Surveillance and Tracking Eleplant Network Path in Wireless Sensor Network

Vanitha P.S, Nithya P.S
PG Scholar NIELIT Calicut, PG Scholar UIT Coimbatore

Abstract— **Wireless sensor networks (WSN) have been considered as promising tools for many locations dependent applications such as area surveillance, search and rescue, mobile tracking, animal tracking etc. In addition, the geographic information of sensor nodes can be critical for improving network management, topology planning, packet routing and security. In wireless sensor network the important function of sensor nodes is to collect and forward data to destination. It is very important to know about the location of collected data. This kind of information can be obtained using localization technique in WSN. Monitoring the elephant Localization and Direction of arrival estimation is one of the crucial examining focuses in Wireless Sensor Network. In previous work, the direction of arrival estimation is done by using limited coverage area and minimum number of sensor nodes. Localization in which the geographical positions of sensors are determined by increasing the distance of sensor coverage area and increasing the number of sensor nodes. Each sensor node can transmit and receive the information based on different topology and event is detected by packet generation algorithm. The additional external factors affecting sensor network coverage are distance between the source and the nodes and also track the movement of the elephant based on event tracking algorithm. Finally the network path was found using the location of the sensor node in the field. In future our work focusing on experimental analysis carried out by real time sensor nodes, data aggregation and also for the multiple object tracking.**

Keywords — localization, event tracking, network path identification, GNDA.

I. INTRODUCTION

A wireless sensor network (WSN) of spatially distributed autonomous sensors to monitor physical or environmental conditions, such as temperature, sound, pressure, etc. and to cooperatively pass their data through the network to a main location. The more modern networks are bi-directional, also enabling control of sensor activity. The development of wireless sensor networks was motivated by military applications such as battlefield surveillance; today such networks are used in many industrial and consumer applications, such asindustrial process monitoring and control, machine health monitoring, animal tracking and so on.

II. SENSOR NETWORK FOR ANIMAL HABITANT:

Wireless Sensor Networks (WSNs) are being deployed in very diverse application scenarios, including rural and forest environments. In these particular contexts, specimen protection and conservation is a challenge, especially in natural reserves, dangerous locations or hot spots of these reserves. The WSN based system for generic target (animal) tracking in the surrounding area of wildlife passages built to establish safe ways for animals to cross transportation infrastructures. In addition, it allows target identification through the use of video sensors connected to strategically deployed nodes. This deployment is designed on the basis of the IEEE 802.15.4 standard, but it increases the lifetime of the nodes through an appropriate scheduling. The system has been evaluated for the particular scenario of wildlife monitoring in passages across roads. For this purpose, different schemes have been simulated in order to find the most appropriate network operational parameters. Moreover, a novel prototype, provided with motion detector sensors, has also been developed and its design feasibility demonstrated. Habitat and environmental monitoring represent a class of sensor network applications with enormous potential benefits for scientific communities and society as a whole. Incrementing natural spaces with numerous networked micro sensors can enable long-term data collection at scales and resolutions that are difficult, if not impossible, to obtain otherwise.

The intimate connection with its immediate physical environment allows each sensor to provide localized measurements and detailed information that is hard to obtain through traditional instrumentation. The integration of local processing and storage allows sensor nodes to perform complex filtering and triggering functions, as well as to apply application specific or sensor-specific data compression algorithms. The ability to communicate not only allows information and control to be communicated across the network of nodes, but nodes to cooperate in performing more complex tasks, like statistical sampling, data aggregation, and system health and status monitoring. In this phase the sensor node deployment and implementation of the essential network services, including power management, communications, packet transformation and tracking of elephant network path are analyzed. Among the most promising sensors for animal habitant monitoring application were microphones, motion sensors, and accelerometers.

III. EVENT -DRIVEN SYSTEM DESIGN:

The system architecture of WSN is based on event driven system for the animal habitant. In our system, the motes prepare for tracking by going through an initialization process. This process is used to synchronize the motes, set up communication routes, and configure the system with the correct control parameters. The initialization process proceeds in a sequence of phases and the transition between phases is time-driven The duration of each phase is a control parameter that can be dynamically configured by the base station.

i)Sensor Deployment
ii)Event Detection
iii)Event Tracking
iv)Network Path identification

3.1 SENSOR DEPLOYMENT

A) Basic Initialization:

We observe that three functions in our system need system-wide broadcast: time synchronization, network backbone creation and system-wide reconfiguration. These functions can be isolated into three different modules that perform separately. However, such a design would not be bandwidth and energy efficient due to the multiple flooding phases required. Instead, we use a unique application specific design to perform these operations simultaneously in one flooding.

B) Time Synchronization:

System initialization begins with time synchronization. Several schemes proposed recently are able to achieve a high synchronization precision; however they do not match well with our system requirements. GPS-based schemes typically achieve persistent synchronization with a precision of about 200 ns. However, GPS devices are expensive and bulky. The reference broadcast scheme (RBS) proposed in maintains information relating the phase and frequency of each pair of clocks in the neighborhood of a node. The relation is then used to perform time conversion when comparing the timestamps of two different nodes. While RBS achieves a precision of about 1 μs, the message overhead in maintaining the neighborhood information is high and may not be energy-efficient in large systems.

The fine-grained clock synchronization achieved by costly periodic beacon exchanges may not be suitable for the energy-constrained surveillance system. Moreover continuous adjustment through beaconing in these solutions defeats our purpose of stealthiest. In our system, we value energy-efficiency and stealthiest above high synchronization precision. Therefore, we use a lightweight scheme that synchronizes the motes only during initialization phase, using a synchronization beacon broadcast by the base station at the beginning of each initialization cycle. Since the under-lying MAC layer provided by OMNet++ does not guarantee re-liable delivery, the base station retransmits the synchronization beacon multiple times. Receivers take the timestamp from the beacon plus a transmission delay as their own local time. The synchronization beacons are propagated across the network through limited flooding with timestamp values reassigned at intermediate motes immediately prior to transmission. To satisfy the stealthiest requirement, we confine time synchronization within the initialization phase.

3.2 EVENT DETECTION:

a) Packet Generation Algorithm:

When an event occur the node in the coverage area will generate the packet and pass the packet to the nearby node. The generation of the packet includes the event number, source IP address, destination IP address, router (intermediate node details), and message id. By using AODV protocol the routing table is constructed by sending Hello message to the neighbours in the network. Additional parameter is updated in the message field that event has occurred in the particular location.

Packets: A packet is defined by a (port; header) tuple, where the port denotes a packet's position in the network at any time instant (each physical port in the network is assigned a unique number).

Switch: A switch transfer function, T , models a network device, such as a switch or router. Each network device contains a set of forwarding rules (e.g., the forwarding table) that determine how packets are processed. An arriving packet is associated with exactly one rule by matching it against each rule in descending order of priority, and is dropped if none match.

Rules: A rule generates a list of one or more output packets, corresponding to the output port(s) the packet is sent to; and defines how packet fields are modified.

Packet Generation Algorithm

```
function network(packets,
switches, T) for pk0 ∈ packets do

T ←find_switch(pk0.p,
switches) for pk1 ∈ T (pk0 )
do

if pk1.p ∈ Edge Ports
then #Reached edge
record(pk1 )
else
#Find next hop
network(T(pk1 ),
switches,T)
```

The packet is generates when event encounter in the particular location , the event is stored in the routing table as an entry with time specification and the next hop detail is discover. The sensor node contains LQueue for storing the incoming packet and wait till the output port is available to transmit the packet. Once the network path is free it transmit the data to nearby neighbour and update it routing table entry and mark the packet is successfully deliver to neighbour node.

3.3 EVENT TRACKING:

WSNs are composed of a large number of sensor nodes therefore; an algorithm for a WSN is implicitly a distributed algorithm. In WSNs the scarcest resource is energy, and one of the most energy-expensive operations is data transmission. For this reason, algorithmic research in WSN mostly focuses on the study and design of energy aware algorithms for data transmission from the sensor nodes to the bases stations. Data transmission is usually multi-hop (from node to node, towards the base stations), due to the polynomial growth in the energy-cost of radio transmission with respect to the transmission distance.

a) Neighbour Discovery:

After the basic initialization phase, the motes make a transition to a neighbor discovery phase. Motes notify their neighbors by locally broadcasting HELLO message. In the HELLO message, a sender sends its identifier, its status indicating whether it is a sentry or not, and the number of sentries that are currently covering it. The sender also identifies the sentry mote it reports to if it is covered by at least one sentry. This local information is used to build a neighborhood table at each mote, and forms the basis for sentry selection.

b) Proposed Algorithhm

The performance of network increases by considering good nodes into the account. Categorizations of good and bad nodes depend upon signal strength, flow capacity of nodes. Moreover, how fast each node can receive the complete information. Proposed approach is analyzed by using AODV routing protocol. Implementation of routing protocols is developed by using Linux-5.3 and omnet++. In wireless mode, we have used IEEE 802.11a for this approach.

Adhoc on demand distance vector routing protocol is an on demand routing protocol. In this protocol, routing discovery process is initiated when route is required. In this protocol, If source node or intermediate node moves then the moving nodes realizes link failure and send this link failure notification to its upstream neighbors and so on till it reaches to source. So, source can reinitiate route discovery if needed. Fig 3.1 presents process flow of AODV. Fig.3.1 considers three actors such as source, destination and intermediate node in which all processes are defined and eleven activities are designed for this same process. It has been analyzed that performance of network gradually decreases in certain following cases:

- If transmission range of node is larger than transmission range of network.
- If neighbor node is flooding unnecessary RREQ messages to other nodes.
- If time is high to reach hello messages between two nodes.

- If packet dropping ratio of a neighbor node is maximum.

All above four cases shows that presence of any one case decreases performance of on demand routing protocol. In this approach, initially all nodes maintain their own transmission range. It has been assumed the transmission range of the network is 250 meters. Now, transmission range (NTr) of each node present in the network with the total transmission of network (TTrN) of the node is compared. Determination of transmission power is required to send a message between node n and its neighbor n1. It can be measured by calculating the received power of hello message. When node n receives hello messages from a neighbor node n1, it can estimate the minimum power level needed to reach n1 by comparing the received power of hello message with maximum transmit power.This approach is enhanced by adding parameters in the neighbor table such as flow capacity, signal strength. Reaching time of hello messages between node and its neighbor is calculated. Address of node is stored into the neighbor table based on their transmission range. If (NTr > TTrN), then adjust energy of this node accordingly, otherwise calculate signal strength by using equation (1). If threshold value is maximum then evaluate position of node and also set timer for the same. Further work is preceded by calculating the flow capacity of a node as mentioned in equation (2). If flow capacity of a node is good then store address of a node otherwise remove address of the node from routing table. Suggested algorithm is an optimal solution for finding good nodes. Categorization of nodes is based on performance metrics such as transmission range and power of node, signal strength, capacity of node for high packet forwarding and relative position of node. Neighbor routing table maintains address of node for maintaining record of the entire nodes.

These stored nodes are used for data transmission and forwarding. This approach minimizes energy consumption of node and increases its battery life. Thus any node can forward data to other node if they exist within the range of 250 meters. And location or relative position of node can be verified by using routing table.

Nodes	**60-100**
Simulation Time	**10 sec**
MAC layer	**IEEE 802.11**
Packet Size	**512**
Pause time	**0-100 ns**
Initial energy	**50**
Trans. range	**250 m**
CS range	**550m**
Tx thres. power	**0.281838**
Tx & Rx power	**0.173, 0.05**

SYSTEM PARAMETERS

CALCULATION 1:

Signal Strength of a node is computed by using well known formula which is as follows

$$\text{Transmitter}_{\text{signal strength}} = \begin{cases} S_H - \left\{ \dfrac{S_H - S_{\text{threshold}} * T}{e} \right\} & \text{if farther } (T>e) \\ S_H & \text{closer } (T<e) \\ S_{\text{thresh}} & \text{Otherwise} \end{cases}$$

Where ^{S}H signal strength of hello message and T is ^{is} the time period between two successive hello packets and e is the link connectivity between i and j.

CALCULATION 2:

Assume a graph G (V, E) . The capacity of directed edge is denoted as C_{ij} source s and destination d. F is assumed as a flow in G where E belongs to edge (i,j).

If for all (i, j) Є E; 0<= F_{ij} <= $C_{ij;}$ s.t.

$$\sum_{j:(s,j)\in E} F_{sj} - \sum_{i:(i,s)\in E} F_{is} \Big|$$

Let F_{is} and F_{sj} be the counter of amount of bytes that flowed on the link (i, j) upto time t in packets.The process flow of the algorithm, the nodes in the network are initialized with system parameters as shown in the tabular column.

Thus, If signal strength range is negligible then discard this node and delete entry of this node from the neighbor table. Otherwise calculate flow capacity of a node by considering equation (2). Based on flow capacity and packet delivery ratio, good neighbors are identified.Complexity Computation between AODV and GNDA: By adding new parameters into the routing table, suggested approach increases size of routing table.

Thus storage complexity of suggested approach is same as AODV i.e. O(N), where N is the total number of nodes present in the network. It slightly increases overhead by using hello messages but it provides good communication between source and destination as compared to AODV. Thus communication complexity of suggested algorithm in O (N). This approach was not suitable against impersonation attack and also its accuracy decreases in case of high mobility.

3.4 NETWORK PATH IDENTIFICATION:

a) Node Program Design

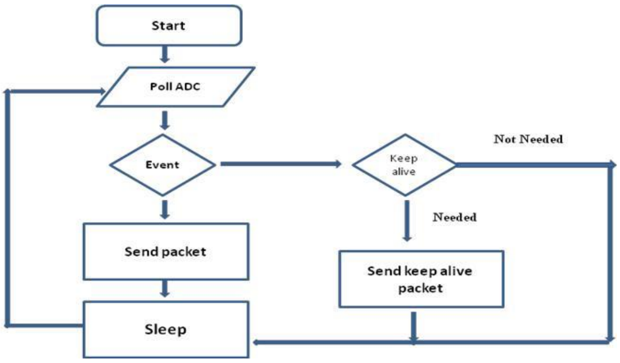
The software for each individual sensor node is a straightforward loop. Each involves monitoring the analog-to-digital converter port and sending packets to the base station if an event is detected. The program flow of the accelerometer and motion sensor nodes is depicted in Each program begins with a startup sequence, initializing needed variables, the analog-to-digital converter, and user button functionality. The startup sequence for the accelerometer also incorporates an initial accelerometer reading to determine the starting orientation.

Following these startup procedures, each node begins monitoring their respective sensors; deciding if an event needs to be generated based upon the sensor readings it acquires. If no event is detected, the node provide user feedback in the form of a blinking LED for easily verifiable functionality. Provided no event occurs for 60 seconds, the node will send a keep-alive packet to the base station, informing it that the mote is, in fact, still functional.

A user can also send this packet from the node manually by depressing the user button on the back of the node. If an event occurs, the node will reset the keep-alive timer, ensuring that the mesh network is not cluttered with spurious data. Additionally, it will send a packet to the base station containing the requisite identifying information before returning to the beginning of the program.

3.4.1NETWORK PATH IDENTIFICATION ALGORITHM:

The event path is identified by using the packet generation of the sensor node. The packet is immediately forwarded to the nearby and reached the cluster head. The cluster head store all detail about the coverage area (location of the node) which send the network path to the base station



NODE PROGRAM DESIN

The event tracking algorithm is run by every node in the network which will give hop details.

The parameter used for tracking the event based on the location of the node, app_id, packet, Event number, Time, Router id, input/output port id, Message id, Hop count.

1. Get the input packet from the packet generation algorithm and initialize the event number, hop count and timer for the nodes in the sensor networks.

2. The app_ id is created for the packet by the source node.

3. Depending on good neighbour detection algorithm the routing table is constructed, choose the best neighbour depending on Signal Strength, Flow capacity and Traffic less routing.

4. Route the packet in Ri (i=1,2....n) with the port number P_in , P_out.

5. If the packet is received by the neighbour, the msg_id is generated and send as ACK to the source node.

6. If the Single Transmission Finished then hops count is incremented by 1.

7. Repeated the step from 3 to 6, till destination reached.

8. Once the destination reached the app_id created by the destination node and the whole network path with hop count is displayed as output.

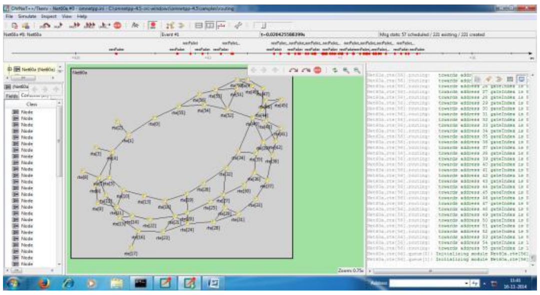
3.4.2 Network Path Identification Algorithm:

```
    trace_path (app_id,packets,msg_id)
      event_no=1;   timer=0.001   for
      pk1 ∈ packets do
          A← gnda(source_id,pk1,dest_id )
              if inter_id ∈ des_id then
                 if Q_in!=Full && Q_out==Full then
                       Q_in(pk1) ← P_in
                       Generate Qin_id
                       event_no++, time++
                 else
                       Q_out(pk1) ←P_out

                  Generate Qout_id and route_ id
                  event_no++, time++
            if pk1 ∈ route_id then
                #Reached hop details (intermediate node)
                 Generate msg_id, increment hop_count
                 record(pk1 )
                  event_no++, time++
                 #Repeat till destination.
       # Print the whole path from source to destination.
```

The network path identification table is created after the execution of GNDA and trace path algorithm. It traces the path from the event detection (where the event is detected) by generating the packet and the next hop detail is chosen from the signal strength and flow capacity of the sensor node. The node with high efficiency acts as destination and the information is forwarded to that node meanwhile the event may move to the nearby node depending on the direction the other nodes in the network also generate packet, thus lets to duplication of the packet. The same event is detected by more than one node in the sensor network then data aggregation has to process. Depending upon the time of the event occur the duplication may be detected and message is passed to the efficient node in the network. The node with high signal strength and flow capacity makes the packet to reach the destination without any packet loss.

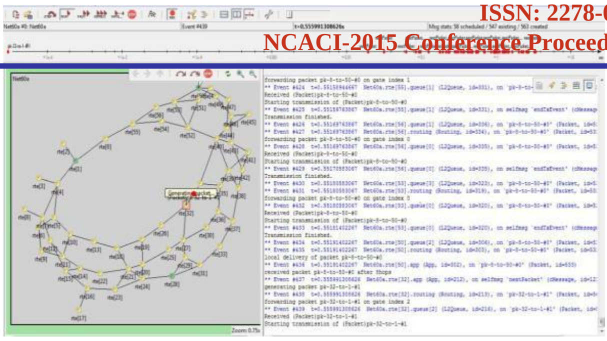
The simulation are showed below, the sensor nodes are deployed in the field.


OMNeT++/Tkenv - Net60a #0 - omnetpp.ini - C:\omnetpp-4.5-src-windows\omnetpp-4.5\samples\routing
File Simulate Inspect View Help
Net60a
Zoom: 0.75x
11:41
16-11-2014

Msg stats: 57 scheduled / 221 existing / 221 created

```
Net60a.rte[56].routing:   towards addr
Net60a.rte[56].routing:   towards addr
Net60a.rte[56].routing:   towards address 26 gateIndex is
Net60a.rte[56].routing:   towards address 27 gateIndex is
Net60a.rte[56].routing:   towards address 28 gateIndex is
Net60a.rte[56].routing:   towards address 29 gateIndex is
Net60a.rte[56].routing:   towards address 30 gateIndex is
Net60a.rte[56].routing:   towards address 31 gateIndex is
Net60a.rte[56].routing:   towards address 32 gateIndex is
Net60a.rte[56].routing:   towards address 33 gateIndex is
Net60a.rte[56].routing:   towards address 34 gateIndex is
Net60a.rte[56].routing:   towards address 35 gateIndex is
Net60a.rte[56].routing:   towards address 36 gateIndex is
Net60a.rte[56].routing:   towards address 37 gateIndex is
Net60a.rte[56].routing:   towards address 38 gateIndex is
Net60a.rte[56].routing:   towards address 39 gateIndex is
Net60a.rte[56].routing:   towards address 40 gateIndex is
Net60a.rte[56].routing:   towards address 41 gateIndex is
Net60a.rte[56].routing:   towards address 42 gateIndex is
Net60a.rte[56].routing:   towards address 43 gateIndex is
Net60a.rte[56].routing:   towards address 44 gateIndex is
Net60a.rte[56].routing:   towards address 45 gateIndex is
Net60a.rte[56].routing:   towards address 46 gateIndex is
Net60a.rte[56].routing:   towards address 47 gateIndex is
Net60a.rte[56].routing:   towards address 48 gateIndex is
Net60a.rte[56].routing:   towards address 49 gateIndex is
Net60a.rte[56].routing:   towards address 50 gateIndex is
Net60a.rte[56].routing:   towards address 51 gateIndex is
Net60a.rte[56].routing:   towards address 52 gateIndex is
Net60a.rte[56].routing:   towards address 53 gateIndex is
Net60a.rte[56].routing:   towards address 54 gateIndex is
Net60a.rte[56].routing:   towards address 55 gateIndex is
Net60a.rte[56].queue[1]: Initializing module Net60a.rte[56]
Net60a.rte[56].queue[1]: Initializing module Net60a.rte[56]
```

When the event occurs with the coverage area then, packet is generated and forwarded to the nearby node in the sensor field.

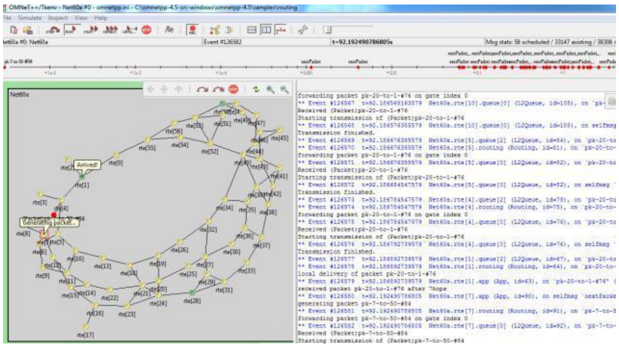


t=0.555991308626s

Msg stats: 58 scheduled / 547 existing / 563 created

```
forwarding packet pk-8-to-50-#0 on gate index 1
** Event #624  t=0.55158944667  Net60a.rte[55].queue[1] (L2Queue, id=331), on 'pk-8-to-
Received (Packet)pk-8-to-50-#0
Starting transmission of (Packet)pk-8-to-50-#0
** Event #625  t=0.55158763867  Net60a.rte[55].queue[1] (L2Queue, id=331), on selfmsg 'endTxEvent' (cMessag
Transmission finished.
** Event #626  t=0.55169763867  Net60a.rte[56].queue[1] (L2Queue, id=336), on 'pk-8-to-50-#0' (Packet, id=
** Event #627  t=0.55169763867  Net60a.rte[56].routing (Routing, id=334), on 'pk-8-to-50-#0' (Packet, id=53
forwarding packet pk-8-to-50-#0 on gate index 0
** Event #628  t=0.55169763867  Net60a.rte[56].queue[0] (L2Queue, id=335), on 'pk-8-to-50-#0' (Packet, id=
Received (Packet)pk-8-to-50-#0
Starting transmission of (Packet)pk-8-to-50-#0
** Event #629  t=0.55170583067  Net60a.rte[56].queue[0] (L2Queue, id=335), on selfmsg 'endTxEvent' (cMessag
Transmission finished.
** Event #630  t=0.55180583067  Net60a.rte[53].queue[3] (L2Queue, id=323), on 'pk-8-to-50-#0' (Packet, id=
** Event #631  t=0.55180583067  Net60a.rte[53].routing (Routing, id=319), on 'pk-8-to-50-#0' (Packet, id=53
forwarding packet pk-8-to-50-#0 on gate index 0
** Event #632  t=0.55180583067  Net60a.rte[53].queue[0] (L2Queue, id=320), on 'pk-8-to-50-#0' (Packet, id=
Received (Packet)pk-8-to-50-#0
Starting transmission of (Packet)pk-8-to-50-#0
** Event #633  t=0.55181402267  Net60a.rte[53].queue[0] (L2Queue, id=320), on selfmsg 'endTxEvent' (cMessag
Transmission finished.
** Event #634  t=0.55191402267  Net60a.rte[50].queue[2] (L2Queue, id=306), on 'pk-8-to-50-#0' (Packet, id=
** Event #635  t=0.55191402267  Net60a.rte[50].routing (Routing, id=303), on 'pk-8-to-50-#0' (Packet, id=53
local delivery of packet pk-8-to-50-#0
** Event #636  t=0.55191402267  Net60a.rte[50].app (App, id=302), on 'pk-8-to-50-#0' (Packet, id=533)
received packet pk-8-to-50-#0 after 5hops
** Event #637  t=0.555991308626  Net60a.rte[32].app (App, id=212), on selfmsg 'nextPacket' (cMessage, id=12
generating packet pk-32-to-1-#1
** Event #638  t=0.555991308626  Net60a.rte[32].routing (Routing, id=213), on 'pk-32-to-1-#1' (Packet, id=
forwarding packet pk-32-to-1-#1 on gate index 2
** Event #639  t=0.555991308626  Net60a.rte[32].queue[2] (L2Queue, id=216), on 'pk-32-to-1-#1' (Packet, id=
Received (Packet)pk-32-to-1-#1
Starting transmission of (Packet)pk-32-to-1-#1
```

The path of the object is traced with the help of generated packet and the information is collected by the cluster head.


OMNeT++/Tkenv - Net60a #0 - omnetpp.ini - C:\omnetpp-4.5-src-windows\omnetpp-4.5\samples\routing
File Simulate Inspect View Help
Net60a #0: Net60a
Net60a
Arrived!
Generating packet...

Event #126582

t=92.192490786805s

Msg stats: 58 scheduled / 33147 existing / 38308 c

```
forwarding packet pk-20-to-1-#76 on gate index 0
** Event #126567  t=92.186568163579  Net60a.rte[10].queue[0] (L2Queue, id=105), on 'pk-
Received (Packet)pk-20-to-1-#76
Starting transmission of (Packet)pk-20-to-1-#76
** Event #126568  t=92.186576355579  Net60a.rte[10].queue[0] (L2Queue, id=105), on selfmsg
Transmission finished.
** Event #126569  t=92.186676355579  Net60a.rte[5].queue[2] (L2Queue, id=84), on 'pk-20-to
** Event #126570  t=92.186676355579  Net60a.rte[5].routing (Routing, id=81), on 'pk-20-to-
forwarding packet pk-20-to-1-#76 on gate index 0
** Event #126571  t=92.186676355579  Net60a.rte[5].queue[0] (L2Queue, id=82), on 'pk-20-to
Received (Packet)pk-20-to-1-#76
Starting transmission of (Packet)pk-20-to-1-#76
** Event #126572  t=92.186684547579  Net60a.rte[5].queue[0] (L2Queue, id=82), on selfmsg
Transmission finished.
** Event #126573  t=92.186784547579  Net60a.rte[4].queue[2] (L2Queue, id=78), on 'pk-20-to
** Event #126574  t=92.186784547579  Net60a.rte[4].routing (Routing, id=75), on 'pk-20-to-
forwarding packet pk-20-to-1-#76 on gate index 0
** Event #126575  t=92.186784547579  Net60a.rte[4].queue[0] (L2Queue, id=76), on 'pk-20-to
Received (Packet)pk-20-to-1-#76
Starting transmission of (Packet)pk-20-to-1-#76
** Event #126576  t=92.186792739579  Net60a.rte[4].queue[0] (L2Queue, id=76), on selfmsg
Transmission finished.
** Event #126577  t=92.186892739579  Net60a.rte[1].queue[2] (L2Queue, id=67), on 'pk-20-to
** Event #126578  t=92.186892739579  Net60a.rte[1].routing (Routing, id=64), on 'pk-20-to-
local delivery of packet pk-20-to-1-#76
** Event #126579  t=92.186892739579  Net60a.rte[1].app (App, id=63), on 'pk-20-to-1-#76'
received packet pk-20-to-1-#76 after 7hops
** Event #126580  t=92.192490786805  Net60a.rte[7].app (App, id=90), on selfmsg 'nextPacke
generating packet pk-7-to-50-#84
** Event #126581  t=92.192490786805  Net60a.rte[7].routing (Routing, id=91), on 'pk-7-to-5
forwarding packet pk-7-to-50-#84 on gate index 0
** Event #126582  t=92.192490786805  Net60a.rte[7].queue[0] (L2Queue, id=92), on 'pk-7-to-
Received (Packet)pk-7-to-50-#84
Starting transmission of (Packet)pk-7-to-50-#84
```

The network path of the object is stored in the cluster head and the information is retrieved on query based**.**

CONCLUSION:

The accurate energy model is used for the evaluation of animal habitant in wireless sensor networks including the event detection, tracking and the network path generation. The arrival of the elephant is detected by the sensor node in the network and the packet is generated in the event detection model. The event tracking algorithm is used to track the movement of the elephant in the network. All information related with good NEIGHBORS are stored in routing table which improves performance of routing protocol in terms of good communication and select the cluster head node based on signal strength and flow capacity improves data throughput and overall performance of the network. The GNDA Algorithm with AODV improves the life time of sensor network. The network path is identified with the help of network path identification algorithm. The information related to the location of the node and coverage is stored in the cluster head which is used to find the movement of the elephant in the sensor network.

In the purposed system the number of nodes and coverage area of the node is increased, the event is detected by generating packet, the network path is identified to predict the movement of elephant in the sensor field. The sensor network field is implemented in OMNet++ and all the above mention operations are performed.

5.2. FUTURE WORK:

System design and engineering is one of the keys to bring sensor network paradigm into reality. The system described in this project is still an ongoing prototype. In the proposed system OMNet++ is used to get the simulated result but in future it may be focused on implementing in real time sensor environment. Many outstanding design issues are yet to be resolved for tracking multiple object and data aggregation.

REFERENCES

1. M.Mayilvaganan and M.Devaki, “Direction Of Arrival Estimation and Error Analysis for Wireless Sensor Networks”. In International Journalof Emerging Trends and Technology in Computer Science Volume 2, Issue 2, March-April 2013, ISSN 2278-6856.
2. M.Mayilvaganan and M.Devaki, “Elephant Vocalization Direction of Arrival Estimation and external factors Affecting Wave propagation using Acoustic Sensor network. In International Journal of Innovative Research in Computer and Communication Engineering Volume 1,Issue 4, june 2013, ISSN 2320-9801.
3. Chinthaka M. Dissanayake, Student Member, IEEE, Ramamohanarao KotagiriMalka, N. Halgamug, Member, IEEE, Bill Moran, Member, IEEE, and Peter Farrell†, “Propagation Constraints in Elephant Localization Using an Acoustic Sensor Network 978-1-4673-1975- 1 2012 IEEE
4. Ruwini Edirisinghe, Dileeka Dias, Rakhitha Chandrasekara, Lanka Wijesinghe, Prasanga Siriwardena and Prasad Kumara Sampath ”wi-alert : A wireless sensor network based intrusion alert prototype for HEC”. International Journal of Distributed and Parallel Systems (IJDPS) Vol.4, No.4, July 2013.
5. J.Nirmal Prince, S.J.Sugumar “Surveillance and tracking of elephants using vocal spectral information” IJRET: International Journal of Research in Engineering and Technology eISSN: 2319
6. S. J. Sugumar and R. Jayaparvathy”An early warning system for elephant intrusion along the forest border areas” vol. 104, no. 11, 10 june 2013.
7. Jennifer Yick, Biswanath Mukherjee, Dipak Ghosal,” Wireless sensor network survey” J. Yick et al. / Computer Networks 52 (2008) 2292–2330