Improvement on Gossip Routing Protocol using TOSSIM in Wireless Sensor Networks

Raju Dutta
Narula Institute of Technology
Dept. of Mathematics
Kolkata, West Bengal

Shishir Gupta
Indian School of Mines
Dept. of Applied Mathematics
Dhanbad, Jharkhand

Debraj Paul
Narula Institute of Technology
Dept. of Computer Science
Kolkata, West Bengal

ABSTRACT
Multicast routing protocols in the networks are inefficient to handle high priority traffic, network congestion, excessive processing of control information and retransmission procedures. Gossiping routing protocol is suitable and controlled form of reducing the overhead of routing protocols for forwarding a message. Gossiping protocol is characterized by each node randomly choosing to exchange information with another node. Modified Gossip has been considered as a scalable broadcast solution, considering destination node in wireless network. In this paper, we analyse Gossiping protocol and compare the performance of Gossiping and Modified Gossiping protocol, using new operating system TOSSIM, is a widely used operating system for modern wireless sensors. The protocols are simulated by TOSSIM and comparison results between the protocols are shown by different graphs

Keywords
Protocol; Gossip; Modified Gossip; TOSSIM; energy consumption; data transmission; lifetime; Wireless Sensor Network.

1. INTRODUCTION
Gossiping is implemented as a suitable and controlled form of flooding. Messages are propagated independently through the network with less congestion in the wireless medium. Gossiping routing protocol is reducing the overhead of routing protocols for forwarding a message and widely used for distributed protocol design. Gossiping may be of either uniform gossip or non-uniform gossip [1]. In uniform gossiping each node chooses its partner uniformly and randomly from all other neighbor node. Non-uniform gossiping is based on considerations of proximity and network topology. In gossiping based approach, each node forwards a message with probability to reduce the over-head of the routing protocols [2].The topological characteristics of the network have a global impact on suitability of gossip parameters. Several gossip based multicast protocols such as SRM, RMTP and PGM have been proposed to improve robustness of the networks. NACK based protocols are not suitable for networks where routes are changing rapidly. Gossip provide probabilistic reliability in wired networks well matched to theneeds of ad-hoc networks, since it is a controlled form of flooding with less congesting the medium and independent of topology. But the comparative performance analysis of the gossiping multicast protocols has not been done. We analyze in this paper the, packet delivery, mobility of gossip based multicast protocols and provide performance analysis of protocols having different characteristics in Modified Gossiping. The Gossiping and Modified Gossiping protocols were simulated by new language TinyOS compared the efficiency of the protocols that has been shown in different graphs.

2. RELATED WORK
The applications of the WSNs in military surveillance are very important and other areas on [3, 4, 5]. In [6] J.Haas et. al. proposed new techniques to increase the performance of routing protocol Gossip for Ad hoc network. They have given more concentration on packet delivery fraction, average delay and normalized routing load. Next M. Frikhaet. al in [7] suggested a routing protocol based on the current energy status of the node. There is an improvement in the control packets such as Route Request (RREQ), Route Reply (RREP) in all reactive protocol such as AODV and DSR. Due to flooding technique numbers of control packets are generated and the energy consumption of each node and the network increases. *AODV:* Ad hoc on demand Distance Vector protocol [8] is a reactive routing protocol that finds routes on demand basis. AODV have the characteristics of both DSR [9] and DSDV [10] protocols. They have routing table as in DSDV that contains the sequence number, next hop information which is also used to differentiate stale and fresh routes.

3. TinyOS AND NesC
TinyOS [11] is an open source component based operating system designed for wireless sensor networks. It features a component-based architecture which enables rapid innovation and implementation while minimizing code size as required by the severe memory constraints inherent in sensor networks. The TinyOS system, libraries, and applications are written in nesC, a dialect of C programming language optimized for the memory limitations of the sensor networks. nesC supports the TinyOS concurrency model and its programs are a set of software components which are connected to each other using interfaces. A nesC [12] application consists of components which can use or provide interfaces and different components are connected using these interfaces. An interface in a nesC application consists of commands and events. A component which provides the interface has to provide the implementation for the commands in that interface and can signal the events to the components using that interface. On the other hand, a component using the interface has to provide the implementation for the event handlers in the interface. Modules in TinyOS provide the implementation of the components and the configuration. A scenario is a collection of components and the wiring between the interfaces of these components which describes the complete application.NesC programs are built out of software components some of which are hardware abstractions. TinyOS provides interfaces and components for common abstractions such as packet

communication, routing and sensing. The framework developed in this paper, provides the user with higher level abstractions with some generic services so that they can be directly used. It also aims at providing the user a global view of application while abstracting the underlying communication details. TOSSIM is nesC based language. In nesC application a component provides and uses interfaces which are the only point of access to the component. An interface generally models some services in nesC. They contain commands and events. Commands are typically request to execute an operation i.e sending some request for job such as *message.send()* this command function is sending temp component to Main Component through StdControl interface. Fig. 1 shows route advertisement Interface Command in Gossiping Protocol by the command *(RoutePacket *) & routeMsg.data[]*. The command *sendMsg.dend()* and toggle red LED to indicate transmit data by command *led.redToggle()*. In this phase one node invites other nodes in the neighbor table to accept the broadcast massage and act as a broadcasting node. In Fig. 2 *routeselect.isActive()* always return a valid route in Gossiping when nodes broadcast massages. Fig. 3 shows route advertisement by the nodes and massage send Command in Modified Gossiping Protocol by the command *SendMsg.send(TOSS_BCAST_ADDR, length++, &routeMsg)*. In Fig. 4 it shows route selection of Modified Gossiping Protocol till packets send to the destination.

```
task void advertise()
{
RoutePacket *pRP = (RoutePacket *) &routeMsg.data[0];
uint8_t length = sizeof(RoutePacket);
if (sendRouteBusy == TRUE)
{
return;
}
pRP->nodeID = TOS_LOCAL_ADDRESS;
if (call SendMsg.send(TOS_BCAST_ADDR, length, &routeMsg) ==
SUCCESS)
{
atomic
{
call Leds.redToggle();
}
dbg(DBG_TEMP, "MHGossipingPSM - Advertising presence\n");
atomic sendRouteBusy = TRUE;
}
}
```

Fig 1: Route Advertisement Interface Command in Gossiping Protocol

```
command result_t RouteSelect.selectRoute(TOS_MsgPtr msg, uint8_t id)
{
MHMessage *pMHMsg = (MHMessage *) &msg->data[0];
if (pMHMsg->sendingNode != TOS_LOCAL_ADDRESS)
{
pMHMsg->sendingNode = TOS_LOCAL_ADDRESS;
pMHMsg->hopCount++;
}
if (TOS_LOCAL_ADDRESS == BASE_STATION_ADDRESS)
{
msg->addr = TOS_UART_ADDR;
}
else if (entries > 0)
{
uint16_t num = call Random.rand() % entries;
int i;
for (i = 0; i < entries; i++)
{
if (neighbourTable[num].nodeAlive == TRUE)
{
msg->addr = neighbourTable[num].nodeID;
break;
}
else
{
num++;
num = num % entries;
}
}
if (pMHMsg->hopCount >= MAX_HOP_COUNT)
{
dbg(DBG_TEMP, "MHGossipingPSM - Failed to select route\n");
return FAIL;
}
}
```

Fig 2: Route Selection or fail interface code in Gossiping Protocol

```
task void advertise()
{
uint16_t lowestStrength;
RoutePacket *pRP = (RoutePacket *) &routeMsg.data[0];
uint8_t length = sizeof(RoutePacket);
if (sendRouteBusy == TRUE)
{
dbg(DBG_TEMP,"SORRY NO ROUTE ");
return FAIL;
}
pRP->nodeID = TOS_LOCAL_ADDRESS;
if ( call SendMsg.send(TOS_BCAST_ADDR , length++, &routeMsg) ==
SUCCESS)
{
atomic
{
call Leds.redToggle();
}
dbg(DBG_TEMP, "MHGossipingPSM - Advertising presence here now
\n",pRP->nodeID);
atomic sendRouteBusy = TRUE;
}
}
```

Fig 3: Route Advertisement Interface Command in Modified Gossiping Protocol

```
command result_t RouteSelect.selectRoute(TOS_MsgPtr msg, uint8_t id)
{
MHMessage *pMHMsg = (MHMessage *) &msg->data[0];
if (pMHMsg->sendingNode == TOS_BCAST_ADDR)
{
pMHMsg->sendingNode = TOS_BCAST_ADDR;
for(pMHMsg->hopCount=0 ; pMHMsg->hopCount <= MAX_HOP_COUNT; pMHMsg->hopCount++ )
{
dbg(DBG_TEMP,"hopcount increaseing := ",pMHMsg->hopCount++);
}
}
if (TOS_LOCAL_ADDRESS == BASE_STATION_ADDRESS)
{

dbg(DBG_TEMP,"DESTINATION:=" );
dbg(DBG_TEMP, "MHGossipingPSM - Failed to select route - hop count exceeded\n");
//return FAIL;

msg->addr = TOS_UART_ADDR;
}
```

Fig 4: Route Selection or fail interface code in Modified Gossiping Protocol

4. GOSSIPING PROTOCOL

Gossiping routing protocol is controlled form of flooding routing protocol [13]. Gossiping is proposed to overcome some critical problems of the flooding. In the gossiping scheme, it broadcast the message to its neighbor shows in Fig 5 and then randomly selects one node from the neighbor table as source to send packets. This improves the problem of heavy packet overhead and reduces the number of packet retransmission and consumes less energy than flooding protocol. However, it may cause other problems, the long packet delay which may cause of packet loss and it suffers from latency caused by data propagation. Because the sender randomly selects the subset of the result in a router neighbors to transmit data, the selected sensors may result farther than the shortest path between the sender and the sink. Hence, this may extend the packet delay time.

4.1 Network Topology

Gossiping, instead of the broadcasting each packet to all neighbors the packet is sent to a single neighbor chosen at random from a neighbor table. Having received the packet the neighbor chooses another random node to send to. This can include the node which sent the packet .This continues until the packet reaches its destination or the maximum hop the packet is exceeded.

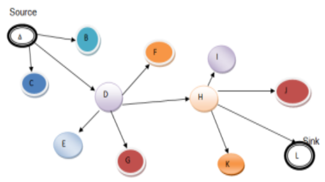


Fig.5. The Gossiping protocol. (1) Node A starts advertising its data to all of its neighbors. (2) Node D responds by sending a request to node A. (3) after receiving the request, node A sends the data. (4) Node D again sends advertisement out to its neighbors.

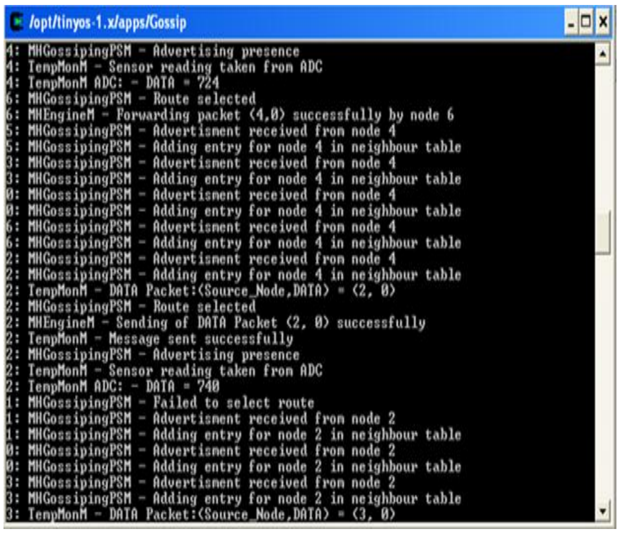
5. MODIFIED GOSSIPING PROTOCOL

In Gossiping routing protocol one nodes broadcast massage to its neighbour node according to route packet length. When timer is fired then previous length is used by Gossiping routing protocol where always the neighbour selecetion path range is same. It takes more hopcount to reach packet at destination. But in Modified Gossiping we have implemented a new concept, length of route incremants according to every time fired. Route of packet is increasing with increasing of length when data packet transferred. It takes minimum hop to reach packet at distination. We have calculated energy consumption and packet transmission in both routing protocol.

Gossiping routing protocol runs according to maximum hopcount i.e. when the hop reaches maximum hopcount then it is assumed that data reaches to the destination but there is still remains one question wheather data reached at distination or not ? But to overcome the situation we have implimented a new concept of node destination. In Modified Gossiping we fix destination node and at runtime simulation untill it will find destination node and when node reached at destination system automatically generates “Failed to route select- Hop count Exceeded” because it reached at destination and again it will broadcast for another destination.

6. SIMULATION AND RESULTS

Gossiping uses broadcasting with multi-hop delivery to send the packets to the network which will follow the best possible routes to its destination until the hop count exceed. But in Modified Gossiping destination node can fix accordingly. Fig. 6 and Fig. 9 nodes are broadcasting massage to the neighbor that selects the route. In Modified Gossiping we used the concept of length, it increases till destination node finds. In Fig. 7 shows that when hop count exceeds the massage received at sink and in TinyOS the result “Failed to route select” will be seen at runtime simulation if massage not received at sink. But in Fig. 10 it shows when packets reaches at destination node then no need for packet broadcast and it shows “Failed to route select- Hop count Exceeded”. The graphical display of Gossiping protocol showed nodes periodically broadcasting packets as expected where the blue circles are indicate packet broadcast. The TinyViz window is shown in Fig. 8, where purple lines are cot continuous, indicates finding of route to the destination is not efficient.



```
4: MHGossipingPSM - Advertising presence
4: TempMonM - Sensor reading taken from ADC
4: TempMonM ADC: - DATA = 724
6: MHGossipingPSM - Route selected
6: MHEngineM - Forwarding packet (4,0) successfully by node 6
5: MHGossipingPSM - Advertisment received from node 4
5: MHGossipingPSM - Adding entry for node 4 in neighbour table
3: MHGossipingPSM - Advertisment received from node 4
3: MHGossipingPSM - Adding entry for node 4 in neighbour table
0: MHGossipingPSM - Advertisment received from node 4
0: MHGossipingPSM - Adding entry for node 4 in neighbour table
6: MHGossipingPSM - Advertisment received from node 4
6: MHGossipingPSM - Adding entry for node 4 in neighbour table
2: MHGossipingPSM - Advertisment received from node 4
2: MHGossipingPSM - Adding entry for node 4 in neighbour table
2: TempMonM - DATA Packet:(Source_Node,DATA) = (2, 0)
2: MHGossipingPSM - Route selected
2: MHEngineM - Sending of DATA Packet (2, 0) successfully
2: TempMonM - Message sent successfully
2: MHGossipingPSM - Advertising presence
2: TempMonM - Sensor reading taken from ADC
2: TempMonM ADC: - DATA = 740
1: MHGossipingPSM - Failed to select route
1: MHGossipingPSM - Advertisment received from node 2
1: MHGossipingPSM - Adding entry for node 2 in neighbour table
0: MHGossipingPSM - Advertisment received from node 2
0: MHGossipingPSM - Adding entry for node 2 in neighbour table
3: MHGossipingPSM - Advertisment received from node 2
3: MHGossipingPSM - Adding entry for node 2 in neighbour table
3: TempMonM - DATA Packet:(Source_Node,DATA) = (3, 0)
```

Fig.6 Massage broadcasting with multi-hop delivery in Gossiping

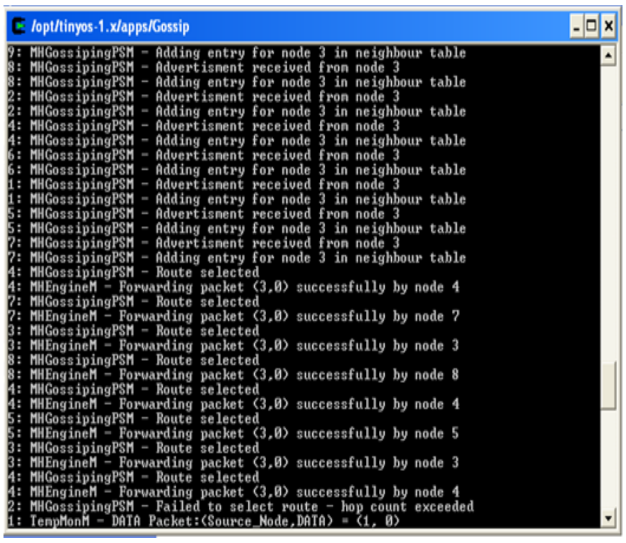

```
/opt/tinyos-1.x/apps/Gossip
9: MHGossipingPSM - Adding entry for node 3 in neighbour table
8: MHGossipingPSM - Advertisment received from node 3
8: MHGossipingPSM - Adding entry for node 3 in neighbour table
2: MHGossipingPSM - Advertisment received from node 3
2: MHGossipingPSM - Adding entry for node 3 in neighbour table
4: MHGossipingPSM - Advertisment received from node 3
4: MHGossipingPSM - Adding entry for node 3 in neighbour table
6: MHGossipingPSM - Advertisment received from node 3
6: MHGossipingPSM - Adding entry for node 3 in neighbour table
1: MHGossipingPSM - Advertisment received from node 3
1: MHGossipingPSM - Adding entry for node 3 in neighbour table
5: MHGossipingPSM - Advertisment received from node 3
5: MHGossipingPSM - Adding entry for node 3 in neighbour table
7: MHGossipingPSM - Advertisment received from node 3
7: MHGossipingPSM - Adding entry for node 3 in neighbour table
4: MHGossipingPSM - Route selected
4: MHEngineM - Forwarding packet (3,0) successfully by node 4
7: MHGossipingPSM - Route selected
7: MHEngineM - Forwarding packet (3,0) successfully by node 7
3: MHGossipingPSM - Route selected
3: MHEngineM - Forwarding packet (3,0) successfully by node 3
8: MHGossipingPSM - Route selected
8: MHEngineM - Forwarding packet (3,0) successfully by node 8
4: MHGossipingPSM - Route selected
4: MHEngineM - Forwarding packet (3,0) successfully by node 4
5: MHGossipingPSM - Route selected
5: MHEngineM - Forwarding packet (3,0) successfully by node 5
3: MHGossipingPSM - Route selected
3: MHEngineM - Forwarding packet (3,0) successfully by node 3
4: MHGossipingPSM - Route selected
4: MHEngineM - Forwarding packet (3,0) successfully by node 4
2: MHGossipingPSM - Failed to select route - hop count exceeded
1: TempMonM - DATA Packet:(Source_Node,DATA) = (1, 0)
```

Fig.7.Hop count exceeds in Gossiping

The protocol execution was looked at visually using TinyViz. The graphical display showed that packets were being sent to a random node within range. The TinyViz window of Modified Gossiping is shown in Fig 11, where blue circles indicate packet broadcast and the purple lines represent direct communication between nodes. Here purple lines are always continuous till destination node arrived.

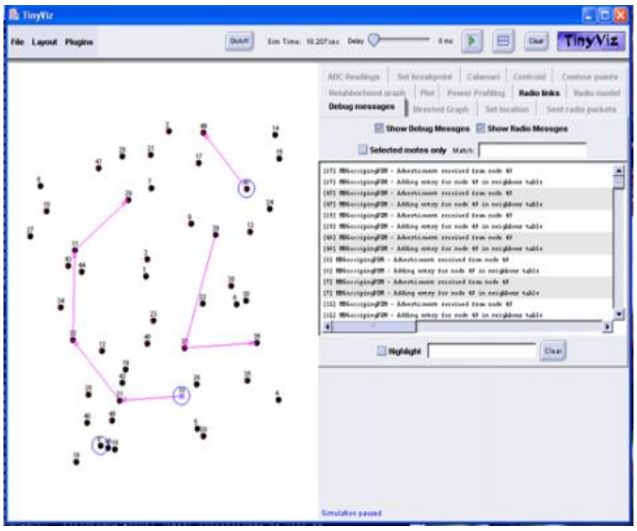


Fig. 8:Gossiping protocol displayed using TinyViz

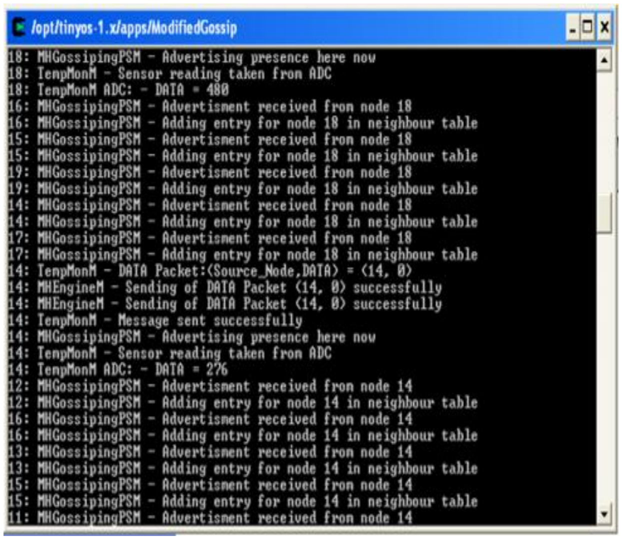

```
/opt/tinyos-1.x/apps/ModifiedGossip
18: MHGossipingPSM - Advertising presence here now
18: TempMonM - Sensor reading taken from ADC
18: TempMonM ADC: - DATA = 480
16: MHGossipingPSM - Advertisment received from node 18
16: MHGossipingPSM - Adding entry for node 18 in neighbour table
15: MHGossipingPSM - Advertisment received from node 18
15: MHGossipingPSM - Adding entry for node 18 in neighbour table
19: MHGossipingPSM - Advertisment received from node 18
19: MHGossipingPSM - Adding entry for node 18 in neighbour table
14: MHGossipingPSM - Advertisment received from node 18
14: MHGossipingPSM - Adding entry for node 18 in neighbour table
17: MHGossipingPSM - Advertisment received from node 18
17: MHGossipingPSM - Adding entry for node 18 in neighbour table
14: TempMonM - DATA Packet:(Source_Node,DATA) = (14, 0)
14: MHEngineM - Sending of DATA Packet (14, 0) successfully
14: MHEngineM - Sending of DATA Packet (14, 0) successfully
14: TempMonM - Message sent successfully
14: MHGossipingPSM - Advertising presence here now
14: TempMonM - Sensor reading taken from ADC
14: TempMonM ADC: - DATA = 276
12: MHGossipingPSM - Advertisment received from node 14
12: MHGossipingPSM - Adding entry for node 14 in neighbour table
16: MHGossipingPSM - Advertisment received from node 14
16: MHGossipingPSM - Adding entry for node 14 in neighbour table
13: MHGossipingPSM - Advertisment received from node 14
13: MHGossipingPSM - Adding entry for node 14 in neighbour table
15: MHGossipingPSM - Advertisment received from node 14
15: MHGossipingPSM - Adding entry for node 14 in neighbour table
11: MHGossipingPSM - Advertisment received from node 14
```

Fig.9. Advertising massage and route select in Modified Gossiping

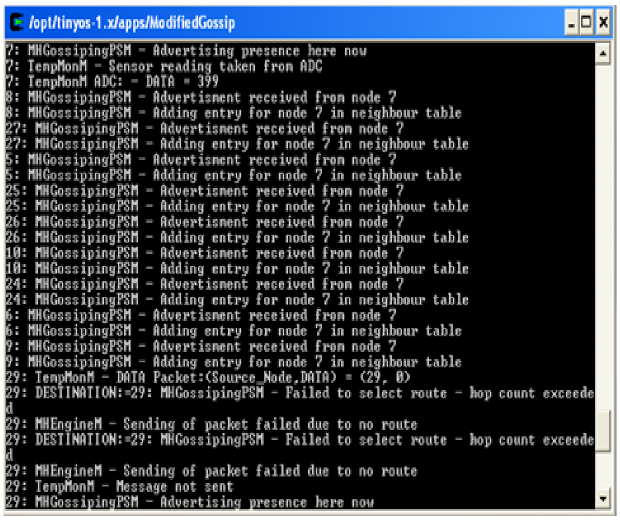

```
/opt/tinyos-1.x/apps/ModifiedGossip
7: MHGossipingPSM - Advertising presence here now
7: TempMonM - Sensor reading taken from ADC
7: TempMonM ADC: - DATA = 399
8: MHGossipingPSM - Advertisment received from node 7
8: MHGossipingPSM - Adding entry for node 7 in neighbour table
27: MHGossipingPSM - Advertisment received from node 7
27: MHGossipingPSM - Adding entry for node 7 in neighbour table
5: MHGossipingPSM - Advertisment received from node 7
5: MHGossipingPSM - Adding entry for node 7 in neighbour table
25: MHGossipingPSM - Advertisment received from node 7
25: MHGossipingPSM - Adding entry for node 7 in neighbour table
26: MHGossipingPSM - Advertisment received from node 7
26: MHGossipingPSM - Adding entry for node 7 in neighbour table
10: MHGossipingPSM - Advertisment received from node 7
10: MHGossipingPSM - Adding entry for node 7 in neighbour table
24: MHGossipingPSM - Advertisment received from node 7
24: MHGossipingPSM - Adding entry for node 7 in neighbour table
6: MHGossipingPSM - Advertisment received from node 7
6: MHGossipingPSM - Adding entry for node 7 in neighbour table
9: MHGossipingPSM - Advertisment received from node 7
9: MHGossipingPSM - Adding entry for node 7 in neighbour table
29: TempMonM - DATA Packet:(Source_Node,DATA) = (29, 0)
29: DESTINATION:=29: MHGossipingPSM - Failed to select route - hop count exceede
d
29: MHEngineM - Sending of packet failed due to no route
29: DESTINATION:=29: MHGossipingPSM - Failed to select route - hop count exceede
d
29: MHEngineM - Sending of packet failed due to no route
29: TempMonM - Message not sent
29: MHGossipingPSM - Advertising presence here now
```

Fig.10. Source massage sends successfully at Destination in Modified Gossiping

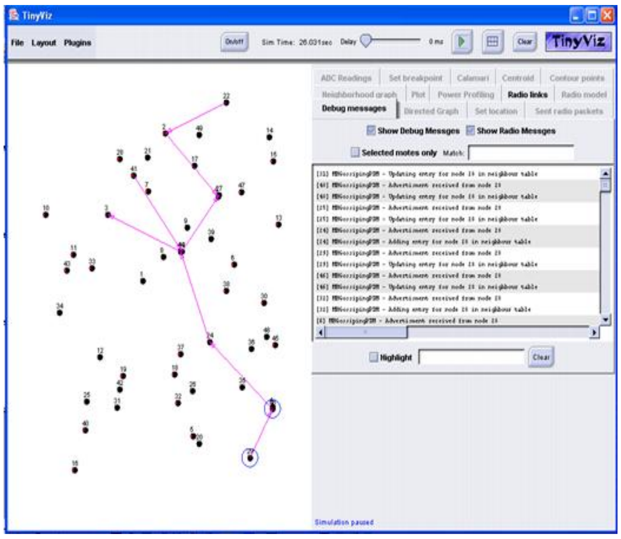


Fig 11: Modified Gossiping protocol displayed using TinyViz

7. SIMULATION RESULT

Simulation of Modified Gossip and Gossip protocol has been done by TinyOS and results are given in different table. In Table 1 the compares the number hopcount with changing of nodes and implies modified Gossip takes less number of hopcount to send data to the base station. Table 2 shows energy consumption with increasing of hopcount, implies modified Gossip consumes less power to send data. Table 3 shows of number of broadcast in our protocol are less compare to Gossip protocol. Table 4 implies energy consumption with increasing of node where modified Gossip performs better than Gossip. Table 5 and Table 6 demonstrate the Packet Delivery with and without delay respectively.

Node	HopCount(Gossip)	HopCount(Modified Gossip)
10	11	6
15	17	13
20	21	18
25	22	18
30	23	21
35	24	22
40	26	25
45	27	26
50	28	25

Table 1: Comparison of Hopcount with increasing of node

HopCount	Energy(Gossip) (nj)	Energy(Modified Gossip) (nj)
30	11	4.8
35	13	7
40	24	9.8
45	27	14
50	29	15.2
55	33	16.1
60	35	19.4
65	37	21
70	38	26.1

Table 2: Comparison of energy with increasing of hopcount

Node	Number of Broadcast(Gossip)	Number of Broadcast(Modified Gossip)
30	22	17
35	29	25
40	33	28
45	37	32
50	40	36

Table 3: Comparison of number of broadcast with increasing of node

Node	Energy(Gossip)(nj)	Energy(Modified Gossip)(nj)
30	15.4	12.6
35	20.3	17.5
40	23.1	19.6
45	25.9	22.4
50	28	25.2

Table 4: Comparison of energy consumption with increasing of node

Time	Packet Delivery with delay(Gossip)	Packet Delivery with delay(Modified Gossip)
10	14	11
15	21	13
20	25	19
25	29	23
30	36	30
35	43	37
40	49	43
45	55	48
50	61	54

Table 5: Comparison of Packet Delivery (with delay) in time (Sec.)

Time	Packet Delivery without delay(Gossip)	Packet Delivery Without Delay(Modified Gossip)
10	15	13
15	23	19
20	31	26
25	37	31
30	43	35
35	49	39
40	55	46
45	60	51
50	64	55

Table 6: Comparison of Packet Delivery (without delay) in time (Sec.)

8. PERFORMANCE ANALYSIS

We present the performance analysis of both Gossiping and Modified Gossiping protocol by using MATLAB and TOSSIM [7]. Implementation of both protocols has been written in nesC programming language in TOSSIM. Here heuristic simulation done and compared the simulated result in various way such as (1) Total Number of packets transferred to reach the sink node for an event with respect to both time (2) Energy consumption of the sensor network with varying both hopcount and sensor nodes in the sensor network. In Fig. 12 and Fig. 13 show the comparison of energy consumption with respect to node and hopcount respectively and in both the figure shows Modified Gossip extending network lifetime than Gossip. In Fig. 14 explains node involvement with varying hopcount, is less in Modified Gossip is the improvement on Gossip protocol under same network topology. Fig. 15 and Fig. 16 shows packet transmission by the different protocol with time (sec.) considering without delay and delay respectively and from simulated result it has been concluded Modified Gossiping performing better than Gossiping**.**

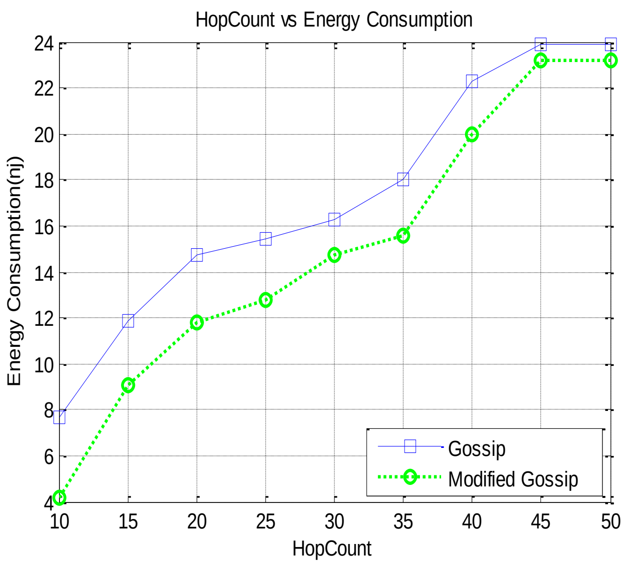


Fig 12: Energy Consumption of network with hopcount

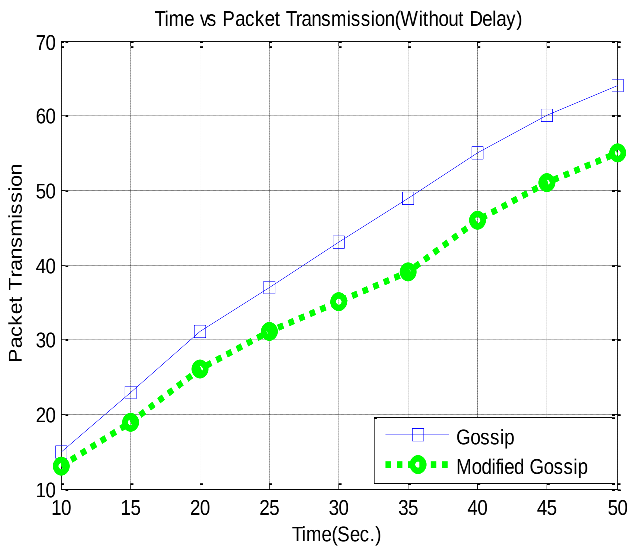


Fig 15: Packet transmission with Time (Sec.) in without delay

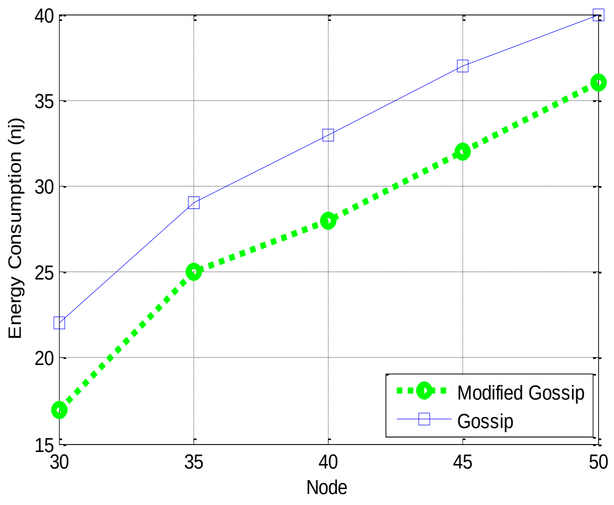


Fig 13: Energy Consumption of network with number of sensor nodes

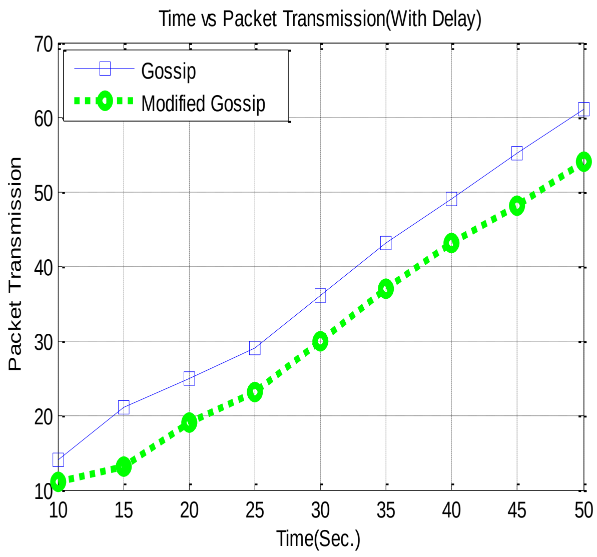


Fig 16: Packet transmission with Time (Sec.) in with delay

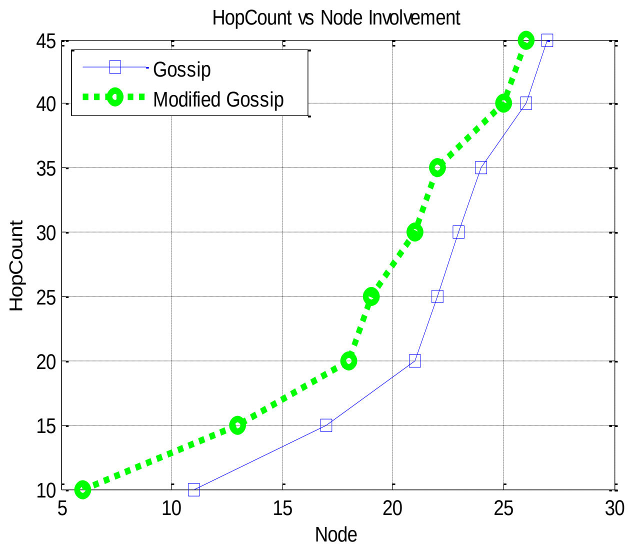


Fig 14: Hopcount with number of node involvement

9. CONCLUSION

The Gossiping and Modified Gossiping protocols were implemented in TinyOS with same topology. The basic evaluation of these protocols was carried out and different phases of comparison results has been made and showed by different tables and graphs. During the design and implementation of the protocols it was clear that performance gains by Modified Gossip is better than Gossip. Thus Modified Gossip prolong network lifetime than Gossip protocol.

10. REFERENCES

[1] Bhandari.V. "A study of gossip as a routing mechanism in Mobile Adhoc Network". CS 598rhk Project Report.

[2] Hass. Z.J., Halpern. Y.J.,Li. L.: Gossip based adhoc routing, In Proc. of INFOCOM 2002, (2002).

[3] Helal. A., Mann. W., Elzabadani. H., King. J., Kaddourah. Y., Jansen. E. 2005."Gator Tech Smart House: A Programmable Pervasive Space", IEEE Computer Magazine, 383, pp 64-74.

[4] McKelvin. M. L., Williams. M. L., Berry. N. M. 2005. “Integrated Radio Frequency Identification and Wireless Sensor Network Architecture for Automated Inventory Management and Tracking Applications”, RichardTapia Celebration of Diversity in Computing Conference, pp. 44-47.

[5] Huang. X. M., Ma. J., Leblanc. L. E. 2004.” Wireless Sensor Network for Streetlight Monitoring and Control”, in Proc. of the SPIE Digital Wireless Communication Conference, vol. 5440.

[6] Haas. Z. J., Halpern. J. Y. 2006. “Gossip Based Ad Hoc Routing”, IEEE Transactions on networking, 14(3).

[7] Frikha. M., Ghandour. F. 2007. “Implementation and Performance Evaluation of an Energy Constraint Routing Protocol for MANET”, 3rd International Conference of Telecommunications (AICT’07).

[8] Perkins. C. E., Royer. E. M., Das. S. 2000. “Ad-Hoc On Demand Distance Vector Routing (AODV)”, draft-ietfmanet-aodv- 05.txt.

[9] Johnson. D. B., Maltz. D. A., Hu. Y.C. 2003 “DSR for Mobile Ad Hoc Network”, Internet-Draft, draft-ietfmanet-drs-09.txt.

[10] Perkins. C. E., Bhagwat. P. 1994. “Highly Dynamic Destination Sequenced Distance Vector Routing (DVDV) for Mobile Computers”, Proceeding of ACM SIGCOMM.

[11] Levis. P., Lee. N. 2003. “TOSSIM: A Simulator for TinyOS Networks”, Included with the TinyOS 1.1.0 software.

[12] Gay. D., Levis. P., Behren. R., Welsh. M., Brewer. E., Culler. D. 2003. “The nesC Language: A Holistic Approach to Networked Embedded Systems”. In *Proceedings of Programming Language Design and Implementation (PLDI)*.

[13] Heinzelman. W. R., Kulik. J., Balakrishnan. H. 1999. “ Adaptive Protocols for Information Dissemination in Wireless Sensor Networks”, in Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking, pp. 174-185.