DETECTING MOVEMENTS OF A TARGET USING FACE TRACKING IN WIRELESS SENSOR NETWORKS

A PROJECT REPORT

Submitted by

KARTHIKEYAN A	**Reg.No: 1010107049**
ADITH SETHU SS	**Reg.No: 1110107006**
ARUNKUMAR C	**Reg.No: 1110107302**

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY

COIMBATORE-641049

(An Autonomous Institution Affiliated to Anna University, Chennai)

APRIL 2015

KUMARAGURU COLLEGE OF TECHNOLOGY

COIMBATORE-641049

(An Autonomous Institution Affiliated to Anna University, Chennai)

KUMARAGURU COLLEGE OF TECHNOLOGY

COIMBATORE-641049

(An Autonomous Institution Affiliated to Anna University, Chennai)

BONAFIDE CERTIFICATE

Certified that this project report titled **"DETECTING MOVEMENTS OF A TARGET USING FACETRACKING IN WIRELESS SENSOR NETWORKS"** is the bonafied work of "**ADITH SETHU.S.S, ARUNKUMAR.C, KARTHIKEYAN.A"** who carried out the project work under my supervision.

SIGNATURE

Ms.S.Umamaheshwari, M.E., Ph,D.,
Associate Professor/E.C.E
Kumaraguru College of Technology
Coimbatore.

SIGNATURE

Dr. Rajeswari Mariappan M.E., Ph.D.,
HEAD OF THE DEPARTMENT
Electronics & Communication Engineering
Kumaraguru College of Technology
Coimbatore.

The candidates with Register numbers 1110107006, 1110107302, and 1010107049 are examined by us in the project viva-voce examination held on …….

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

First we would like to express our praise and gratitude to the Lord, who has showered his grace and blessing enabling us to complete this project in an excellent manner. He has made all things in beautiful in his time.

We express our sincere thanks to our beloved Joint Correspondent,

Shri. Shankar Vanavarayar for his kind support and for providing necessary facilities to carry out the project work.

We would like to express our sincere thanks to our beloved Principal **Dr.R.S.Kumar M.E., Ph.D.,** who encouraged us with his valuable thoughts.

We would like to express our sincere thanks and deep sense of gratitude to our HOD, **Dr. Rajeswari Mariappan M.E., Ph.D.,** for her valuable suggestions and encouragement which paved way for the successful completion of the project.

We are greatly privileged to express our deep sense of gratitude to the Project Coordinator **Ms.S.Nagarathinam M.E., (Ph.D)**, Assistant Professor (SRG), for her continuous support throughout the course.

In particular, We wish to thank and express our everlasting gratitude to the Supervisor **Ms.S.Umamaheshwari, M.E., (Ph,D),** Assistant Professor for her expert counselling in each and every steps of project work and we wish to convey our deep sense of gratitude to all teaching and non-teaching staff members of ECE Department for their help and cooperation.

Finally, we thank our parents and our family members for giving us the moral support in all of our activities and our dear friends who helped us to endure our difficult times with their unfailing support and warm wishes.

TABLE OF CONTENT

ABSTRACT

Target tracking is one of the key applications of wireless sensor networks (WSNs). Existing work mostly requires organizing groups of sensor nodes with measurements of a target's movements or accurate distance measurements from the nodes to the target, and predicting those movements. These are, however, often difficult to accurately achieve in practice, especially in the case of unpredictable environments, sensor faults, etc. In this paper, we propose a new tracking framework, called Face Track, which employs the nodes of a spatial region surrounding a target, called a face. Instead of predicting the target location separately in a face, we estimate the target's moving toward another face. We introduce an edge detection algorithm to generate each face further in such a way that the nodes can prepare ahead of the target's moving, which greatly helps tracking the target in a timely fashion and recovering from special cases, e.g., sensor fault, loss of tracking. Also, we develop an optimal selection algorithm to select which sensors of faces to query and to forward the tracking data. Simulation results, compared with existing work, show that Face Track achieves better tracking accuracy and energy efficiency.

CHAPTER 1

INTRODUCTION

A wireless sensor network (WSN) consists of a large number of sensors which are densely deployed over a large area. Each sensor monitors a physical environment and communicates via wireless signals. With the advancements in hardware miniaturization and wireless communication technologies, WSNs have been used in various applications such as education, warfare, and traffic monitoring. Regardless of the applications, extending the network lifetime is a critical issue in WSNs. This is because the sensors are battery-powered and generally difficult to be recharged. One of the main objectives. Unlike detection, a target tracking system is often required to ensure continuous monitoring, i.e., there always exist nodes that can detect the target along its trajectory. In target tracking applications, idle listening is a major source of energy waste.

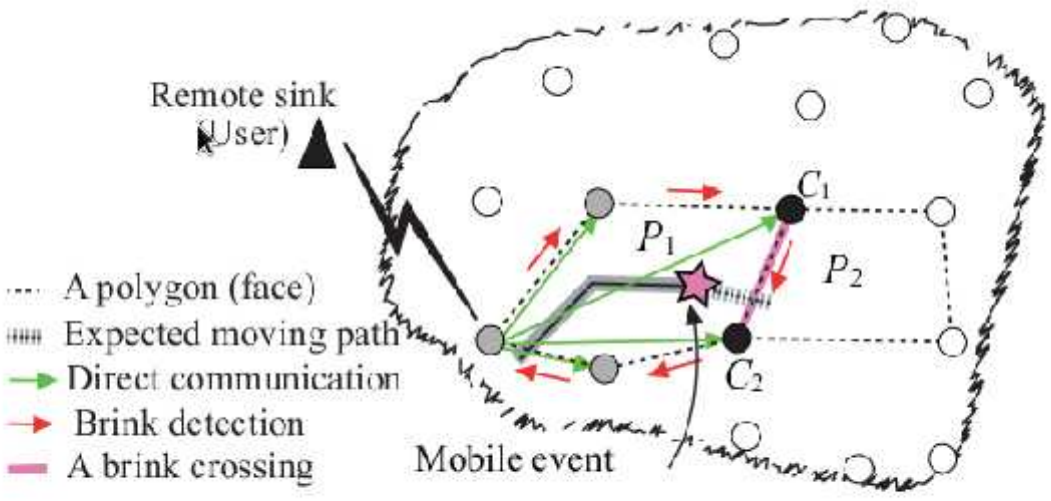


Fig. 1. An example application with a sink showing a vehicle being tracked through a polygonal-shaped area.

Fig. 1 illustrates a typical scenario of an enemy vehicle tracking application.

Sensor nodes are informed when the vehicle under surveillance is discovered, while some nodes (such as black nodes) detect the vehicle and send a vigilance message to the nodes on the vehicle's expected moving path, so as to wake them up. Thus, the nodes (such as grey nodes) in the vehicle's moving path can prepare in advance and remain vigilant in front of it as it moves. To be energy efficient and to accurately track the vehicle, only the nodes close to the path can participate in tracking and providing continuous coverage.

Research about target tracking can be roughly divided into three categories:

1) tree-based schemes
2) cluster-based schemes
3) prediction-based schemes.

In this project, we propose Face Track, a framework to detect movements of a target using face tracking in a WSN, which does not fall into existing categories and is, to the best of our knowledge, the first of its kind. The concept of Face Track is depicted in Fig. 1, and is inspired by computational geometry, geographic routing, and face routing, in particular. Face Track mitigates the discussed difficulties, when satisfying our objectives achieving tracking ability with high accuracy and reducing the energy cost of WSNs.

MAIN CONCEPT

The idea of a plenaries graph, such as the Voronoi diagram and related neighborhood graph (RNG), is mostly used in the network domain. In such a graph, a plane with p points is partitioned into spatial non-overlapping regions, known as faces, by using the term in a face routing strategy, such that each face contains some of the points that are connected. Every two points of a face share an

edge that is also a common edge between two neighboring faces. A target is assumed to be surrounded by the perimeter of it face Pi , e.g., the target lying inside P1, as shown in Fig. 1, can be detected as it goes across an edge/link (such as, < C1; C2 >) toward P2 . The two points (e.g., the black nodes in Fig. 1) become couple nodes chosen from all of the points (neighboring nodes), through a selection process to lead the tracking the target from Pi to Pj.

CHAPTER 2

ROUTING PROTOCOLS IN AD HOC NETWORK

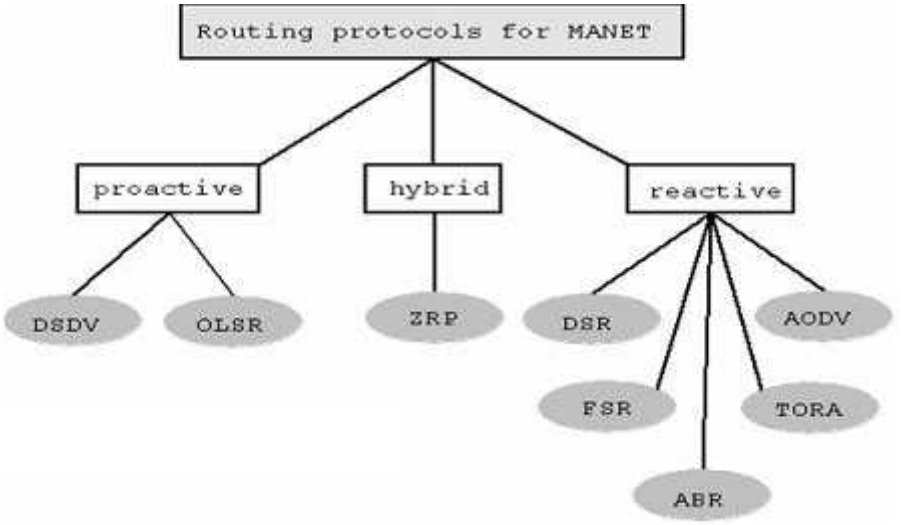
2.1 PROTOCOLS

Table-Driven (or Proactive)

The nodes maintain a table of routes to every destination in the network, for this reason they periodically exchange messages. At all times the routes to all destinations are ready to use and as a consequence initial delays before sending data are small. Keeping routes to all destinations up-to-date, even if they are not used, is a disadvantage with regard to the usage of bandwidth and of network resources.

On-Demand (or Reactive)

These protocols were designed to overcome the wasted effort in maintaining unused routes. Routing information is acquired only when there is a need for it. The needed routes are calculated on demand. This saves the overhead of maintaining unused routes at each node, but on the other hand the latency for sending data packets will considerably increase.



2.1.1 PROACTIVE

DSDV (Destination-Sequence Distance Vector)

DSDV has one routing table, each entry in the table contains: destination address, number of hops toward destination, next hop address. Routing table contains all the destinations that one node can communicate. When a source A communicates with a destination B, it looks up routing table for the entry which contains destination address as B. Next hop address C was taken from that entry. A then sends its packets to C and asks C to forward to B. C and other intermediate nodes will work in a similar way until the packets reach B. DSDV marks each entry by sequence number to distinguish between old and new route for preventing loop.

DSDV use two types of packet to transfer routing information: full dump and incremental packet. The first time two DSDV nodes meet, they exchange all of their available routing information in full dump packet. From that time, they only use incremental packets to notice about change in the routing table to reduce the packet size. Every node in DSDV has to send update routing information

periodically. When two routes are discovered, route with larger sequence number will be chosen. If two routes have the same sequence number, route with smaller hop count to destination will be chosen. DSDV has advantages of simple routing table format, simple routing operation and guarantee loop-freedom. The disadvantages are (i) a large overhead caused by periodical update (ii) waste resource for finding all possible routes between each pair, but only one route is used.

2.1.2 REACTIVE

On-demand Routing Protocols

In on-demand trend, routing information is only created to requested destination. Link is also monitored by periodical Hello messages. If a link in the path is broken, the source needs to rediscovery the path. On-demand strategy causes less overhead and easier to scalability. However, there is more delay because the path is not always ready. The following part will present AODV, DSR, TORA and ABR as characteristic protocols of on-demand trend.

AODV Routing

Ad hoc on demand distance vector routing (AODV) is the combination of DSDV and DSR. In AODV, each node maintains one routing table. Each routing table entry contains:

1. Active neighbor list: a list of neighbor nodes that are actively using this route entry.

2. Once the link in the entry is broken, neighbor nodes in this list will be informed.
3. Destination address
4. Next-hop address toward that destination
5. Number of hops to destination
6. Sequence number: for choosing route and prevent loop
7. Lifetime: time when that entry expires

Routing in AODV consists of two phases: Route Discovery and Route Maintenance. When a node wants to communicate with a destination, it looks up in the routing table. If the destination is found, node transmits data in the same way as in DSDV. If not, it start Route Discovery.

Mechanism: Source node broadcast the Route Request packet to its neighbor nodes, which in turns rebroadcast this request to their neighbor nodes until finding possible way to the destination. When intermediate node receives a RREQ, it updates the route to previous node and checks whether it satisfies the two conditions: (i) there is an available entry which has the same destination with RREQ (ii) its sequence number is greater or equal to sequence number of RREQ. If no, it rebroadcast RREQ. If yes, it generates a RREP message to the source node. When RREP is routed back, node in the reverse path updates their routing table with the added next hop information. If a node receives a RREQ that it has seen before (checked by the sequence number), it discards the RREQ for preventing loop. If source node receives more than one RREP, the one with greater sequence number will be chosen. For two RREPs with the same sequence number, the one will less number of hops to destination will be chosen. When a route is found, it is maintained by Route Maintenance mechanism: Each node periodically send Hello packet to its neighbors for proving its availability. When Hello packet is not

received from a node in a time, link to that node is considered to be broken. The node which does not receive Hello message will invalidate all of its related routes to the failed node and inform other neighbor using this node by Route Error packet. The source if still want to transmit data to the destination should restart Route Discovery to get a new path. AODV has advantages of decreasing the overhead control messages, low processing, quick adapt to network topology change, more scalable up to 10000 mobile nodes. However, the disadvantages are that AODV only accepts bi-directional link and has much delay when it initiates a route and repairs the broken link.

DYNAMIC SOURCE ROUTING PROTOCOL

DSR is a reactive routing protocol which is able to manage a MANET without using periodic table-update messages like table-driven routing protocols do. DSR was specifically designed for use in multi-hop wireless ad hoc networks. Ad-hoc protocol allows the network to be completely self-organizing and self-configuring which means that there is no need for an existing network infrastructure or administration.

For restricting the bandwidth, the process to find a path is only executed when a path is required by a node (On-Demand-Routing). In DSR the sender (source, initiator) determines the whole path from the source to the destination node (Source-Routing) and deposits the addresses of the intermediate nodes of the route in the packets.

Compared to other reactive routing protocols like ABR or SSA, DSR is beacon-less which means that there are no hello-messages used between the nodes to notify their neighbors about her presence.

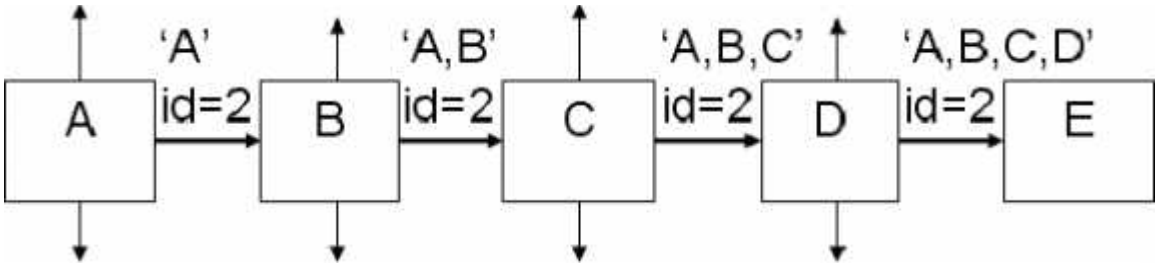
DSR was developed for MANETs with a small diameter between 5 and 10 hops and the nodes should only move around at a moderate speed.

DSR is based on the Link-State-Algorithms which mean that each node is capable to save the best way to a destination. Also if a change appears in the network topology, then the whole network will get this information by flooding.

DSR contains 2 phases

1. Route Discovery (find a path)
2. Route Maintenance (maintain a path)

Route Discovery:

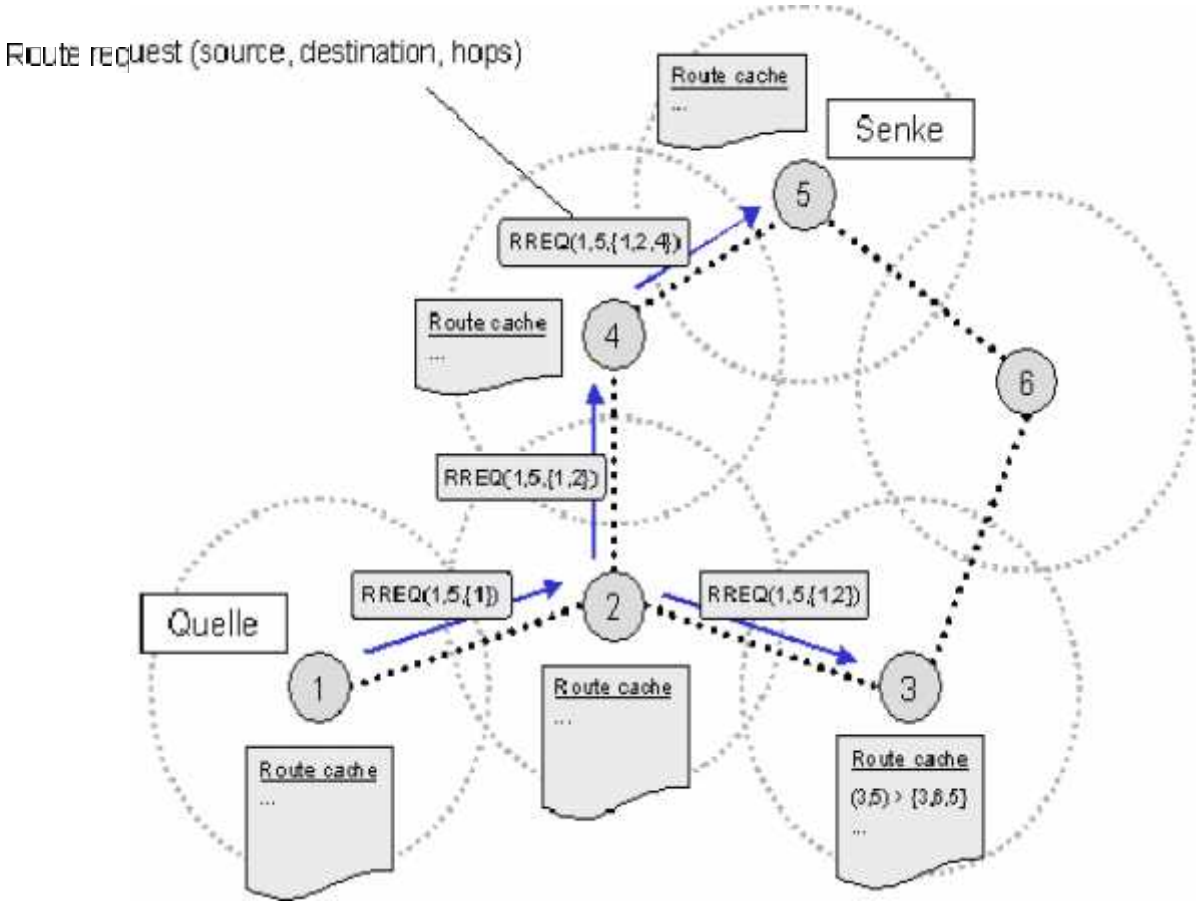


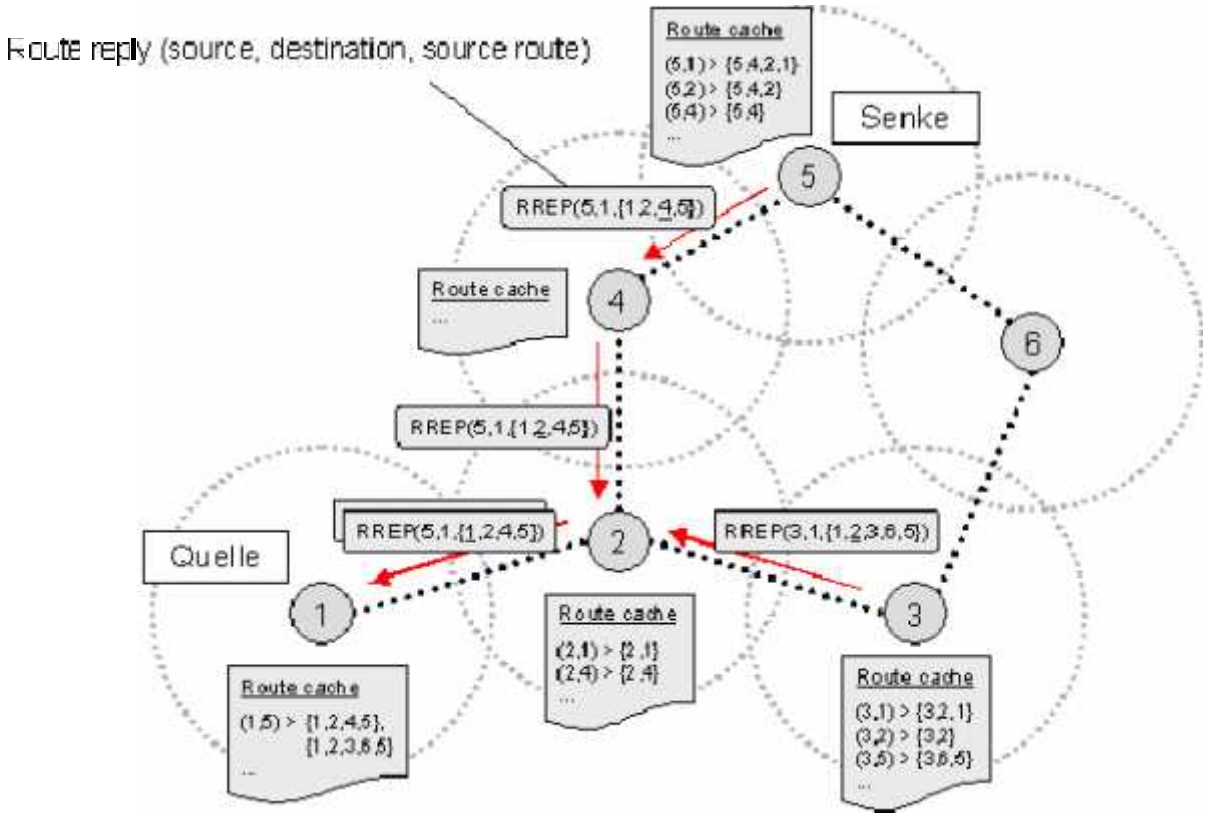
If node A has in his Route Cache a route to the destination E, this route is immediately used. If not, the Route Discovery protocol is started:

1. Node A (initiator) sends a Route Request packet by flooding the network
2. If node B has recently seen another Route Request from the same target or if the address of node B is already listed in the Route Record, Then node B discards the request!

3. If node B is the target of the Route Discovery, it returns a Route Reply to the initiator. The RouteReply contains a list of the “best” path from the initiator to the target. When the initiator receives this Route Reply, it caches this route in its Route Cache for use in sending subsequent packets to this destination.
4. Otherwise node B isn’t the target and it forwards the Route Request to his neighbors (except to the initiator).

Path-finding-process: Route Request & Route Reply

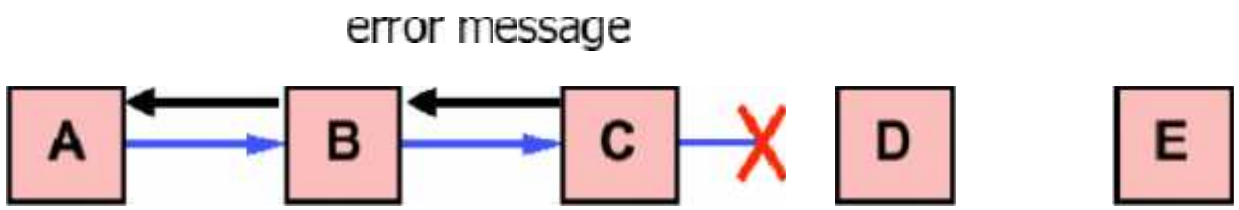




Route Maintenance

In DSR every node is responsible for confirming that the next hop in the Source Route receives the packet. Also each packet is only forwarded once by a node (hop-by-hop routing). If a packet can't be received by a node, it is retransmitted up to some maximum number of times until a confirmation is received from the next hop.

Only if retransmission results then in a failure, a Route Error message is sent to the initiator that can remove that Source Route from its Route Cache. So the initiator can check his Route Cache for another route to the target. If there is no route in the cache, a Route Request packet is broadcasted.



1. If node C does not receive an acknowledgement from node D after some number of requests, it returns a Route Error to the initiator A.
2. As soon as node receives the Route Error message, it deletes the broken-link-route from its cache. If A has another route to E, it sends the packet immediately using this new route.
3. Otherwise the initiator A is starting the Route Discovery process again.

Advantages

Reactive routing protocols have no need to periodically flood the network for updating the routing tables like table-driven routing protocols do. Intermediate nodes are able to utilize the Route Cache information efficiently to reduce the control overhead. The initiator only tries to find a route (path) if actually no route is known (in cache). Current and bandwidth saving because there are no hello messages needed (beacon-less).

Disadvantages

The Route Maintenance protocol does not locally repair a broken link. The broken link is only communicated to the initiator. The DSR protocol is only efficient in MANETs with less than 200 nodes. Problems appear by fast moving of more hosts, so that the nodes can only move around in this case with a moderate speed. Flooding the network can cause collusions between the packets. Also there is always a small time delay at the begin of a new connection because the initiator must first find the route to the target.

2.1.3. TORA (Temporary Ordered Routing Algorithm)

TORA is based on link reversal algorithm. Each node in TORA maintains a table with the distance and status of all the available links. Detail information can be seen at [38]. TORA has three mechanisms for routing:

Route Creation: TORA uses the "height" concept for discovering multiple routes to a destination. Communication in TORA network is downstream, from higher to lower node. When source node does not have a route to destination, it starts Route Creation by broadcasting the Query messages (QRY). QRY is continuing broadcasted until reaching the destination or intermediate node that have the route to the destination. The reached node then broadcast Update (UPD) message which includes its height. Nodes receive this UPD set a larger height for itself than the height in UPD, append this height in its own UPD and broadcast. This mechanism is called reversal algorithm and is claimed to create number of direct links from the originator to the destination.

Route Maintenance: Once a broken link is discovered, nodes make a new reference height and broadcast to their neighbors. All nodes in the link will change their reference height and Route Creation is done to reflect the change.

Route Erasure: Erases the invalid routes by flooding the "clear packet" through the network. The advantages of TORA are: having multiple paths to destination decreases the route creation in link broken case therefore decrease overhead and delay to the network. TORA is also claimed to be effective on large and mildly congested network. The drawbacks are requiring node synchronization due to "height" metric and potential for oscillation. Besides that, TORA may not guarantee to find all the routes for reserving in some cases.

CHAPTER 3

HARDWARE AND SOFTWARE SPECIFICATION

3,1 HARDWARE SPECIFICATION

To develop this system with IBM compatible personal computer with Pentium IV processor was used.

Main processor : Pentium IV processor 1.13 GHz

Hard disk capacity: 40GB

Cache memory : 512 MB

3.2 SOFTWARE SPECIFICATION

Operating system	:	Fedora 8 (linux)
Scripting language	:	Network Simulator 2.33
Protocol developed	:	C++
Scripting	:	Tool Command Language

CHAPTER 4

IMPLEMENTATION

4.1 PROBLEM EXISTING

All wireless sensors are activated initially and idle listening is a major source of energy waste. Once an active sensor runs out of energy, that sensors are not present in the network. So communication is not fully completed.

Drawbacks:

1) All sensors are activated initially (power consumption high)
2) Network lifetime is low.
3) Drop high.
4) Throughput is low.

4.2 PROPOSED METHOD:

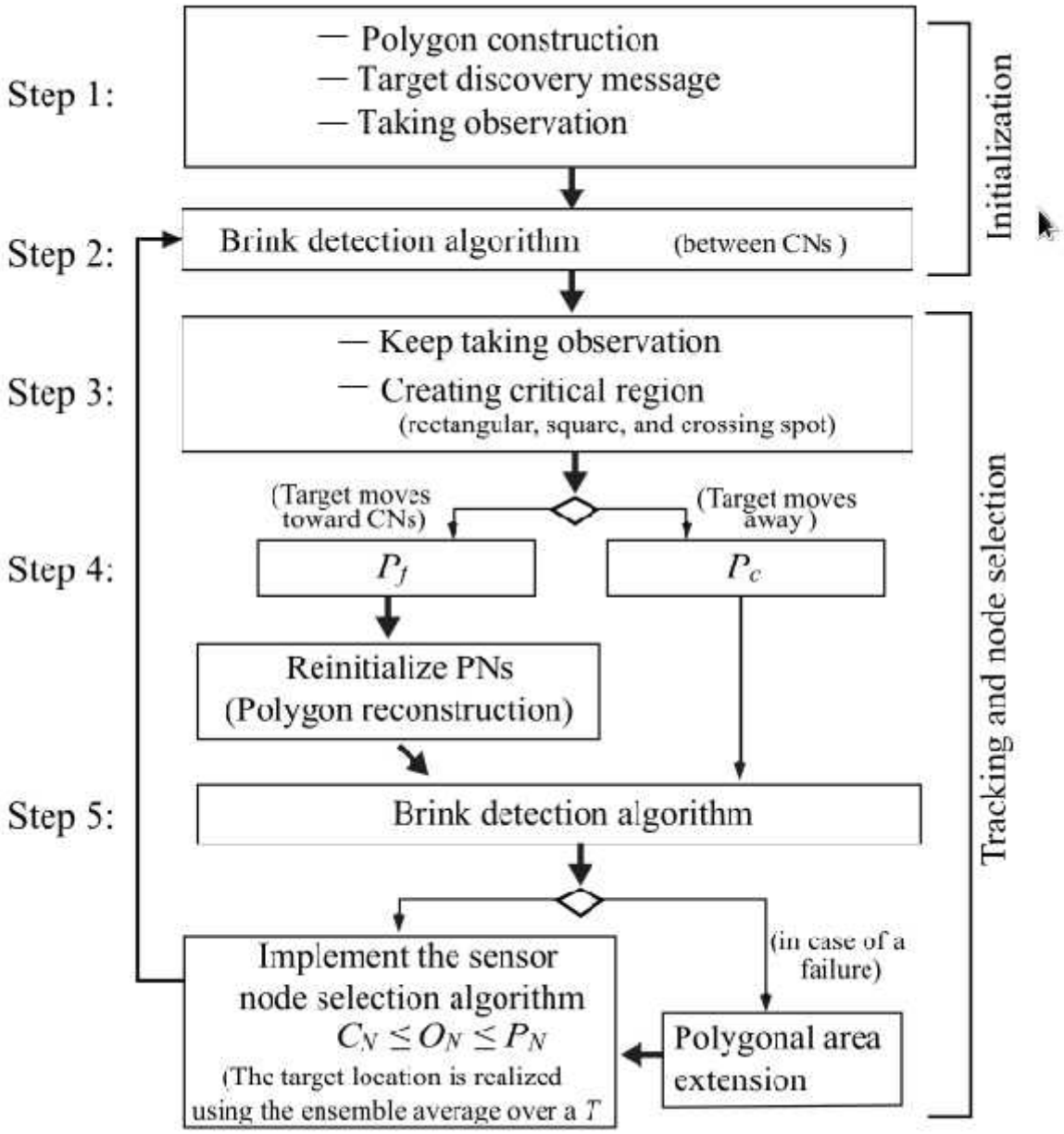


Fig 5: Illustration of polygon-based tracking framework.

The framework of the tracking in Face Track is shown in Fig. There are five steps in the framework. The Step 1 is about the system initialization, including initial polygon construction in the plane. A node has all of the corresponding

polygons' information after the WSN planarization. Initially, all nodes in the WSN are in a low-power mode and wake up at a predefined period to carry out the sensing for a short time. We presume that a sensor node has three different states of operation, namely, active (when a node is in a vigilant manner and participates in tracking the target), awakening (when a node awakes for a short period of time), and inactive (when a node is in a sleeping state). We consider that a sensor should be kept awake so long as its participation is needed for a given task. In the beginning, when a target is detected by some nodes, that communicate to all of its adjacent neighbors with their detection information, and reconstruct the polygon (Step 2). Once the target is surrounded by the perimeter of a polygon, it becomes Pc. Steps 3 to 5 (including brink detection through the three-phase detection, optimal sensor selection, and polygonal area extension in the case of faults in the WSN or loss of tracking) are continued during the target tracking. Whenever the CNs are selected by the optimal selection algorithm, the detection probability, confirms that the target is about to cross the rectangular phase and then the crossing phase (Step 3). A joint-request message is sent to Pf at the moment the target touches the rectangular phase, saying that the target is approaching (Step 4). All NNs in Pf receive the request, change their state to an awakening, and then start sensing. When the target crosses the brink, another joint-request message is sent to the nodes in Pf, saying that the target is crossing the brink. After the target crosses over the brink (i.e., it is now in the new Pc), another message is sent to the NNs in the previous Pc. After receiving the message, all NNs, except the previous CNs, return to the inactive state. The target may move in any way toward any brink. When the target speed is lower or the target moves away, it does not influence the tracking. We think of the target's faster speed. When it is faster, the movements may be abrupt. The CNs keep sensing continuously until the target

leaves/enters the square phase. The CNs uses the difference in distance dij between two consecutive sensing results. The results are measured by reducing CRLB covariance to obtain fewer errors in three-phase detection. Since the target travels across the square phase, and then the rectangular phase, dij decreases accordingly. The CNs are aware of it. If the target leaves the square phase for the same Pc, the CNs send a message instantly to the NNs in Pc . The Nns remain active and are ready to receive the message. If they receive the message, shortly there-after, they start sensing further. The next procedures go on in the aforementioned way. However, if any rectangular phase is not generated, there is no Pf selected.

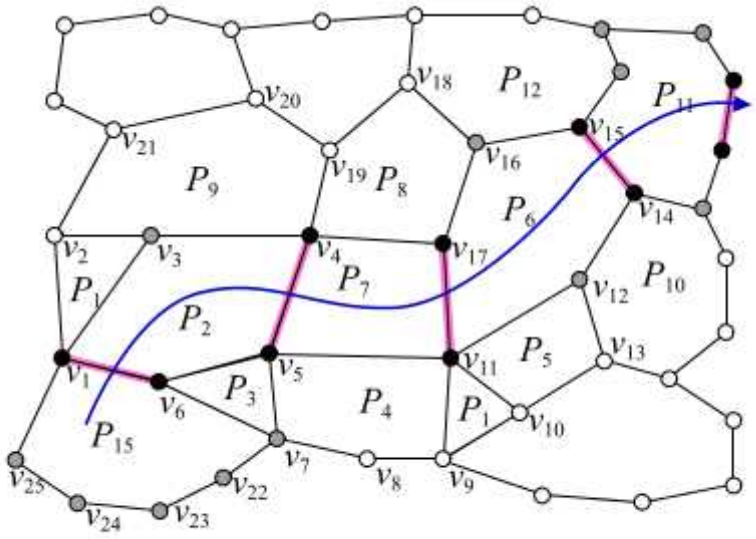


Fig. 6. Detecting target's movements through polygons.

According to the framework in Fig. 5, Fig. 6 illustrates the target movement detection through the polygons. The target is initially detected by sensors v1 and v6 (shaded black to indicate the CNs) in the polygon P15, and the rest of the

corresponding nodes (shaded grey) in P15 are in the vigilant manner, and the rest of the nodes in the sensor network are in the inactive state when the target is in P15. As shown in Fig. 6, the target travels through the polygons. The tracking of the polygons represents the target tracks. A tracking sequence can be P15-> P2-> P7->
P6 -> P11, and so on.

We enhance the proposed Face Tracking design to detect and track multiple targets by activating the sensor nodes along the boundaries so as to detect multiple targets coming from different ends once we detect the Target we will track the target and send the current information about the target to the base station.

Advantages:

1. Selected sensors nodes are activated.
2. Network lifetime is high.
3. Reduce the drop.
4. Increase throughput.
5. Detects multiple mobile Targets.

MODULES:

1. Topology Formation.
2. .Predicting the target and creating Local Active environment.
3. Face Tracking
4. Modified Face Tracking protocol.

1.Topology Formation :

All sensors are deployed initially. Each sensor updates their information to its neighbour sensor. This is called Initial Neighbor Discovery.

2.Predicting the target and creating Local Active environment :

All sensors communicate with each other and updates the routing information once object is detected creates a Local Active environment predicts the Target movement and sends the information to base station.

3.Face Tracking:

Once Target is detected creates an Awake region and based on the prediction results assigns Sleep scheduling to individual sensors at synchronised time and the graph is ploted for Energy efficiency in comparison with the Existing concept along with Throughput, Packet Delivery ratio.

4.Modified Face Tracking protocol:

In this phase we are synchronizing the proposed PPSS protocol, i.e., Local Active environment with Boundary selection nodes in which the sensors along the boundary of the field region are activated, thus the Mobile target that comes from different directions are detected, once it detects the Moving object along the boundaries, it will start sending the information about the mobile target to the base station, so we are enhancing the proposed concept to detect multiple target along with improved power efficiency.

CHAPTER 5

SOFTWARE DESCRIPTION

5.1 THE NETWORK SIMULATOR 2.33 (NS2)

Network Simulator (NS2) is a discrete event driven simulator developed at UC Berkeley. It is part of the VINT project. The goal of NS2 is to support networking research and education. It is suitable for designing new protocols, comparing different protocols and traffic evaluations. NS2 is developed as a collaborative environment. It is distributed freely and open source. A large amount of institutes and people in development and research use, maintain and develop NS2. This increases the confidence in it. Versions are available for FreeBSD, Linux, Solaris, Windows and Mac OS X.

5.2 STRUCTURE OF NS2

NS2 is built using object oriented methods in C++ and OTcl (object oriented variant of Tcl.

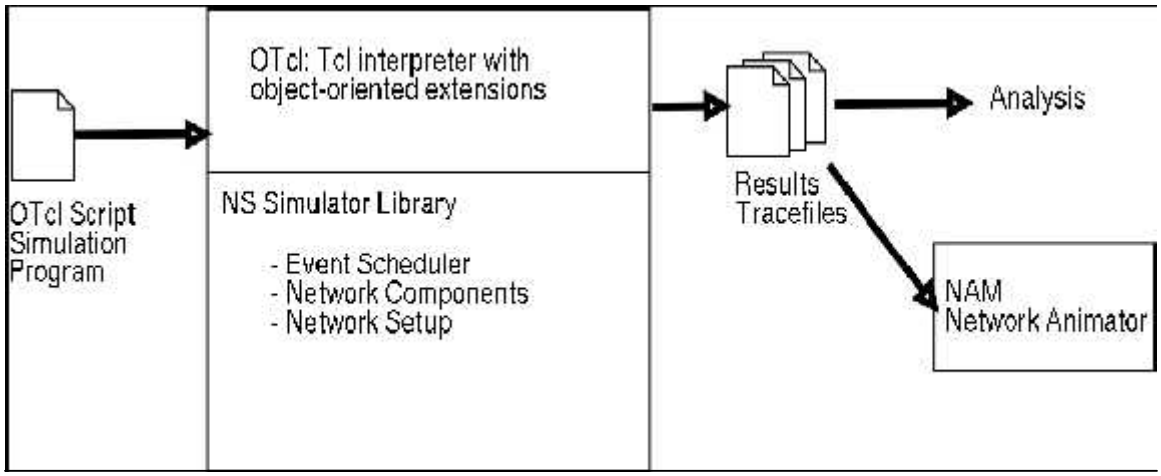


Fig 5.1 Simplified User's View of NS

Can see in Fig 5.1, NS2 interprets the simulation scripts written in OTcl. A user has to set the different components (e.g. event scheduler objects, network components libraries and setup module libraries) up in the simulation environment. The user writes his simulation as a OTcl script, plumbs the network components together to the complete simulation. If he needs new network components, he is free to implement them and to set them up in his simulation as well. The event scheduler as the other major component besides network components triggers the events of the simulation (e.g. sends packets, starts and stops tracing). Some parts of NS2 are written in C++ for efficiency reasons. The data path (written in C++) is separated from the control path (written in OTcl). Data path object are compiled and then made available to the OTcl interpreter through an OTcl linkage (tclcl) which maps methods and member variables of the C++ object to methods and variables of the linked OTcl object. The C++ objects are controlled by OTcl objects. It is possible to add methods and member variables to a C++ linked OTcl object.

5.3 FUNCTIONALITIES OF NS2.33

Functionalities for wired, wireless networks, tracing, and visualization are available in NS2.

• Support for the wired world include

– Routing DV, LS, and PIM-SM.

– Transport protocols: TCP and UDP for unicast and SRM for multicast.

– Traffic sources: web, ftp, telnet, cbr (constant bit rate), stochastic, real audio.

– Different types of Queues: drop-tail, RED, FQ, SFQ, DRR.

– Quality of Service: Integrated Services and Differentiated Services.

– Emulation.

• Support for the wireless world include

– Ad hoc routing with different protocols, e.g. AODV, DSR, DSDV, TORA

– Wired-cum-wireless networks

– Mobile IP

– Directed diffusion

– Satellite

– Senso-MAC

– Multiple propagation models (Free space, two-ray ground, shadowing)

– Energy models

• Tracing

• Visualization

– Network Animator (NAM)

– Trace Graph

• Utilities

– Mobile Movement Generator

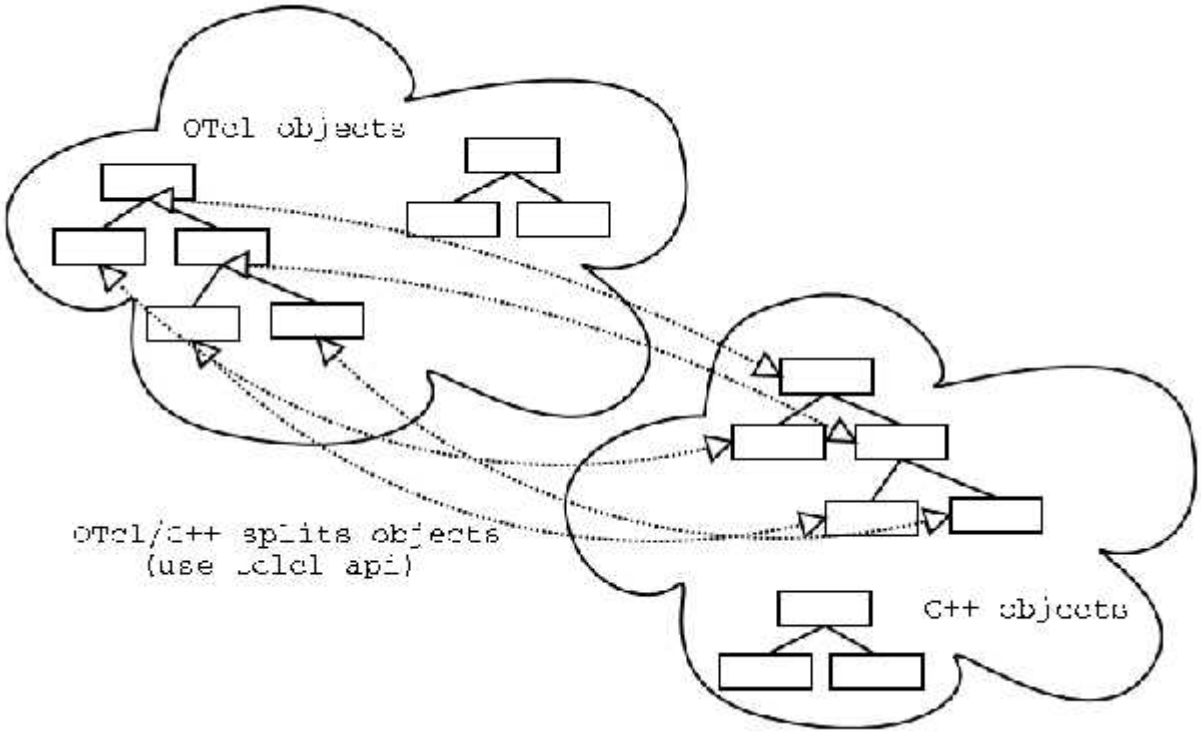


Fig 5.2 OTcl and C++: the duality

5.3.1 MOBILE NETWORKING IN NS2.33

This section describes the wireless model that was originally ported as CMU's Monarch group's mobility extension to NS2. The first section covers the original mobility model ported from CMU/Monarch group. In this section, we cover the internals of a mobile node, routing mechanisms and network components that are used to construct the network stack for a mobile node. The components that are covered briefly are Channel, Network interface, Radio propagation model, MAC protocols, Interface Queue, Link layer and Address resolution protocol model (ARP). CMU trace support and Generation of node movement and traffic scenario files are also covered in this section. The original CMU model allows simulation of pure wireless LANs or multi hop ad-hoc networks. Further

extensions were made to this model to allow combined simulation of wired and wireless networks. Mobile IP was also extended to the wireless model.

5.3.2 THE BASIC WIRELESS MODEL IN NS

The wireless model essentially consists of the Mobile Node at the core, with additional supporting features that allows simulations of multi-hop ad-hoc networks, wireless LANs etc. The Mobile Node object is a split object. The C++ class Mobile Node is derived from parent class Node. A Mobile Node thus is the basic Node object with added functionalities of a wireless and mobile node like ability to move within a given topology, ability to receive and transmit signals to and from a wireless channel etc. A major difference between them, though, is that a Mobile Node is not connected by means of Links to other nodes or mobile nodes. In this section we shall describe the internals of Mobile Node, its routing mechanisms, the routing protocols dsdv, aodv, tora and dsr, creation of network stack allowing channel access in Mobile Node, brief description of each stack component, trace support and movement/traffic scenario generation for wireless simulations.

5.3.3 MOBILE NODE: CREATING WIRELESS TOPOLOGY

Mobile Node is the basic *ns* Node object with added functionalities like movement, ability to transmit and receive on a channel that allows it to be used to create mobile, wireless simulation environments. The class Mobile Node is derived from the base class Node. Mobile Node is a split object. The mobility features including node movement, periodic position updates, maintaining topology boundary etc are implemented in C++ while plumbing of network components within Mobile Node itself (like classifiers, dmux , LL, Mac, Channel etc) have been implemented in Otcl.

Table 5.1: Available Options For Node Configuration

Option	Available Values	Default
General		
Address type	Flat, Hierarchical	Flat
MPLS	ON,OFF	OFF
Both Satellite and Wireless Oriented		
Wired Routing	ON,OFF	OFF
II Type	LL,LL/sat	OFF
Mac Type	Mac/802_11,Mac/Csma/Ca, Mac/Sat/Unslotted/Aloha,Mac/Tdma	OFF
ifq Type	Queue/DropTail, Queue/Droptail/PriQueue	OFF
Phy Type	Phy/wirelessPhy,Physat	OFF

Option	Available Values	Default
Satellite Oriented		
satNodeType	Polar,Geo,Terminal,Geo-repeater	OFF
downlinkBW	<bandwidth value, e.g “2MB”>	OFF
Wireless Oriented		
Adhoc Routing	DIFFUSION/RATE,DIFFUSION/PROB, DSDV,FLOODING,OMNICAST,AODV,TORA	OFF
propType	Propagation/2RayGround,Propagation Shadowing	OFF
propInstance	Propagation/2RayGround,Propagation Shadowing	OFF
antType	Antenna/Omni Antenna	OFF
Channel	Channel/Wireless Channel,Channel/sat	OFF
topoInstance	<toplogy file>	OFF
MobileIP	ON,OFF	OFF
Energy model	Energy model	OFF
Initial Energy	<value in joules>	OFF
rxPower	<value in W>	OFF
txPower	<value in W>	OFF
Idle Power	<value in W>	OFF

AgentTrace	ON,OFF	OFF
routerTrace	ON,OFF	OFF
macTrace	ON,OFF	OFF
movementTrace	ON,OFF	OFF
Errproc	UniformErrorProc	OFF
FECProc	?	?
toraDebug	ON,OFF	OFF

IMPLEMENTATION ENVIRONMENT

Network simulator 2 is used as the simulation tool in this project. NS was chosen as the simulator partly because of the range of features it provides and partly because it has an open source code that can be modified and extended. There are different versions of NS and the latest version is ns-2.1b9a while ns-2.1b10 is under development

5.4 NETWORK SIMULATOR (NS)

Network simulator (NS) is an object–oriented, discrete event simulator for networking research. NS provides substantial support for simulation of TCP, routing and multicast protocols over wired and wireless networks. The simulator is a result of an ongoing effort of research and developed. Even though there is a

considerable confidence in NS, it is not a polished product yet and bugs are being discovered and corrected continuously.

NS is written in C++, with an OTcl1 interpreter as a command and configuration interface. The C++ part, which is fast to run but slower to change, is used for detailed protocol implementation. The OTcl part, on the other hand, which runs much slower but can be changed very fast quickly, is used for simulation configuration. One of the advantages of this split-language program approach is that it allows for fast generation of large scenarios. To simply use the simulator, it is sufficient to know

OTcl. On the other hand, one disadvantage is that modifying and extending the simulator requires programming and debugging in both languages.

NS can simulate the following:

1. Topology **:** Wired, wireless

2. Scheduling Algorithms: RED, Drop Tail,

3. Transport Protocols **:** TCP, UDP

4. Routing **:** Static and dynamic routing

5. Application **:** FTP, HTTP, Telnet, Traffic generators

5.5 USER'S VIEW OF NS-2

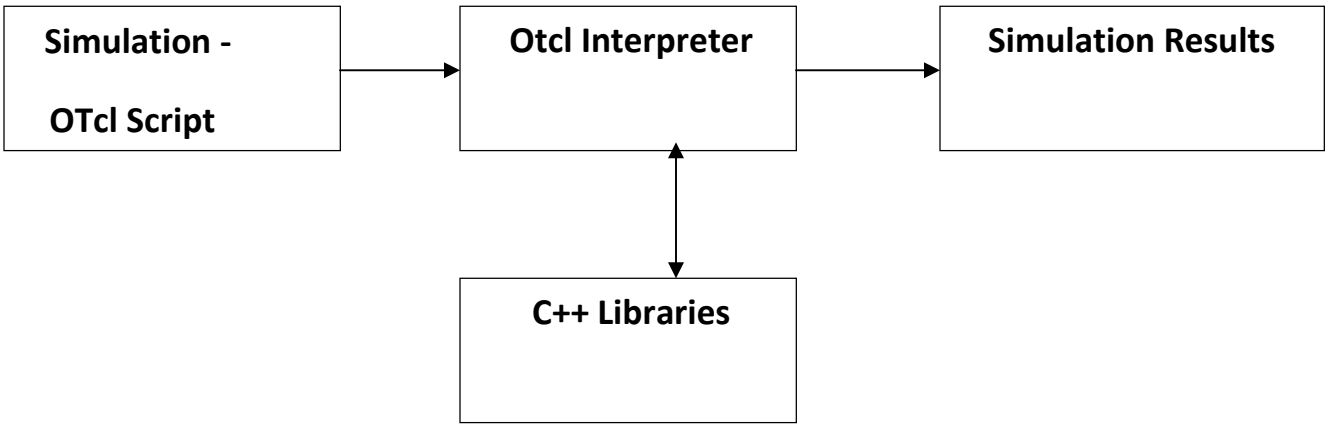


Figure: 5.3 Block diagram of Architecture of NS-2

5.6 NETWORK COMPONENTS

This section talks about the NS components, mostly compound network components. Figure 1.1 shows a partial OTcl class hierarchy of NS, which will help understanding the basic network components.

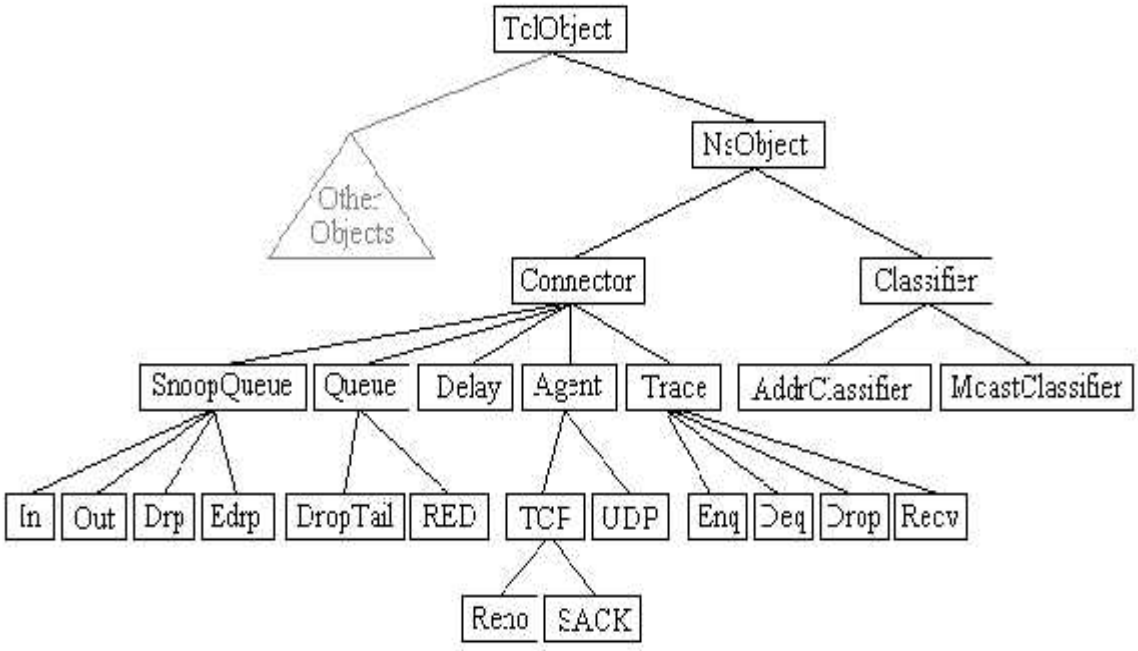


Figure 5.4 OTcl Class Hierarchy

The root of the hierarchy is the TclObject class that is the super class of all OTcl library objects (scheduler, network components, timers and the other objects including NAM related ones). As an ancestor class of TclObject, NsObject class is the super class of all basic network component objects that handle packets, which may compose compound network objects such as nodes and links. The basic network components are further divided into two subclasses, Connector and Classifier, based on the number of the possible output DATA paths. The basic network and objects that have only one output DATA path are under the Connector class, and switching objects that have possible multiple output DATA paths are under the Classifier class.

5.7 CLASS TCL

The class Tcl encapsulates the actual instance of the OTcl interpreter and provides the methods to access and communicate with that interpreter, code. The class provides methods for the following operations:

1) obtain a reference to the Tel instance
2) invoke OTcl procedures through the interpreter
3) retrieve, or pass back results to the interpreter
4) report error situations and exit in an uniform manner
5) store and lookup "TclObjects"
6) acquire direct access to the interpreter.

5.7.1 Obtain a Reference to the class Tcl instance

A single instance of the class is declared in -tclcl/Tcl.cc as a static member variable. The statement required to access this instance is Tel& tel = Tcl::instance();

5.7.2 Invoking OTcl Procedures

There are four different methods to invoke an OTcl command through the instance, tcl. They differ essentially in their calling arguments. Each function passes a string to the interpreter that then evaluates the string in a global context. These methods will return to the caller if the interpreter returns TCL_OK. On the other hand, if the interpreter returns TCL_ERROR, the methods will call tkerror{}. The user can overload this procedure to selectively disregard certain types of errors.

1. **Passing Results to/from the Interpreter:** When the interpreter invokes a C++ method, it expects the result back in the private member variable, tcl-> result.
2. **Error Reporting and Exit:** This method provides a uniform way to report errors in the compiled code.

5.8 COMMAND METHODS: DEFINITION AND INVOCATION

For every TclObject that is created, ns establishes the instance procedure, cmd{}, as a hook to executing methods through the compiled shadow

object. The procedure cmd{} invokes the method command() of the shadow object automatically, passing the arguments to cmd{} as an argument vector to the command() method. The user can invoke the cmd {} method in one of two ways, by explicitly invoking the procedure, specifying the desired operation as the first argument, or implicitly, as if there were an instance procedure of the same name as the desired operation. Most simulation scripts will use the latter form.

Consider the distance computation in SRM is done by the compiled object. It is often used by the interpreted object. It is usually invoked as $srmObject distance? (agentAddress)If there is no instance procedure called distance? the interpreter will invoke the instance procedure unknown{}, defined in the base class TclObject. The unknown procedure then invokes

$srmObject cmd distance? (agentAddress)

To execute the operation through the compiled object's command() procedure. The user could explicitly invoke the operation directly. One reason for this might be to overload the operation by using an instance procedure of the same name.

For example,

```
Agent/SRM/Adaptive instproc distance? addr
{
$self instvar distanceCache_($addr)
if![info exists distanceCache_($addr)]
 {
set distanceCache_($addr) [$self cmd distance? $addr]
```

```
}
set distanceCache_($addr)
}
```

The following shows how the command() method using SRMAgent::command()

```
int ASRMAgent::command(int argc, const char*const*argv)
{
Tcl& tcl = Tcl::instance();
if (argc == 3) {
if (strcmp(argv[1], "distance?") == 0)
{
int sender = atoi(argv[2]);
SRMinfo* sp = get_state(sender);
tcl.tesultf("%f", sp->distance_);
return TCL_OK;
}
}
return (SRMAgent::command(argc, argv));
```

The following observations are made from this piece of code:

The function is called with two arguments. The first argument (argc) indicates the number of arguments specified in the command line to the interpreter. The command line arguments vector (argv) consists of argv[0] contains the name of the method, "cmd" and argv[1] specifies the desired operation. If the user specified any arguments, then they are placed in argv[2...(argc - 1)]. The arguments are passed as strings. They must be converted to the appropriate data type. If the operation is successfully matched, the match should return the result of the operation, command () itself must return either TCL_OK or TCL_ERROR to indicate success or failure as its return code. If matched in this method, it must invoke its parent's command method, and return the corresponding result. This permits the user to conceive of operations as having the same inheritance properties as instance procedures or compiled methods. In the event that this command method is defined for a class with multiple inheritances, one of two implementations can be chosen

1. Either they can invoke one of the parent's command methods, and return the result of that invocation.

2. They can each of the parent's command methods in some sequence, and return the result of the first invocation that is successful. If none of them are successful, then they should return an error.

5.9 MOBILE NETWORKING IN NS

The wireless model essentially consists of the Mobile Node at the core with additional supporting features that allows simulations of multi-hop ad-hoc networks, wireless LANs etc. The Mobile Node object is a split object. The C++ class Mobile Node is derived from parent class Node. A Mobile Node thus is the basic Node object with added functionalities of a wireless and mobile node like ability to move within a given topology, ability to receive and transmit signals to and from a wireless channel etc. A major difference between them is that a mobile Node is not connected by means of Links to other nodes or mobile nodes.

Mobile Node is the basic nsNode object with added functionalities like movement, ability to transmit and receive on a channel that allows it to be used to create mobile, wireless simulation environments. The class Mobile Node is derived from the base class Node. The four ad-hoc routing protocols that are currently supported are, Dynamic Source Routing (DSR), Temporally ordered Routing Algorithm (TORA) and Adhoc On-demand Distance Vector (AODV).

The general structure for defining a mobile node in ns2 is described as follows:

$ns node-config -adhocRouting $opt (adhocRouting)

-IIType $opt (II)

-macType $opt (mac)

-ifqType $opt (ifq) -ifqLen $opt (ifqlen)

-antType $opt (ant)

-proplnstance [new $opt (prop) -phyType $opt (netif)

-channel [new $opt (chan)]

-topolnstance $topo

-wiredRouting OFF

-agentTrace ON

-routerTrace OFF

-macTrace OFF

The above API configures for a mobile node with all the given values of ad hoc-routing protocol, network stack, channel, topography, propagation model,

with wired routing turned on or off (required for wired-cum-wireless scenarios) and tracing turned on or off at different levels (router, mac, agent).

CONCLUSION

The main functionality of a surveillance wireless sensor network is to track an unauthorized target in a field. The challenge is to determine how to perceive the target in a WSN efficiently. We proposed a unique idea to achieve a WSN system for detecting movements of a target using polygon (face) tracking that does not adopt any prediction method. Evaluation results demonstrated that the proposed tracking framework can estimate a target's positioning area, achieve tracking ability with high accuracy, and reduce the energy cost of WSNs. From the framework, two facts can be highlighted emphatically:

1) The target is always detected inside a polygon by means of a brink detection,
2) It is robust to sensor node failures and target localization errors.

Two interesting problems, which we are currently investigating, are as follows:

1) The performance of variable brink lengths of the polygon versus adjustable transmission power levels in a WSN for target detection and its energy cost in the WSNs.
2) The impact of the target's dynamic movements, brink detection, and real-time polygon forwarding in target tracking.

REFERENCES

[1] O. Kaltiokallio, M. Bocca, and L.M. Eriksson, “Distributed RSSI Processing for Intrusion Detection in Indoor Environments,” Proc. Ninth ACM/IEEE Int’l Conf. Information Processing in Sensor Networks (IPSN), pp. 404-405, 2010.

[2] Y. Wang, M. Vuran, and S. Goddard, “Analysis of Event Detection Delay in Wireless Sensor Networks,” Proc. IEEE INFOCOM, pp. 1296-1304, 2011.

[3] Z. Zhong, T. Zhu, D. Wang, and T. He, “Tracking with Unreliable Node Sequence,” Proc. IEEE INFOCOM, pp. 1215-1223, 2009.

[4] W. Zhang and G. Cao, “Dynamic Convoy Tree-Based Collabora tion for Target Tracking in Sensor Networks,” IEEE Trans. Wireless Comm., vol. 3, no. 5, Sept. 2004.

[5] Z. Wang, W. Lou, Z. Wang, J. Ma, and H. Chen, “A Novel Mobility Management Scheme for Target Tracking in Cluster-Based Sensor Networks,” Proc. Sixth IEEE Int’l Conf. Distributed Computing in Sensor Systems (DCOSS), pp. 172-186, 2010.

[6] L.M. Kaplan, “Global Node Selection for Localization in a Distributed Sensor Network,” IEEE Trans. Aerospace and Electronic Systems, vol. 42, no. 1, pp. 113-135, Jan. 2006.

[7] T. He, P. Vicaire, T. Yan, L. Luo, L. Gu, G. Zhou, R. Stoleru, Q.Cao, J. Stankovic, and T. Abdelzaher, “Achieving Real-Time Target Tracking Using Wireless Sensor Networks,” Proc. 12th IEEE Real-Time and Embedded Technology and Applications Symp. (RTAS), pp. 37-48, 2006.

[8] M. Waelchli, M. Scheidegger, and T. Braun, “Intensity-Based Event Localization in Wireless Sensor Networks,” Proc. Conf. Int’l Federation for Information Processing Wireless On-Demand Network Systems and Services (IFIP WONS), pp. 41-49, 2006.

[9] Y. Zhou, J. Li, and D. Wang, “Posterior Cramer-Rao Lower Bounds for Target Tracking in Sensor Networks with Quantized Range-Only Measurements,” IEEE Signal Processing Letters, vol. 17, no. 2, pp. 377-388, Feb. 2010.

[10] X. Wang, M. Fu, and H. Zhang, “Target Tracking in Wireless Sensor Networks Based on the Combination of KF and MLE Using Distance Measurements,” IEEE Trans. Mobile Computing, vol. 11, no. 4, pp. 567-576, Apr. 2012.

[11] M. Chu, H. Haussecker, and F. Zhao, “Scalable Information Driven Sensor Querying and Routing for Ad Hoc Heterogeneous Sensor Networks,” J. High Performance Computing Applications, vol. 16, no. 3, pp. 293-313, 2002.

[12] B. Leong, S. Mitra, and B. Liskov, “Path Vector Face Routing: Geographic Routing with Local Face Information,” Proc. IEEE Int’l Conf. Network Protocols (ICNP), pp. 47-158, 2005.

[13] Y.-J. Kim, R. Govindan, B. Karp, and S. Shenker, “Geographic Routing Made Practical,” Proc. USENIX Networked. Systems Design and Implementation (NSDI), pp. 217-230, 2005.

[14] M.A. Rajan, M.G. Chandra, L.C. Reddy, and P. Hiremath, “Concepts of Graph Theory Relevant to Ad-Hoc Networks,” J. Computers, Comm. and Control, vol. 3, no. 2008, pp. 465-469, 2008.

[15] Q. Huang, C. Lu, and G.-C. Roman, “Mobicast: Just-in-Time Multicast for Sensor Networks under Spatiotemporal Constraints,” Proc. ACM/IEEE Int’l Conf. Information Processing in Sensor Networks (IPSN), pp. 442-457, 2003.

[16] K. Liu, N. Abu-Ghazaleh, and K.D. Kang, “JiTS: Just-in-time Scheduling for Real-Time Sensor Data Dissemination,” Proc. IEEE Pervasive Computing and Comm. (PerCom), pp. 42-46, 2006.

[17] M.Z.A. Bhuiyan, G. Wang, and J. Wu, “Polygon-Based Tracking Framework in Surveillance Wireless Sensor Networks,” Proc. IEEE Int’l Conf. Parallel and Distributed Systems (ICPADS), pp. 174-181, 2009.

[18] L.M. Kaplan, “Local Node Selection for Localization in a Distributed Sensor Network,” IEEE Trans. Aerospace and Electronic Systems, vol. 42, no. 1, pp. 136-146, Jan. 2006.

[19] TinyOS Reference Manual, http://www.tinyos.net, 2013.

[20] Imote2 Datasheet, http://docs.tinyos.net/tinywiki/index.php/Imote2, 2013.

[21] M.Z.A. Bhuiyan, G. Wang, and J. Wu, “Target Tracking with Monitor and Backup Sensors in Wireless Sensor Networks,” Proc. IEEE Int. Conf. Computer Comm. and Networks (ICCCN), pp. 1-6, 2009.

[22] J. Cartigny, F. Ingelrest, D. Simplot, and I. Stojmenovic, “Localized LMST and RNG Based Minimum-Energy Broadcast Protocols in Ad Hoc Networks,” Ad Hoc Networks (Elsevier), vol. 3, no. 2, pp. 1-16, 2005.

[23] G. Toussaint, “The Relative Neighborhood Graph of Finite Planar Set,” Pattern Recognition, vol. 12, no. 4, pp. 261-268, 1980.

[24] M. Crocker, Handbook of Acoustics. John Wiley & Sons, 1998.

[25] M.D. Berg, M.V. Kerveid, M. Overmars, and O. Schwarzkof, Computational Geometry. Springer, 1998.