Throughput Maximization using Spatial Reusability in Multi-Hop Wireless Networks

Abhilash S. Bhise¹, Dr. Sachin D. Babar²

M. E. Department of Computer, SIT, Lonavala, India¹
Professor, Department of Computer Engineering, SIT, Lonavala, India²

ABSTRACT: In multi-hop wireless networks it is very important to find best path from source node to destination node to achieve high end-to-end throughput. For delivering a single packet and to find path with less transmission count or time today’s large number of routing protocols have been developed but such transmission count or time minimizing protocols cannot guarantee to achieve maximum end-to-end throughput. For improve the end-to-end throughput in multi-hop wireless networks in this paper we consider the spatial reusability technique. We propose two algorithms first is for single path namely spatial reusability-aware single path routing (SASR) and second for any path routing namely spatial reusability any path routing (SAAR) and for finding shortest path we propose Euclidean Distance algorithm.

KEYWORDS: Routing, wireless network, Hop to Hop to communication.

I. INTRODUCTION

It is extremely important to select route carefully which can maximize the end-to-end throughput especially in multi-hop wireless network this is because there is limited capacity of wireless communication media are available. Recently there large number of routing protocols has been developed for multi-hop wireless networks. [4],[14],[15]. The problem with existing protocols is that it minimizing the overall number of transmissions and does not maximize the end-to-end throughput for deliver a single packet from source node to destination node.

In this paper we work on two routing algorithm including single-path routing and anypath routing. In single-path routing protocols the task is to select path which is cost minimizing path with packets are delivered from source to destination node. Most of existing routing protocols like single path or anypath routing protocols working on link-quality aware routing metrics as link transmission count-based metrics (e.g., ETX[16] and EATX[17]) and link transmission time-based metrics (e.g., ETT[5] and EATT[8]). For minimizing the overall transmission count or time for delivering a packet they select any path.

This remainder of this paper is organized as follows. Section 2 discusses the related works. Section 3 include system architecture section 4 presents the System analysis. Finally, we conclude the paper and discuss or future work.

II. RELATED WORK


   Algorithm:
   - Striping Algorithm
   - Round Robin Algorithm

   Advantages:
   - Multiple radios are assigning a flow to a particular channel based on the load across all channels and to maintain this assignment for the duration of the flow.

   Disadvantages:
   - The neighboring nodes are not able to communicate with each other when that neighbours chosen by orthogonal to the channel and this chooses by wireless node.

2. Highly dynamic destination sequenced distance-vector routing (DSDV) for mobile computers [2].
Algorithm:
- Shortest-path algorithm
- Distributed Bellman-Ford (DBF) algorithm
- Distance-vector routing algorithm

Advantage and Disadvantage:
The problems arising with large populations of mobile hosts, which can cause route updates to be received in an order delaying the best metrics until after poorer metric routes are received, we have separated the route tables into two distinct structures.

3. A performance comparison of multi-hop wireless ad hoc network routing protocols [3].

Algorithm:
- Temporally-Ordered Routing

Advantage:
The DSDV over traditional distance vector protocols has advantages that it guarantees loop freedom.

Disadvantage:
These missing pieces greatly simplify the problem faced by the routing protocol, capture effects, propagation delay, MAC-layer collisions, and the effects of congestion due to large packet sizes are unaccounted for. Furthermore, broadcast and unicast packets were delivered with the same probability, and, as noted in this is not a realistic assumption.

III. SYSTEM ARCHITECTURE / SYSTEM OVERVIEW

1. System Model

For SAAR algorithm spatial reusability any path routing we have to define novel approach in spatial reusability aware routing scheme for participating node selection, cost calculation, and forwarding list determination. With different data rates in network simulator we design SASR algorithm and SAAR algorithm. The evaluation results show that our algorithms works improvement to end-to-end throughput compared with existing ones. Specifically, for single-path routing, a throughput gain up to 5.3 with a median of more than 60 percent is achieved in the case of single-flow, and an average gain of more than 20 percent is achieved with multiple flows; for any path routing, a median gain of 13.2 percent and the maximum gain up to 71.6 percent can be realized. Proposed work presents the results of a detailed packet level simulation comparing four multi-hop wireless ad hoc network routing protocols that cover a range.

2. System Architecture

Fig. 2. System Architecture
3. Spatial reusability-aware anypath routing

The single-path routing, which restricts the packets to be forwarded through a predetermined path from the source to the destination, anypath routing enables any intermediate node who overhears the packet to participate in packet forwarding. Therefore, in the case of any path routing, our objective is to pick a set of participating nodes \( Q \) (including the source), and the corresponding profile of distance/cost.

\[
C = (C_i) i \in Q \quad (1)
\]

and forwarder lists \( F = (F_i) i \in Q \) to minimize the spatial reusability aware any path cost \( C_{src} \). Here, having a smaller \( C_i \) means that node \( i \) is closer to the destination.

Algorithm 1. SASR-MIN Algorithm

This algorithm used for finding minimal path.
\( C = \) Path delivery time
\( Q = \) Covered link
\( M^* = \) Maximal non-interfering sets
\( I = \) non-interfering sets
\( P = \) Path

1. \( C \leftarrow 0 \)
   Initially cost mean delivery time is zero.
2. \( Q \leftarrow \emptyset \)
   Covered Set initially empty
3. while \( Q \neq P \)
   In this Step loop for \( Q \) is not equal to select path then it executed. Enter in loop.
4. foreach \( I \in M^* \) do
   In above step for loop executed up to \( I \) is an element of set \( M^* \).
5. \( I \leftarrow I \setminus Q \)
   In that Store link in \( I \) when set that contains all those link of \( I \) that are not in \( Q \).
6. if \( I \neq \emptyset \)
   If \( I \) is not equals to Empty and also we check value of cost of \( I \) up to threshold i.e. particular link.
   \( \alpha \leftarrow \frac{c(I)}{|I|} \)
   Store the value of cost in \( \alpha \).
8. \( \text{Temp} \leftarrow I \)
   And also non-interfering link are stored in Temp Variable.

9. End //End if loop
10. End //End Foreach loop
11. \( C \leftarrow C + c(\text{Temp}) \)
   Store total cost in \( C \).
12. \( I \leftarrow I \cup \{\text{Temp}\} \)
   Store the non-interfering link in \( I \).
13. \( Q \leftarrow Q \cup \{\text{Temp}\} \)
   Stored covered link in \( Q \)
14. End // End while loop
15. Return \( C \) and \( I \)

Algorithm 2. SASR-FF Algorithm

This algorithm used for minimal cost of path selection from set of paths.

\( P = \) Path
\( C = \) cost of path
$\mathcal{I} =$ non-interfering sets

1. Sort the links in $\mathcal{P}$ by cost in non-increasing order $L$
2. Initially we give one variable
   \begin{align*}
   K & \leftarrow 0 \\
   C & \leftarrow 0; \quad I \leftarrow \emptyset
   \end{align*}
3. Cost of path initially zero and $I$ is empty
   \begin{align*}
   C & \leftarrow 0; \quad I \leftarrow \emptyset
   \end{align*}
4. Execute the for loop for all links are sorted in $L$ and $(i, j)$ are the values are particular node
   \begin{align*}
   \text{foreach} \ (i, j) \in L \ do
   \end{align*}
5. \begin{align*}
   \text{fused} & \leftarrow \text{FALSE}; \\
   \text{In this step fused is false then go to for loop}
   \end{align*}
6. \begin{align*}
   \text{for} \ l \leftarrow 1 \ to \ k \ do
   \end{align*}
7. \begin{align*}
   \text{If route will be reusable then it assign to TRUE}
   \end{align*}
8. \begin{align*}
   \text{foreach} \ (i', j') \in I \ do
   \end{align*}
9. \begin{align*}
   \text{if} \ [(i, j), (i', j')] \in E \ then
   \end{align*}
10. \begin{align*}
   \text{reusable} & \leftarrow \text{FALSE}; \quad \text{If route is not reusable then assign to FALSE i.e. if node not sending more packet for different route .}
   \end{align*}
11. \begin{align*}
   \text{break;} \quad \text{end loop}
   \end{align*}
12. \begin{align*}
   \text{if} \ \text{reusable} \ then
   \end{align*}
13. \begin{align*}
   \text{If reusable is TRUE}
   \end{align*}
14. \begin{align*}
   \text{Then find max cost of path add to non-interfering set}
   \end{align*}
15. \begin{align*}
   I_k & \leftarrow I_k \cup \{(i, j)\}; \quad c_k \leftarrow \max\{(i', j')\}; \\
   \text{fused} & \leftarrow \text{TRUE}; \quad \text{If fusable is TRUE}
   \end{align*}
16. \begin{align*}
   \end{align*}
17. \begin{align*}
   \end{align*}
18. \begin{align*}
   \text{if} \ \text{not fused} \ then
   \end{align*}
19. \begin{align*}
   k & \leftarrow k+1 \\
   I & \leftarrow I \cup \{(i, j)\}; \quad c_k \leftarrow t_{ij}
   \end{align*}
20. \begin{align*}
   \text{end}
   \end{align*}
21. \begin{align*}
   \text{end loop}
   \end{align*}
22. \begin{align*}
   \text{for} / \leftarrow 1 \ to \ k \ do
   \end{align*}
23. \begin{align*}
   \quad C & \leftarrow C + c_i
   \end{align*}
24. \begin{align*}
   \text{end}
   \end{align*}
25. \begin{align*}
   \text{return} \ C \ and \ I
   \end{align*}

Algorithm 3. SAAR Algorithm

Input- $G= (N, E)$
A network graph, a source node Src, a destination node Dst.
Output- A set of participating nodes $Q$, and the corresponding profile of cost $C$ and forwarder lists $F$.

1. \begin{align*}
   \text{foreach} \ i \ in N \ do
   \end{align*}
   \begin{align*}
   \text{i node belong to N}
   \end{align*}
2. \begin{align*}
   C_i & \leftarrow +\infty; \quad F_i \leftarrow \emptyset; \quad \Omega(i, i) \leftarrow 1
   \end{align*}
   \begin{align*}
   \text{Initially C set infinity, and F set empty and probability matrix } \Omega \text{ set 1 for all nodes.}
   \end{align*}
3. end //end foreach loop
4. $C_{\text{Dst}} \leftarrow 0; q \leftarrow \text{Dst}; Q \leftarrow \{ \text{Dst} \}; I \leftarrow \{ \{ \text{Dst} \} \};$
   q=\text{last picked min-cost node}
5. while $q \neq \text{Dst}$ do
6.   foreach $(i, q) \in E$ $\land i \notin Q$ do
7.     $F_i \leftarrow F_i \cup \{q\}; \Omega_i \leftarrow \Omega$
8.     Add last picked min-cost node into forwarder list $F_i$ and probability matrix add node $i$.
9.     foreach $j \in Q$ do
10.    $\Omega_i(i, j) \leftarrow 0$
11.   Then set probability matrix set Zero.
12.    end //end foreach loop
13.   end //end foreach loop
14.   $(C_i, I_i) \leftarrow \text{CalculateCost} (i, I, \Omega, F)$;
15.   Calculate minimum cost with node $I$ and interfering set, probability matrix and forwarder list.
16. end //end foreach loop
17. $q=\arg\min (C_i); Q \leftarrow Q \cup \{q\};$ //last picked min-cost node add cost
18. and participating node set $Q$ into add $q$.
19. i.e $\forall q \in Q$
20. $\Omega \leftarrow \Omega_q; I \leftarrow I_q$
21. Set probability matrix into add last min-cost node and Interfering set also add $q$.
22. end //end while loop
23. return $Q, C, \text{and } F$;
24. Output is participating node and min-cost and forwarding list.

Algorithm 4. Function CalculateCost ()

Input-

$i=$ node
$I=$ non-interfering sets
$\Omega=$ a profile of relay probabilities
$F=$ a profile of forwarding sets

Output-

The fused anypath cost $C_i$ from node $i$ to the destination node, and updated non-interfering sets $I$.

1. $\text{fused} \leftarrow \text{FALSE}; C_i \leftarrow 0$;
2. Initially fused node is false and Cost set also set zero.
3. foreach $I \in I$ do
4.   $\text{reusable} \leftarrow \text{TRUE}$
   If node is reusable then Sst reusable node is true.
5.   foreach $(j, F_i) \in \text{do}$
6.   Node $j$ and forwarder node belongs to interfering set
7.   if $\text{if } (i, F_i)$ interfere with $(j, F_i)$ then
Algorithm 5. Euclidean Distance Algorithm

distance= distance
p and q are Euclidean Vectors
if p = (p1, p2, ..., pn) and q = (q1, q2, ..., qn) are two points in Euclidean n-space, then the distance (d) from p to q, or from q to p is given by the Pythagorean formula:

1. n=number of nodes
2. for i ← 1 to n do
3. for j ← 1 to n do
4. node[i]=p, q;
5. node[j]=p, q;
6. distance[i, j]=\sqrt{((q1 - p1)^2) + ((q2 - p2)^2) + \ldots + ((qn - pn)^2)}
7. end
8. end
9. return distance;

IV. EXPERIMENTAL RESULTS

We assess the performance of our SASR and SAAR algorithms in java. Considering that all the nodes use the same transmission rate, we can compare our algorithms with transmission count-based routing protocols and metrics. To be specific, we compared them with the ETX-based DSR [13] (announce by DSR-ETX) and the shortest any path first algorithm [13] based on MORE [12], respectively. Table 1 lists the parameters in our simulation. Specifically, to uniformly distribute 80 nodes in a 2,000 meter to 2,000 meter area, and considered two data rates of 802.11, including 11 and 54 Mbps.
TABLE I
SIMULATION PARAMETERS SETUP

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Nodes</td>
<td>10</td>
</tr>
<tr>
<td>Terrain Area</td>
<td>2,000m×2,000m</td>
</tr>
<tr>
<td>RTS/CTS</td>
<td>OFF</td>
</tr>
<tr>
<td>Packet Size</td>
<td>1,500 Bytes</td>
</tr>
<tr>
<td>Traffic Generator</td>
<td>CBR</td>
</tr>
<tr>
<td>CBR Rate</td>
<td>5 Mbps / 20 Mbps</td>
</tr>
<tr>
<td>802.11 Data Rate</td>
<td>11 Mbps / 54 Mbps</td>
</tr>
</tbody>
</table>

Fig 4. Implementation of SASR and Sending Packet

Fig 5. Throughput and Overhead Charts for SASR
V. CONCLUSION

Spatial reusability aware routing can efficiently improve the source to destination communication with high end throughput in multi-hop wireless networks, by carefully considering spatial reusability of the wireless communication media. This is done by the protocols, SASR and SAAR, for spatial reusability-aware single-path routing and any path routing, respectively. To contribute more for better energy efficiency system implement opportunistic routing to reduce energy consumption. Resource allocation algorithm is inquired to maximize the energy efficiency (EE) in multiuser decode-and-forward (DF) relay interference networks.

REFERENCES