian algorithm using 2-hop and 1-hop topology. In this paper we developed an RBF algorithm that can be achieved in O(nlog n) time complexity avoiding redundancy and collisions. As part of proposed algorithm, the key issue is to address the following problem: Given a node v, for a set of its neighbors {a1;a2;... ;an}, how to compute boundary of the union of cover areas of {a1;a2…an} in O(log n).

RBF algorithm:

In RBF algorithm, it checks whether the message being received previously, if not it will initiates a defer timer by a random number and starts listening to the channel. The node continues to listen until the channel is idle, then it starts decreasing the defer timer at the end of each time unit. The message broadcasting is decided when the defer timer expires. First we will see how $O(n^2)$ is achieved and in later phases we will show O(nlog n) algorithm implementation.

Algorithm 1 – O(n^2) implementation:

As part of this approach the author proposed in two stages,

First stage: whenever a receiver receives a message from the sender, the receiver divides the nodes inside its coverage area into three categories using location of the nodes.

Second stage: Now, retransmission rule 1 as explained below applied, if the rule is satisfied then the message won't be transmitted, if not it will be transmitted. Every neighbor node which receives the message repeats this procedure.

Considerations: In order to achieve 1- hop flooding, the author used the following considerations.

a) *Coverage disk of a node:* Let's say for a node "u", the coverage disk would be defined as c(u) where radius is equal to transmission range of "u" and the disk centered at "u". So, if a point "v" is in a neighborhood set of "u", then it is said to be "v" lies in coverage of "u".
b) *Coverage area of node-set:* The coverage area of a node set "A" is defined as C(A) i.e. the union of coverage disks of nodes in "A".
c) *Distance between two nodes:* We assume that the distance we compute for two points p1 and p2 will be Euclidean distance and is denoted by $d(u,v)=\sqrt{(u_x-v_x)^2+(u_y-v_y)^2}$

Algorithm in-detail Explanation:

Let's consider the image right side, we can see that the node "s" acts as the sender and the receiver "r" receives the message from "s", then the algorithm divides the neighbors of "r" into three groups.

Group 1: Nodes in the transmission range of s whose distance to s is shorter than that from r to s

$\{u \mid u \in c(r) \cap u \in c(s) \cap d(u,s) \leq d(r,s)\}$

Group 2: Nodes in the transmission range of s whose distance to s is larger than the distance from r to s

$\{u \mid u \in c(r) \cap u \in c(s) \cap d(u,s) > d(r,s)\}$

Group 3: Nodes that are not in transmission range "s" $\{u \mid \hat{u} \in c(r) \cap u \notin c(s)\}$

Transmitting Message:

Retransmission rule: If all nodes in Group 3 are covered by Group 2 then retransmission with respect to "r" is redundant and can be omitted. $\forall\, v \notin Group\#3,\, v \notin C(Group\#2)$: r should not retransmit

Now, if we observe the figure shown above, we can see that if the nodes in group 2 covers all the nodes in group 3, then transmission from "r" and again from group 2, will cause double transmissions to nodes in group 3, so to avoid this, we will restrict that either "r" or nodes in group "2" will retransmit and other won't/ Here author restricted "r" because nodes in group 2 can cover larger area, so if the nodes in group 3 are covered by group 2, then "r" won't retransmit.

```
1  Foreach node u in coverage area of r
2    If d(u,s) ≤ d(r,s) then Add u to Group#1
3    If d(u,s) > d(r,s) then Add u to Group#2
4    If u ∉ c(s) then Add u to Group#3
5  Endfor
```

Pseudo Code for diving into groups

```
1  Foreach node u in Group #3
2    cover = false
3    Foreach node v in Group #2
4      If u ∈ c(v) then
5        cover = true
6        break
7    If !cover then return false
8  return true
```

Pseudo Code for retransmission logic

Algorithm 2: O(nlog n) implementation:

The following considerations are made in order to illustrate the algorithm. Let's say if node "v" receives a flooding message, N(v) denotes neighbors of "v". During the defer time if the node "v" receives the same message from its neighbor set S(v)⊂N(v). The cover area of node in S(v) is denoted as C(S(v)). P(v) and Q(v) are subsets of v's such that P(v)⊂N(v) and Q(v)⊂N(v).

$$P(v) = \{a \mid a \in N(v), a \in C(S(v)) \text{ and } ID_a < ID_v\}$$
$$Q(v) = \{a \mid a \in N(v), a \notin C(S(v))\}$$

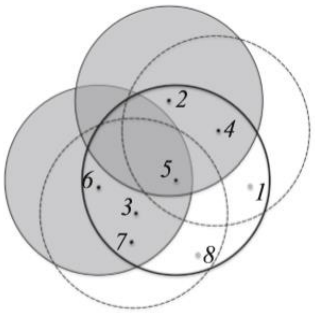


Fig 2: Illustration figure of RBF

As shown in the right side figure, let's say node N5 receives the flooding message from N2 and N6 during the defer time. We will have S(N_5) = {N_2, N_6}, the shaded area Is C($S(N_5)$) the cover area of N_2 and N_6. From the picture, we can also see that nodes, N_3, N_4, N_7 are in C($S(N_5)$), P(N_5) = {N_3, N_4} and Q(N_5) = {N_1, N_8}.

The main idea of RBF is that if any node in Q(v) is covered by at least one node in P(v), then node v is not required to forward the flooding message. So, if |uv| < R then the coverage condition holds and "v" does not forward the message. In order to achieve, RBF flooding we need to compute the neighbor set of nodes in P(v), and then check that whether all nodes in Q(v) are in this neighbor set. Let's say O(nlog n) implementation of algorithm to achieve this.

Algorithm Explanation:

In brief, the algorithm is divided into two stages. First we compute the C($S(v)$) followed by that P(v) and Q(v) are computed. Second, boundary of p(v) i.e. C($P(v)$) is computed and we will check whether all nodes in Q(v) are covered by C($P(v)$).

To start off, the boundary for the cover areas of a set of nodes is represented as the union of a set of arcs. Each arc is denoted as a triple (start angle, center, ending angle). For instance, as shown in fig 3, suppose x, y, and z are neighbors of node v, then C({x,y,z}) is represented as

$$\{\widehat{od}, \widehat{db}, \widehat{ba}, \widehat{ac}, \widehat{co}\} = \{(0, z, \angle ovd), (\angle ovd, y, \angle ovb), (\angle ovb, x, \angle ova), (\angle ova, y, \angle ovc), (\angle ovc, z, 2\pi)\}.$$

In fig 3, the cover area of the adjacent nodes of node v is divided into five sectors: ovd, dvb, bva, avc, and cvo. We use $Sector_{arc}$ to denote the corresponding sector of an arc. For a neighbor u of v, its angle is defined as the angle from ov to uv according to the anticlockwise direction. For example, in Fig. 3, the angle of y is ∠ovy.

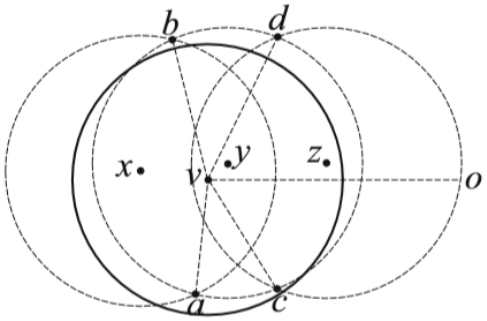


Fig 3: The boundary formed by union of arcs

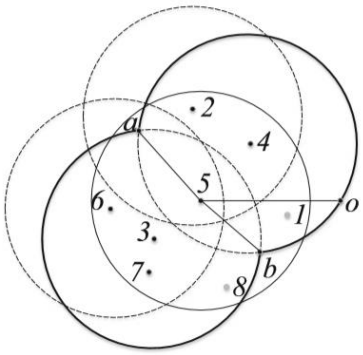


Fig 4: An example illustarting O(nlog n) RBF algorithm

If we consider fig 4, and assume, node 5 receives message from 2 and 6 i.e. S(N_5) = {N_2, N_6} and P(N_5) = {N_3, N_4} and Q(N_5) = {N_1, N_8} as shown in figure 4, then as per the algorithm first we need to compute the $ARC_{P(v)}$, which will be $ARC_{P(v)}$ ={arc_{oa} , arc_{ab}, arc_{bo}}i.e. the bolded arcs as represented in the fig 4. Now, for each point in Q(v), we check whether they fall under one of the sector of P(v), i.e. in this example, sectors of P(v) are {oa,ab,bo}. So, first point in Q(v) is N_8, since N_8 falls under sector ab, we check for next point in Q(v), i.e. for N_1, we can see that N_1also falls under sector bo. Since all the points in Q(v) fall under P(v) sectors, the message is not transmitted anymore from 5.

Details of the implementation is presented in Algorithm 2. In this implementation, the boundary of C(S(v)) and C(P(v)) can be obtained in O(nlogn) time using the Liu's algorithm in [18]. In the first half of Algorithm 2, we traverse $List_{N(v)}$ and $ARC_{S(v)}$ at the same time to compute P(v) and Q(v). In the second half of algorithm, we traverse $List_{Q(v)}$ and $ARC_{P(v)}$ to check that whether every node in Q(v) is covered by C(P(v)) i.e. we check whether they are covered by arcs of P(v).

Step 1 and step 20 of Algorithm 2 respectively have time complexity O(nlogn) and O(mlogm) according to [18], where m is the number of nodes in P(v). With the quick-sort algorithm, step 2 can be finished in O(nlogn) time. Steps 3–19 have time complexity O(n + l), where l is the number of nodes in S(v). Step 21 has time complexity O(klogk), where k is the number of nodes in Q(v). Steps 22–31 have time complexity O(k+m). Since k<=n; l<=n; m<=n, the time complexity of Algorithm 2 is O(nlogn).

Input: $List_{N(v)}, List_{S(v)}$
Output: Output *true* if Coverage Condition 1 holds; Otherwise output *false*.

Compute the boundary of $C(S(v))$, output to $ARC_{S(v)}$
Sort nodes in $List_{N(v)}$ in an ascending order with their angles
Initiate two empty node lists $List_{P(v)}$ and $List_{Q(v)}$
$i \leftarrow 1, j \leftarrow 1$
while *true* **do**
    if *the angle of* $List_{N(v)}[i]$ *is bigger than the end angle of* $ARC_{S(v)}[j]$ **then**
        $j \leftarrow j + 1$
        continue
    if $List_{N(v)}[i]$ *is in sector* $_{ARC_{S(v)}[j]}$ *and* $ID_{List_{N(v)}[i]} < ID_v$ **then**
        $List_{P(v)} \leftarrow List_{P(v)} + \{List_{N(v)}[i]\}$
    else if $List_{N(v)}[i]$ *is not in sector* $_{ARC_{S(v)}[j]}$ **then**
        $List_{Q(v)} \leftarrow List_{Q(v)} + \{List_{N(v)}[i]\}$
    $i \leftarrow i + 1$
    if $i > length(List_{N(v)})$ **then**
        break
if $length(List_{Q(v)}) = 0$ **then**
    return *true*
else if $length(List_{Q(v)}) > 0$ *and* $length(List_{P(v)}) = 0$ **then**
    return *false*
Compute the boundary of $C(P(v))$, output to $ARC_{P(v)}$
Sort $List_{Q(v)}$ in an ascending order with their angles
$i \leftarrow 1, j \leftarrow 1$
while *true* **do**
    if $List_{Q(v)}[i]$ *is in sector* $_{ARC_{P(v)}[j]}$ **then**
        $i \leftarrow i + 1$
    else
        $j \leftarrow j + 1$
    if $i > length(List_{Q(v)})$ **then**
        return *true*
    else if $j > length(ARC_{P(v)})$ **then**
        return *false*

O(n log n) implementation algorithm of RBF

Simulations and Results:

In This project python is used for implementation, As initial step 100 random points are generated in Euclidean plain in 10 x 10 grid. Now, we applied RBF algorithm 1 and 2 and tried to depict the forwarding nodes and as a result the resulting transmitting node are as shown in figure.8.

The step by step process shown in fig …, The combined result of flooding using RBF will look as shown in below figure.6 Where circles represents the coverage and orange nodes are message received nodes, green is a receiver, red is the extra/ initial sender node.

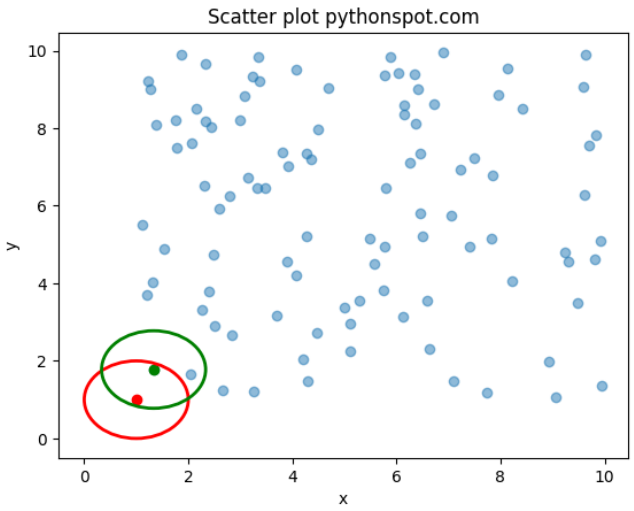


Fig 5: showing the 100 point in with source node in red color

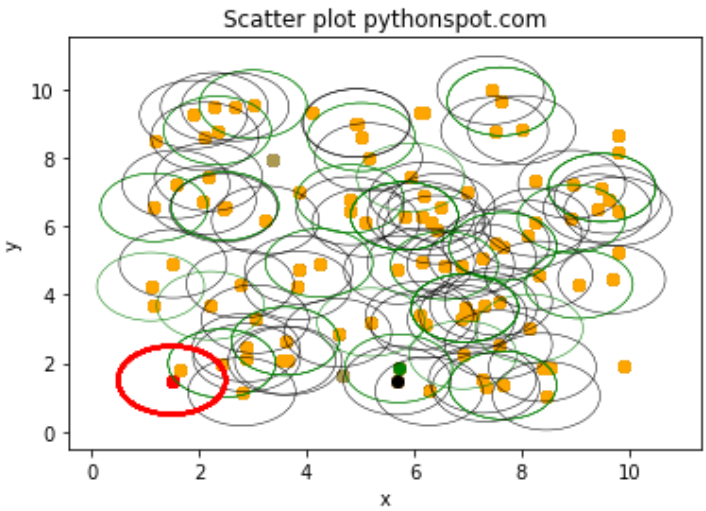
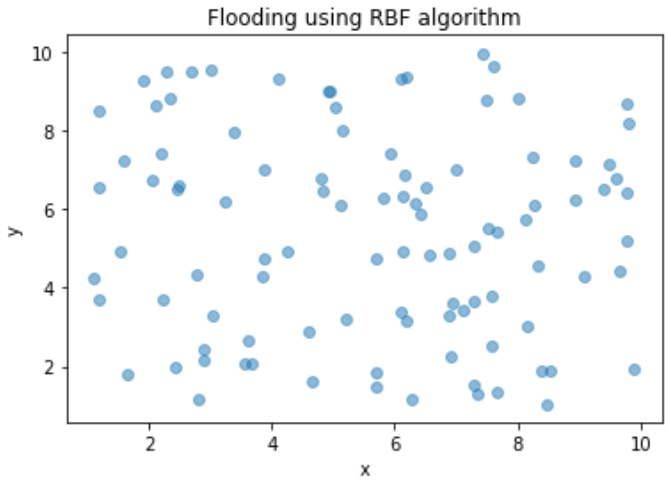


Fig 6: showing the end result after RBF, with range nodes represent that a node received message

Here as mentioned in the above algorithm The RBF algorithm reduces the number of retransmissions. The figures below show the total 100 nodes with blue color, the nodes that need to retransmit using RBF algorithm in red color.



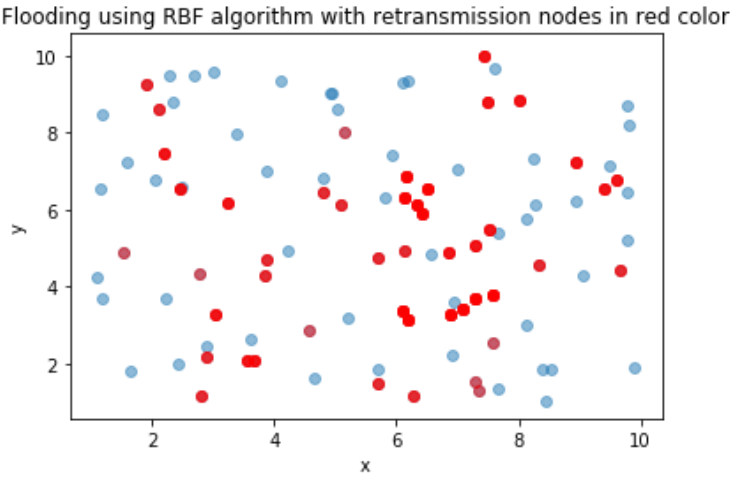


Fig 7: Showing the 100 nodes in a 10x10 grid before flooding Fig 8: Showing the retransmission nodes after flodding and RBF

From the above result and figures, With RBF algorithm in flooding there is only 45 % of retransmission takes place. This lead to decrease in almost 55% of retransmission when compared compared normal flooding. The RBF algorithm as explained can be achieved in O(nlogn). The Forwarding ratio for this case is 45.

Since the constrain of the implementation is for 100 nodes in a 10x10 grid even if increase the number of nodes the forwarding ratio is 99% equal as shown in below simulation.

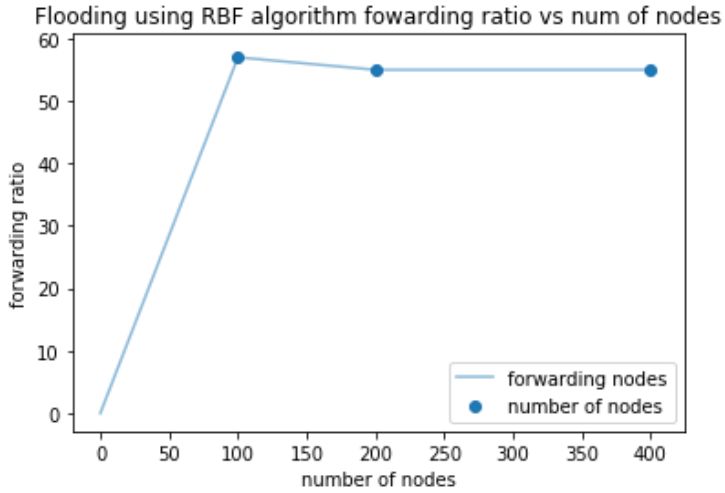
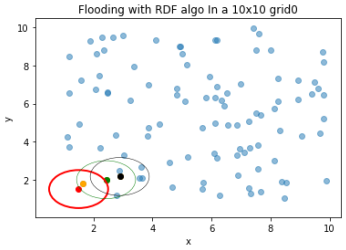


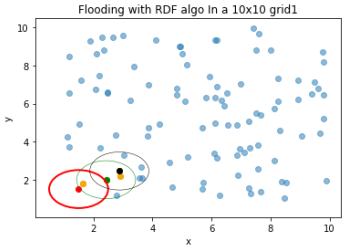
Fig 9: showing the forwarding ratio vs number of nodes

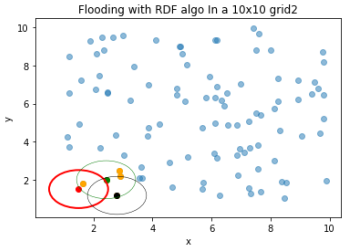
Step by step algorithm execution Screens:

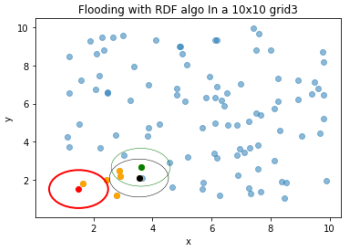
Step by Step results of MANET flooding using RBF algorithm are shown below fig.1, the figures initially consist light of blue color dots, which are 100 nodes in a 10x 10 (randomly generated). Then a source node is introduced, from it the flooding and RBF algorithm working takes places. As mentioned above the algorithm will give a Boolean value whether to retransmit or not, it the message will be retransmitted as

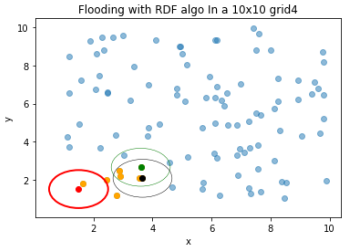
witnessed in below figures. The orange color of nodes indicate the message is received to that node, the green node is the receiver node and black color node is the one neighbor of green node. This process continuous through until the completion of flooding. It is interpretable from below figure that after 58 stages the flooding is done and all the blue nodes become orange. And also, one can interpret the unit disks to the sender, receiver covering their one hops.

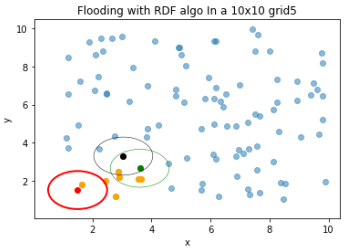


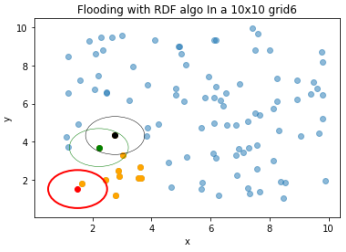


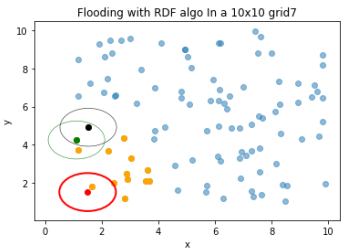


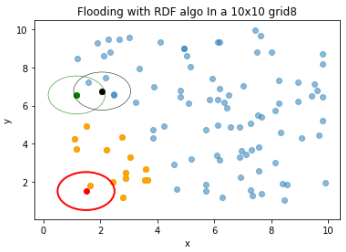


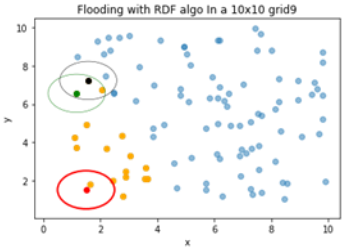


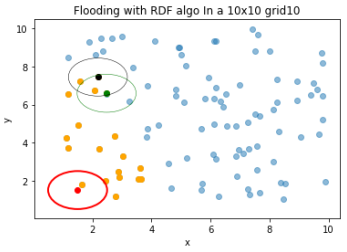


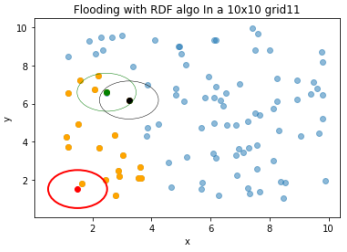


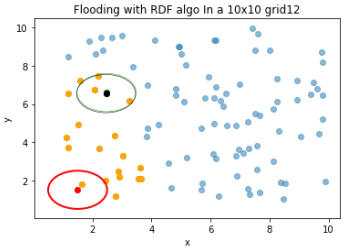


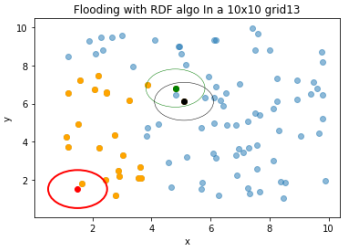


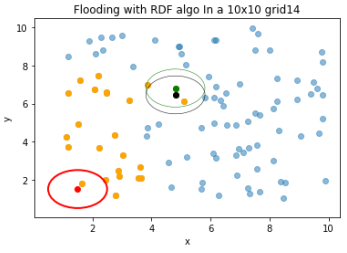


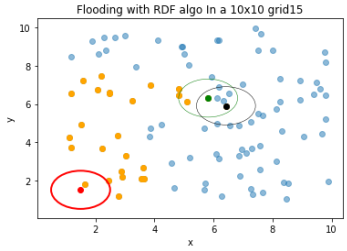


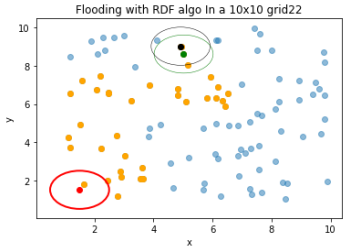


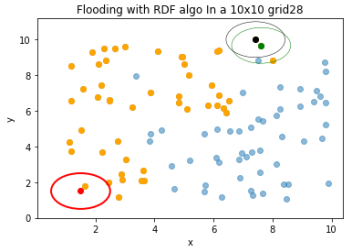


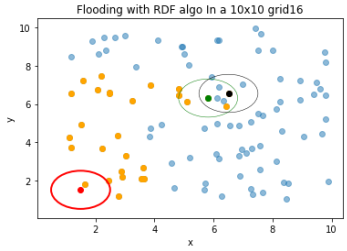


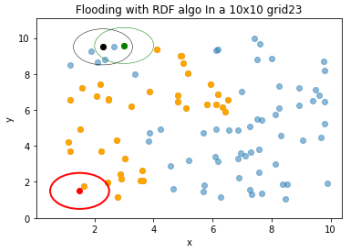


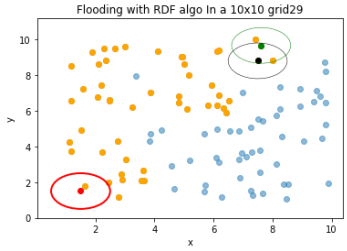

Flooding with RDF algo In a 10x10 grid15
x
y

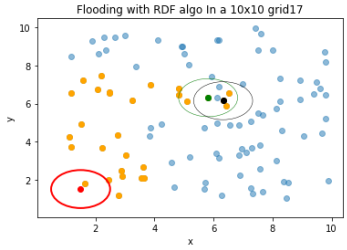

Flooding with RDF algo In a 10x10 grid22
x
y

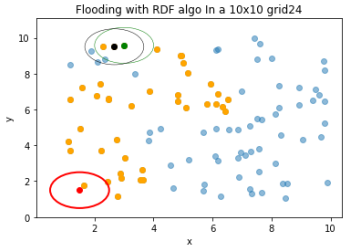

Flooding with RDF algo In a 10x10 grid28
x
y

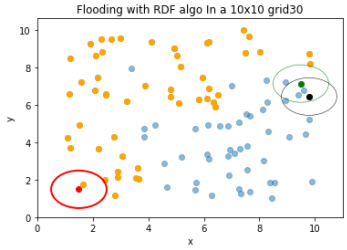

Flooding with RDF algo In a 10x10 grid16
x
y

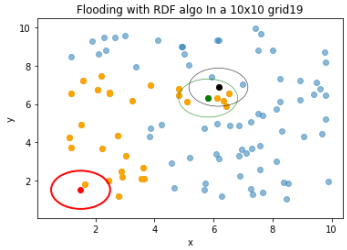

Flooding with RDF algo In a 10x10 grid23
x
y

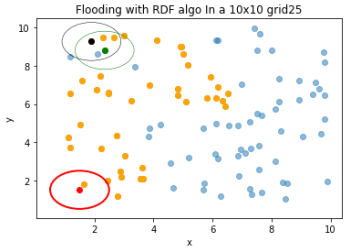

Flooding with RDF algo In a 10x10 grid29
x
y

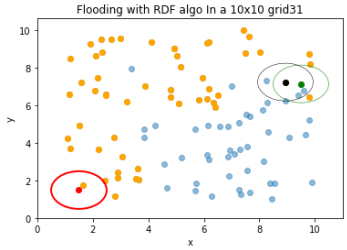

Flooding with RDF algo In a 10x10 grid17
x
y

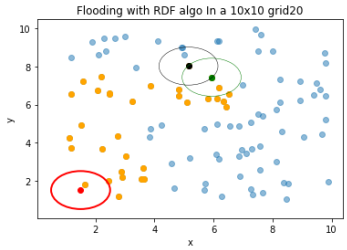

Flooding with RDF algo In a 10x10 grid24
x
y

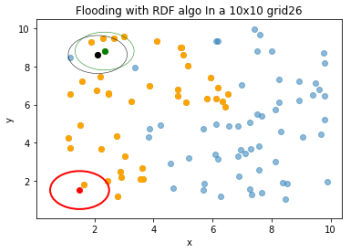

Flooding with RDF algo In a 10x10 grid30
x
y

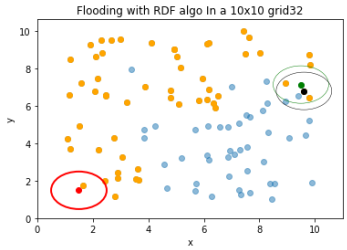

Flooding with RDF algo In a 10x10 grid19
x
y

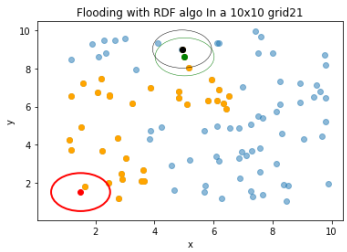

Flooding with RDF algo In a 10x10 grid25
x
y

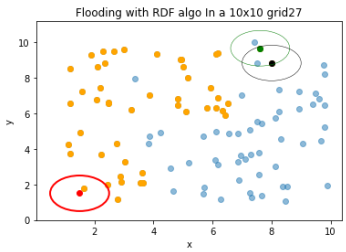

Flooding with RDF algo In a 10x10 grid31
x
y

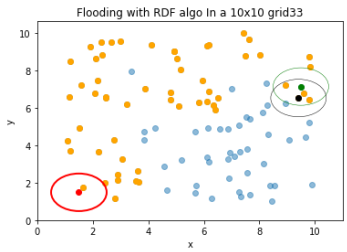

Flooding with RDF algo In a 10x10 grid20
x
y

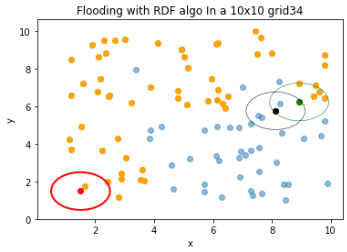

Flooding with RDF algo In a 10x10 grid26
x
y

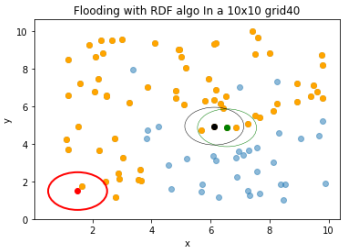

Flooding with RDF algo In a 10x10 grid32
x
y

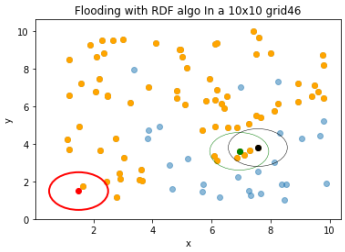

Flooding with RDF algo In a 10x10 grid21
x
y

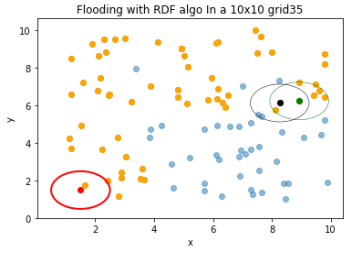

Flooding with RDF algo In a 10x10 grid27
x
y

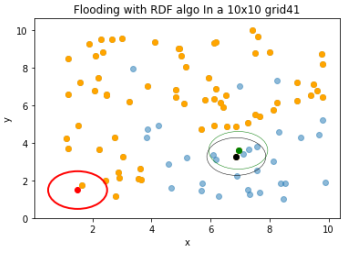

Flooding with RDF algo In a 10x10 grid33
x
y

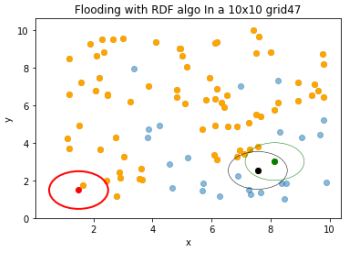

Flooding with RDF algo In a 10x10 grid34
x
y

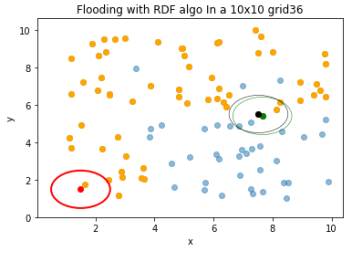

Flooding with RDF algo In a 10x10 grid40
x
y

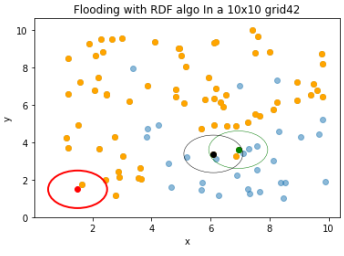

Flooding with RDF algo In a 10x10 grid46
x
y

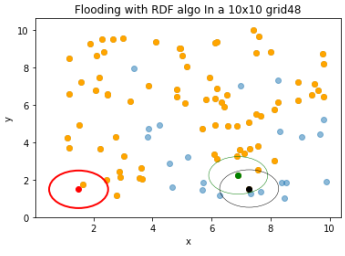

Flooding with RDF algo In a 10x10 grid35
x
y

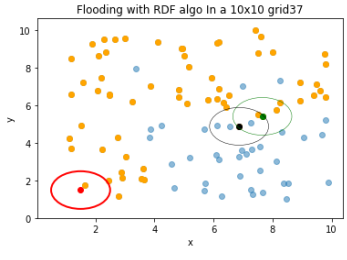

Flooding with RDF algo In a 10x10 grid41
x
y

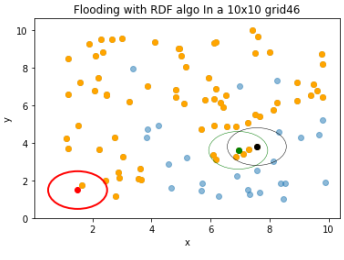

Flooding with RDF algo In a 10x10 grid47
x
y

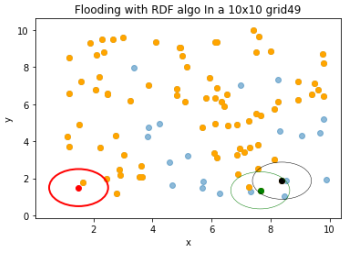

Flooding with RDF algo In a 10x10 grid36
x
y

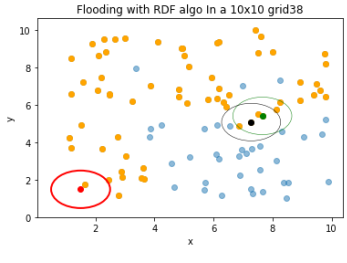

Flooding with RDF algo In a 10x10 grid42
x
y

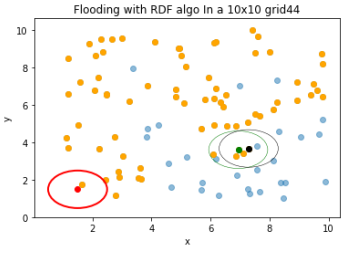

Flooding with RDF algo In a 10x10 grid48
x
y

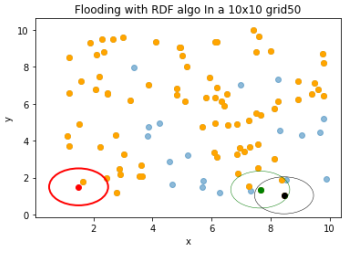

Flooding with RDF algo In a 10x10 grid37
x
y

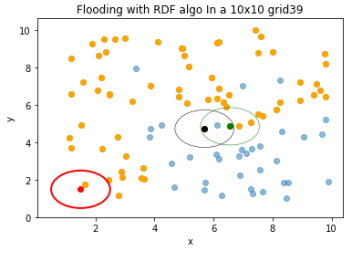

Flooding with RDF algo In a 10x10 grid46
x
y

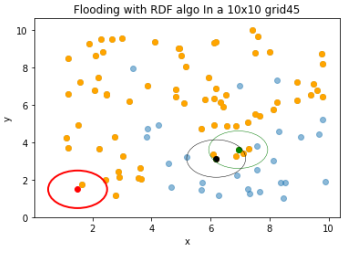

Flooding with RDF algo In a 10x10 grid49
x
y

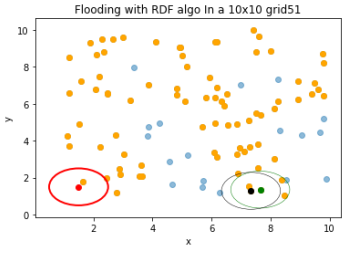

Flooding with RDF algo In a 10x10 grid38
x
y

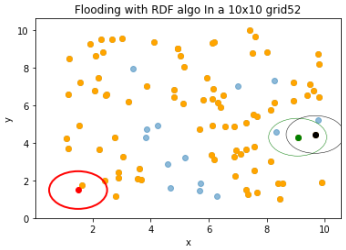

Flooding with RDF algo In a 10x10 grid44
x
y

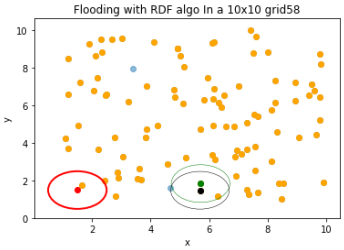

Flooding with RDF algo In a 10x10 grid50
x
y

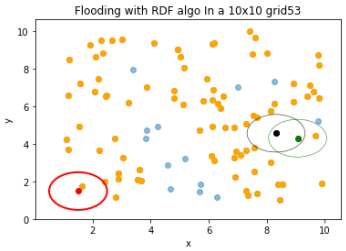

Flooding with RDF algo In a 10x10 grid39
x
y

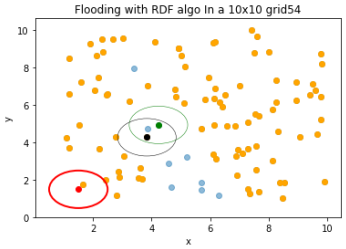

Flooding with RDF algo In a 10x10 grid45
x
y

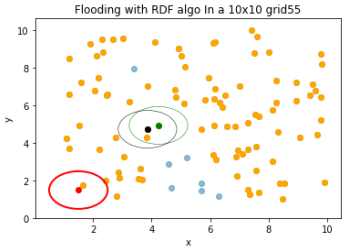

Flooding with RDF algo In a 10x10 grid51
x
y

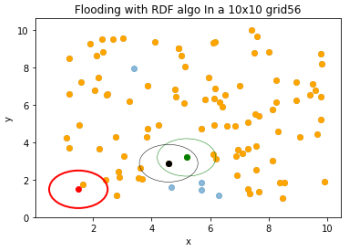

Flooding with RDF algo In a 10x10 grid52
x
y

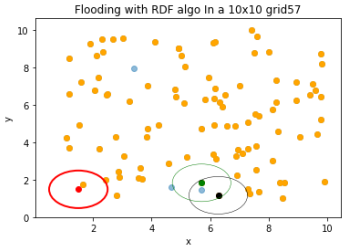

Flooding with RDF algo In a 10x10 grid58
x
y


Flooding with RDF algo In a 10x10 grid53
x
y


Flooding with RDF algo In a 10x10 grid54
x
y


Flooding with RDF algo In a 10x10 grid55
x
y


Flooding with RDF algo In a 10x10 grid56
x
y


Flooding with RDF algo In a 10x10 grid57
x
y

References:

1. Calinescu, G., Mandoiu, I. I., Wan, P. J., & Zelikovsky, A. Z. (2004). Selecting forwarding neighbors in wireless ad hoc networks. Mobile Networks and Applications, 9(2), 101–111.
2. Bai, X., Wei, X. & Bai, S. Efficient receiver-based flooding in mobile ad hoc networks. Wireless Netw 26, 17–31 (2020). https://doi.org/10.1007/s11276-018-1779-z