A Community Based Routing Algorithm for Mobile Opportunistic Networks

D. Fernando & Dr. K. Thabotharan

Department of Computer Science, Faculty of Science, University Of Jaffna

Introduction

- Opportunistic networking is a kind of delay-tolerant networking in which a number of wireless mobile nodes that communicate with each other, without the support a network infrastructure.
- Opportunistic networking uses locally available wireless technologies such as Bluetooth for pair-wise data forwarding hoping that the data will ultimately reach the destination.
- Intermittent connectivity and long delays in data delivery are inherent properties of this kind of opportunistic networking and they pose us challenges in data delivery.

What is the problem with the existing methods?

- The existence of communities among larger groups of people presents us a use case for opportunistic networking where content of interest could be exchanged among the members of communities.
- In such communities members are not exactly fixed to a single community and are usually connected to several communities based on their interests.
- Forwarding and routing content of interest among these communities should take care of the interests of the members of communities and other inherent properties of opportunistic networking.
- Therefore a more efficient routing algorithm that can overcome the inherent problems of such a set up is needed.

In this work we propose a community-based forwarding approach which we name as SWift routing algorithm (SW algorithm) that can be used to send messages among the community members in an opportunistic manner. Our simulation-based results show that our proposed algorithm outperforms three well-known algorithms in the field under varying network conditions.

Methodology

The proposed SW routing algorithm is a multi-copy algorithm and it ensures that when a node receives a message from its neighbor, the node forwards the message only to half the number of its neighbors compared to the previous node which has just forwarded the message to this node. By doing this it ensures that the message is not forwarded infinite number of times among the nodes in the network. At the same time when a message is received the algorithm also compares whether the current time is greater than the total accept time. This Total Accept Time (TAT) is defined as,

TAT = MAST + MCT + MAT

where,

- **MAST** (Message Accept Start Time) is the time when a new message is accepted.
- **MCT** (Message Checking Time) is the amount of time a node will reject an incoming message it has already received.
- **MAT** (Message Accept Time) is the amount of time that is calculated by adding the message generate time with the message time to live and the message check time.

Algorithm 1: SW Routing Algorithm

```
receive a message from the neighbour ;
msg ← message containing the value for count;
c ← message checking time;
a ← message accept time;
if msg > 1 then
    if the node is interested in viewing the message then
        view the message;
        estimate the new value of count;
        send a copy of the message to neighbouring
         nodes;
    else
        estimate the new value of count;
        send a copy of the message to neighbouring
         nodes;
        end;
    end
else
    total_accept_time ←
     message_accept_start_time + c + a;
    if message_get_time > total_accept_time then
        remove it;
    else
        view the message;
        estimate the new value of count;
        send a copy of the message to neighbouring
         nodes;
    end
```

Our SW algorithm uses the following design principle:

1. When a node generates a message, it also determines the number of neighbors the message (which we name as the variable count) needs to be forwarded.
2. The node then passes the generated message to its neighbor along with the variable count.
3. If the receiving node is interested in this message it can view this message and before can forwarding the message to its neighboring.
4. If the node is not interested it can estimate the new value of the variable count by dividing current value of it by two and then can forward the message to its neighbor.
5. At the time of forwarding the node compares the value of the Total Accept Time (TAT) with the current time (message get time plus the duration taken by the message to be sent from the beginning up to this step) and if the current time is greater than the TAT then the message is removed from forwarding.
   Otherwise, it the message gets forwarded.

Experiments

For our simulation based experiments we use the ONE simulator to implement our proposed SW routing algorithm, Spray and wait, Direct Delivery and the Wave algorithm and we ran our experiments with selected simulation parameters listed in Table. During the experiments we also varied the such as message TTL from 300 to 500 minutes, message size from 400 KB to 1MB to 700 KB to 1MB, and the buffer size from 5MB to 7MB. We have collected the test results in trace files and have analyzed them for their performance.

Parameters	Values
Simulator	The ONE
Parameters Simulation Time	2532 sec
Message TTL	300 min
Buffer size	5 MB
Message size (Event size)	400kB - 1MB
Movement Model	Shortest path map based Movement
Community groups	6
Multi copies	10
MessageCheckingTime	100 min
MessageAcceptTime	200 min
Reports	Message Stats Report Delivered Messages Report

Simulation Results

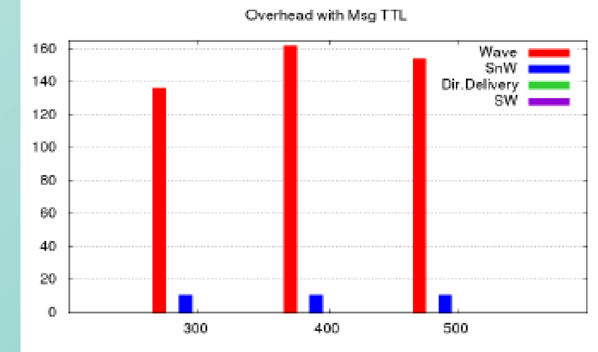

Fig. 1: Overhead with Message TTL

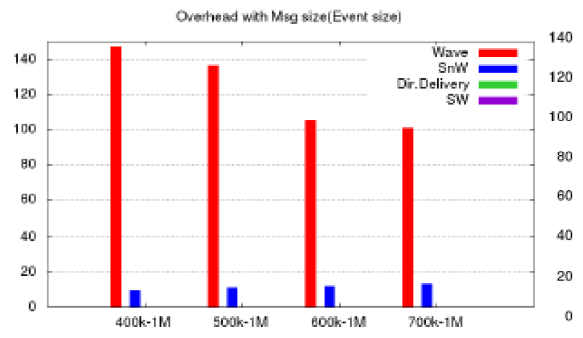

Fig. 2: Overhead with Message size

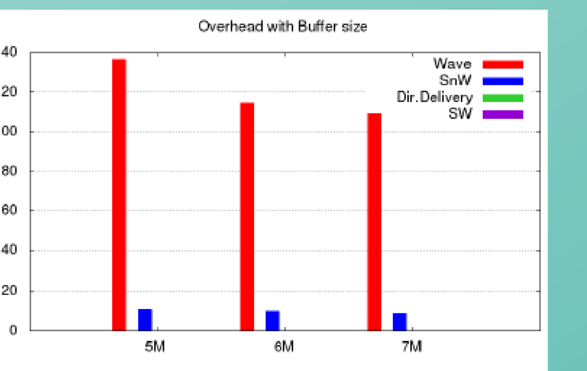

Fig. 3: Overhead with Buffer size

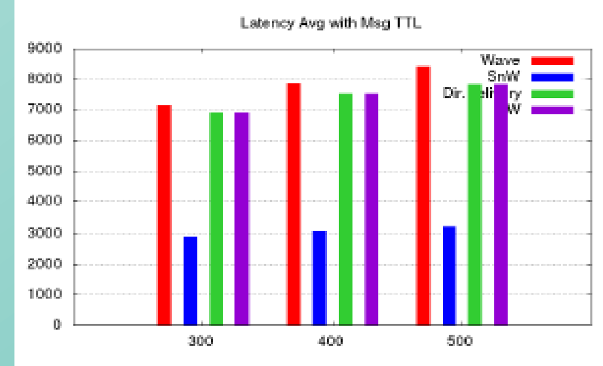

Fig. 4: Average Latency with TTL

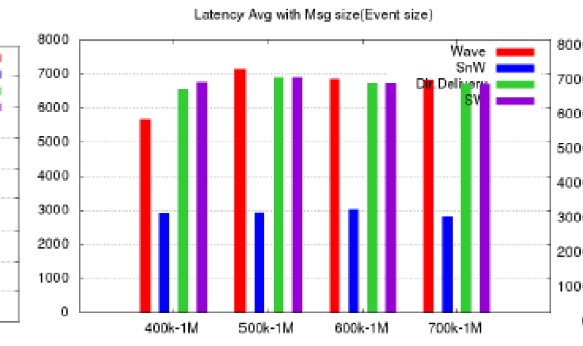

Fig. 5: Average Latency with Message size

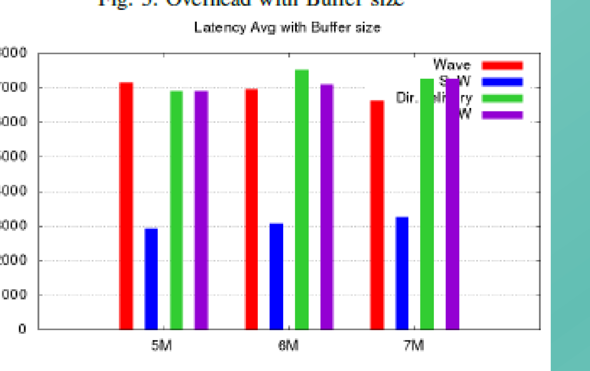

Fig. 6: Average Latency with Buffer size

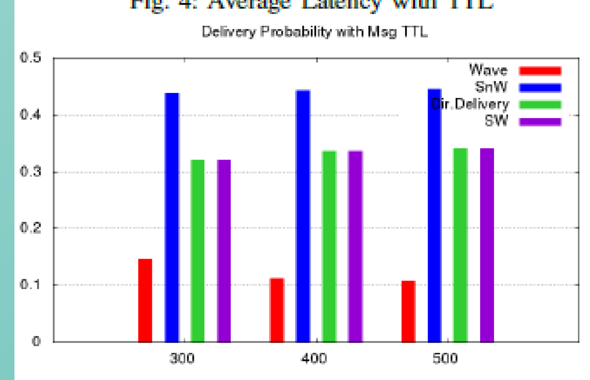

Fig. 7: Delivery probability with TTL

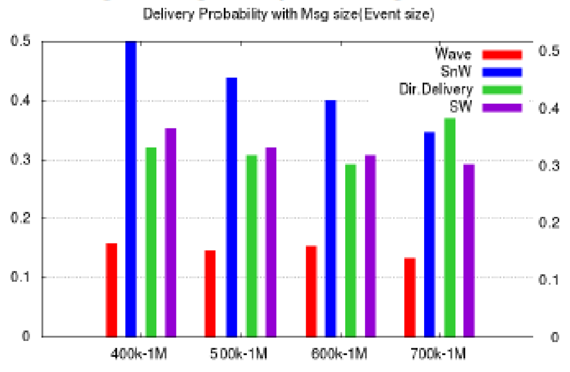

Fig. 8: Delivery probability with Message size

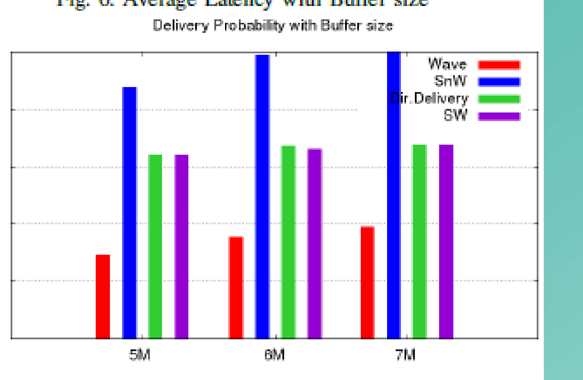

Fig. 9: Delivery probability with Buffer size

Performance Metrics

For the comparison of our proposed SW algorithm along with the three well known algorithms we use the following performance metrics:

Overhead Ratio: The overhead ratio reflects how many redundant packets are relayed to deliver one packet.
Overhead Ratio = (N - D)=D
(where N is the number of messages forwarded by a node, and D is the number of messages that are delivered to their destinations.)

Delivery probability: The delivery probability is the probability between the total number of messages delivered to their destinations and the total number of messages created at the source node.
Delivery ratio = P=T
(where P is a number of messages delivered to the destination and T is a number of messages made.)

Average latency: Average latency is the time between creating messages and receiving messages of a destination.
Average latency= (where n is the number of messages delivered to their destination, Ri is the moment when a message i reaches to its destination, and Si is the moment the message i is created.)

Discussion

Figures Fig. 1 to Fig. 3 show the results of comparison of message overhead with MessageTTL, Message size and Buffer size of Spray and Wait, Direct delivery, Wave and the SW protocols. When comparing the above four, our results show that the overhead of SW and Direct delivery are better than Wave, Spray and wait. The Direct delivery and the SW algorithm both get zero overhead ratio with MessageTTL, Message size and Buffer size. Compared to other algorithms, the SW algorithm shows the best performance.

Figures Fig. 4 to Fig. 6 show minimum latency with MessageTTL, Message Size and Buffer size in Spray and wait algorithm.

SW algorithm performs closer to Direct delivery algorithm. These two perform better when compared to the Wave algorithm.

Figures Fig. 7 to Fig. 9 show a higher delivery probability in Spray and Wait, where as the SW algorithm shows a steady delivery probability in these cases. The SW and the Direct delivery algorithm perform better when compared to the Wave algorithm. In overall, the SW algorithm outperforms the other three algorithms for the overhead with MessageTTL, Message size and Buffer size.

Since SW sends messages based on the node's interest, it was able to achieve this. We were also able to observe that the SW algorithm always shows a better performance than the Wave algorithm in all the test cases.

Conclusion

- Our SW routing algorithm's test results show that the outperforms three existing algorithms when compared with overhead with message TTL, Message size and Buffer size.
- In some of the case the proposed routing algorithm exhibits a steady performance when compared with the three algorithms.
- SW routing algorithm can use to promote business ideas based on customer's interest. We can send promotion messages among community members.
- As a future work we would like to improve the proposed algorithm for larger communities of people.
- Therefore different kinds of communities and their interests vary greatly, and the algorithm accurately takes care of routing in a more efficient manner.