IOSR Journal of Electronics and Communication Engineering (IOSR-JECE)
e-ISSN: 2278-2834,p- ISSN: 2278-8735.Volume 9, Issue 1, Ver. V (Feb. 2014), PP 53-59
www.iosrjournals.org

Adaptive Position Update for Geographic Routing In Mobile Ad Hoc Networks

Gibin N.G[1],S. Ashok Kumar[2]
[1(Hindusthan college of engineering and technology Coimbatore)
[2(Assistant Professor Hindusthan college of engineering and technology Coimbatore)

***Abstract*:** *In geographic routing, nodes need to maintain up-to-date positions of their immediate neighbours for making effectiveforwarding decisions. Periodic broadcasting of beacon packets that contain the geographic location coordinates of the nodes is apopular method used by most geographic routing protocols to maintain neighbour positions. We contend and demonstrate that periodicbeaconing regardless of the node mobility and traffic patterns in the network is not attractive from both update cost and routingperformance points of view. We propose the Adaptive Position Update (APU) strategy for geographic routing, which dynamicallyadjusts the frequency of position updates based on the mobility dynamics of the nodes and the forwarding patterns in the network. APUis based on two simple principles: nodes whose movements are harder to predict update their positions more frequently (and vice versa), and nodescloser to forwarding paths update their positions more frequently (and vice versa).*

I. Introduction

WIRELESS NETWORKS

Wireless networks are generally implemented with some type of remote information transmission system that uses electromagnetic waves, and is commonly associated with a telecommunications network whose interconnections between nodes is implemented without the use of wires.

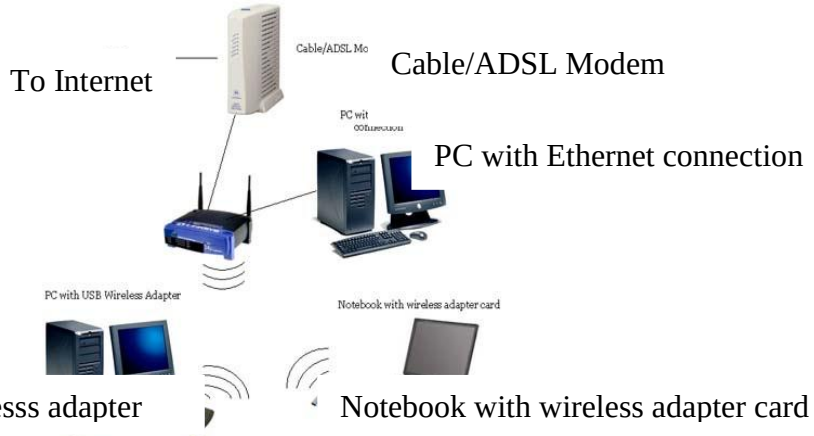

Fig 1 Wireless Network

The wireless network is broadly classified into two types:

- Infrastructure Networks
- Infrastructureless Networks

Infrastructure Networks

Infrastructure network consists of a network with fixed and wired gateways. A mobile host communicates with a bridge in the network (called base station) within its communication radius. The mobile unit can move geographically while it is communicating. When it goes out of range of one base station, it connects with new base station and start communicating through it. This is called handoff.

Infrastructureless Networks

In Infrastructureless Networks or Wireless ad hoc networks all nodes are mobile and can be connected dynamically in an arbitrary manner. All nodes of these networks behave as routers and take part in discovery and maintenance of routes to other nodes in the network.

WIRELESS AD HOC NETWORK

A wireless ad hoc network is a decentralized wireless network. The network is ad hoc because it does not rely on a preexisting infrastructure, such as routers in wired networks or access points in managed (infrastructure) wireless networks. Instead, each node participates in routing by forwarding data to other nodes, and so the determination of which nodes forward data is made dynamically based on the network connectivity. Since mobile hosts generally have poor resources, it is usually impossible for them to have replicas of all data items in the network. For example, let us suppose a situation where a research project team engaged in excavation work constructs an ad hoc network on a mountain. The results obtained from the investigation may consist of various types of data such as numerical data, photographs, sounds, and videos. In this case, although it is useful to have the data that other members obtained, it seems difficult for a mobile host to have replicas of all the data.

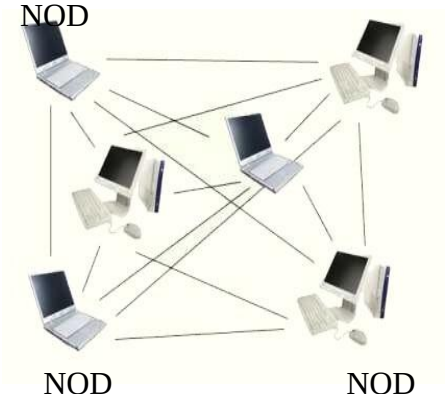

Fig 2 Ad hoc Network

Wireless ad hoc networks can be further classified by their application:

- Mobile ad hoc networks (MANETs)
- Wireless mesh networks
- Wireless sensor networks

ROUTING

Routing is the process of selecting paths in a network to forward the data packets. In packet switching networks, routing directs packet forwarding, the transit of logically addressed packets from their source toward their ultimate destination through intermediate nodes; typically hardware devices called routers, bridges, gateways, firewalls, or switches. Most routing algorithms use only one network path at a time, but multipath routing techniques enable the use of multiple alternative paths.

II. Existing Scheme

In geographic routing, nodes need to maintain up-to-date positions of their immediate neighbors for making effective forwarding decisions. Periodic broadcasting of beacon packets that contain the geographic location coordinates of the nodes is a popular method used by most geographic routing protocols to maintain neighbor positions. We contend and demonstrate that periodic beaconing regardless of the node mobility and traffic patterns in the network is not attractive from both update cost and routing performance points of view. We propose the Adaptive Position Update (APU) strategy for geographic routing, which dynamically adjusts the frequency of position updates based on the mobility dynamics of the nodes and the forwarding patterns in the network. APU is based on two simple principles: 1) nodes whose movements are harder to predict update their positions more frequently (and viceversa), and (ii) nodes closer to forwarding paths update their positions more frequently (and vice versa). Our theoretical analysis, which is validated by NS2 simulations of a well-known geographic routing protocol, Greedy Perimeter Stateless Routing Protocol (GPSR), shows that APU can significantly reduce the update cost and improve the routing performance in terms of packet delivery ratio and average end-to-end delay in comparison with periodic beaconing and other recently proposed updating schemes. The benefits of APU are further confirmed by undertaking evaluations in realistic network scenarios, which account for localization error, realistic radio propagation, and sparse network.

In the periodic beaconing scheme, each node broadcasts a beacon with a fixed beacon interval. If a node does not hear any beacon from a neighbor for a certain time interval, called neighbor time-out interval, the node considers this neighbor has moved out of the radio range and removes the outdated neighbor from its neighbor list. The neighbor time-out interval often is multiple times of the beacon interval.

Disadvantages of existing system

Position updates are costly in many ways. Each update consumes node energy, wireless bandwidth, and increases the risk of packet collision at the medium access control (MAC) layer.

Packet collisions cause packet loss which in turn affects the routing performance due to decreased accuracy in determining the correct local topology (a lost beacon broadcast is not retransmitted).

A lost data packet does get retransmitted, but at the expense of increased end-to-end delay. Clearly, given the cost associated with transmitting beacons, it makes sense to adapt the frequency of beacon updates to the node mobility and the traffic conditions within the network, rather than employing a static periodic update policy.

For example, if certain nodes are frequently changing their mobility characteristics (speed and/or heading), it makes sense to frequently broadcast their updated position. However, for nodes that do not exhibit significant dynamism, periodic broadcasting of beacons is wasteful. Further, if only a small percentage of the nodes are involved in forwarding packets, it is unnecessary for nodes which are located far away from the forwarding path to employ periodic beaconing because these updates are not useful for forwarding the current traffic.

III. Proposed scheme

NETWORK SIMULATOR

A network simulator is a software program that imitates the working of a computer network. In simulators, the computer network is typically modeled with devices, traffic etc and the performance is analyzed. Typically, users can then customize the simulator to fulfill their specific analysis needs. Simulators typically come with support for the most popular protocols in use today, such as Wireless LAN, Wi-Max, UDP, and TCP. A network simulator is a piece of software or hardware that predicts the behavior of a network, without an actual network being present. ns is an object oriented simulator, written in C++, with an OTcl interpreter as a frontend.

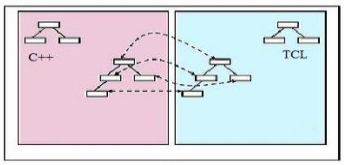

Figure 4.1 Network Simulator

The simulator supports a class hierarchy in C++ and a similar class hierarchy within the OTcl interpreter. The two hierarchies are closely related to each other; from the user's perspective, there is a one-to-one correspondence between a class in the interpreted hierarchy and one in the compiled hierarchy. The root of this hierarchy is the class TclObject. Users create new simulator objects through the interpreter; these objects are instantiated within the interpreter, and are closely mirrored by a corresponding object in the compiled hierarchy. The interpreted class hierarchy is automatically established through methods defined in the class TclClass. User instantiated objects are mirrored through methods defined in the class TclObject. There are other hierarchies in the C++ code and OTcl scripts; these other hierarchies are not mirrored in the manner of TclObject.Uses of network simulators

Network simulators serve a variety of needs. Compared to the cost and time involved in setting up an entire test bed containing multiple networked computers, routers and data links, network simulators are relatively fast and inexpensive. They allow engineers to test scenarios that might be particularly difficult or expensive to emulate using real hardware- for instance, simulating the effects of a sudden burst in traffic or a DoS attack on a network service. Networking simulators are particularly useful in allowing designers to test new networking protocols or changes to existing protocols in a controlled and reproducible environment. Network simulators simulate and then analyze the effect of various parameters on the network performance. Typical network simulators encompasses a wide range of networking technologies and help the users to build complex networks from basic building blocks like variety of nodes and links. With the help of simulators one can design hierarchical networks using various types of nodes like computers, hubs, bridges, routers, optical cross-connects, multicast routers, mobile units, etc. Various types of Wide Area Network (WAN) technologies like TCP, ATM, IP etc and Local Area Network (LAN) technologies like Ethernet, token rings etc. can all be simulated with a typical simulator and the user can test, analyze various standard results apart from devising some novel protocol or strategy for routing etc. There are a wide variety of network simulators, ranging from the very simple to the very complex. Minimally, a network simulator must enable a user to represent a network topology, specifying the nodes on the network, the links between those nodes and the traffic between the nodes. More complicated systems may allow the user to specify everything about the protocols used to handle network traffic.

NETWORK SIMULATOR 2 (NS2):

NS2 is an open-source simulation tool that runs on Linux. It is a discreet event simulator targeted at networking research and provides substantial support for simulation of routing, multicast protocols and IP protocols, such as UDP, TCP over wired and wireless (local and satellite) networks. It has many advantages that make it a useful tool, such as support for multiple protocols and the capability of graphically detailing network traffic. Additionally, NS2 supports several algorithms in routing and queuing. Queuing algorithms include fair queuing, deficit round-robin and FIFO. NS2 started as a variant of the REAL network simulator in 1989. REAL is a network simulator originally intended for studying the dynamic behavior of flow and congestion control schemes in packet-switched data network. NS2 is available on several platforms such as FreeBSD, Linux, SunOS and Solaris. NS2 also builds and runs under Windows.

Node Methods: Configuring the Node

Procedures to configure an individual node can be classified into:

1. Control functions
2. Address and Port number management, unicast routing functions
3. Agent management
4. Adding neighbors

Control functions

1. $node entry returns the entry point for a node. This is the first element which will handle packets arriving at that node. The Node instance variable, entry_, stores the reference this element. For multicast nodes, the entry point is the switch_ which looks at the first bit to decide whether it should forward the packet to the unicast classifier, or the multicast classifier as appropriate.
2. $node reset will reset all agents at the node.

Address and Port number management

1. The procedure $node id returns the node number of the node. This number is automatically incremented and assigned to each node at creation by the class Simulator method, $ns node.
2. The procedure $node agent <port> returns the handle of the agent at the specified port. If no agent at the specified port number is available, the procedure returns the null string.
3. The procedure alloc-port returns the next available port number. It uses an instance variable, np_, to track the next unallocated port number.
4. The procedure, add-route and add-routes, are used by unicast routing to add routes to populate the classifier_.
5. $node add-routes <destination id> <TclObjects> is used to add multiple routes to the same destination that must be used simultaneously in round robin manner to spread the bandwidth used to reach that destination across all links equally.

Agent management

Given an <agent>, the procedure attach{} will add the agent to its list of agents_, assign a port number the agent and set its source address, set the target of the agent to be its (i.e., the node's) entry{}, and add a pointer to the port demultiplexer at the node (dmux_) to the agent at the corresponding slot in the dmux_ classifier. Conversely, detach{} will remove the agent from agents_, and point the agent's target, and the entry in the node dmux_ to nullagent.

Tracking Neighbors

Each node keeps a list of its adjacent neighbors in its instance variable, neighbor_. The procedure add-neighbor{} adds a neighbor to the list. The procedure neighbors{} returns this list.

Creating a Tcl scenario

To define trace files with the data that needs to be collected from the simulation, we have to create these files using the command open:

```
# open the trace file
set traceFile [open out.tr w]
$ns trace-all $traceFile
# open the Nam trace file
set namFile [open out.nam w]
$ns namtrace-all $namFile
# Define the TCP agent:
```

```
set tcp [new Agent/TCP]
$ns attach-agent $n(0) $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n(1) $sink
$ns connect $tcp $sink
```

Node Configuration

Node configuration essentially consists of defining the different node characteristics before creating them. They may consist of the type of addressing structure used in the simulation, defining the network components for mobile nodes, turning on or off the trace options at Agent/Router/MAC levels, selecting the type of adhoc routing protocol for wireless nodes or defining their energy model. The node-config command would look like the following:

```
$ns_ node-config -addressType hierarchical \
        -adhocRouting AODV \
        -llType LL \
        -macType Mac/802_11 \
        -ifqType Queue/DropTail/PriQueue \
        -ifqLen 50 \
        -antType Antenna/OmniAntenna \
        -propType Propagation/TwoRayGround \
        -phyType Phy/WirelessPhy \
        -topologyInstance $topo \
        -channel Channel/WirelessChannel \
        -agentTrace ON \
        -routerTrace ON \
        -macTrace OFF \
        -movementTrace OFF
```

SIMULATION PROCEDURE

Run the script by typing at the Konsole

```
ns filename.tcl
```

On completion of the run, Trace output file "filename-out.tr" and nam output file "filename-out.nam" are created. Running filename-out.nam, the mobile nodes moving in nam window can be seen. The number of nodes is set and the percentage of active senders is determined. The active senders start informing the network about its presence and begin sending data according to the random progress method.

The finish procedure is given as

```
proc finish {} {
        $ns flush-trace
      close $f
        close $nf
      exec nam -r filename.nam &
        exit 0
          }
```

In the finish procedure, the trace file buffer is cleared and graphs are generated in the terminal in a pipelined manner. $nf is used to close the trace field. Now the animator field is generated using command

```
exec nam filename.nam
```

To run the file $ns run command is used and the tcl script is executed.

To execute the graph exec ns graph.tcl command is used.

IV. Results And Discussions

Simulation Scenario

The proposed method addresses the congestion issues considering delay, packet loss and routing overhead. In order to evaluate the performance of the multipath video multicasting and to compare it with conventional multicasting, the below parameters are configured in the network simulator.

Packet Size	:	2000 bytes
No. of Nodes	:	100
Protocol Used	:	AODV
Dimension	:	1000*1000

Channel Type	:	Wireless channel IEEE 802.11
Queue Type	:	Drop Tail/PriQueue
Antenna	:	Omni Antenna
Protocol	:	TCP
Mobility	:	10 m/s

SIMULATION

In this simulation the data packet is transmitted in multipath to multiple receivers. The initial position of nodes are visible in the below fig 5

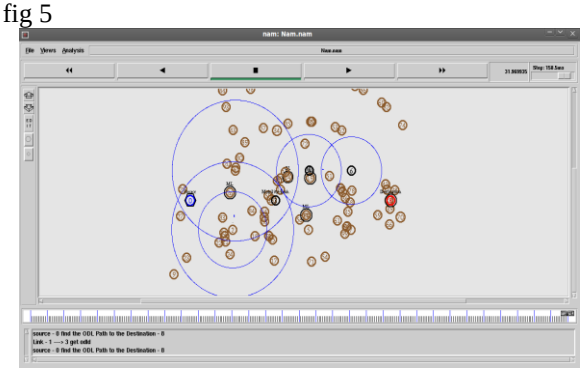

Fig 5 Adaptive routing of nodes

SIMULATIONRESULTS

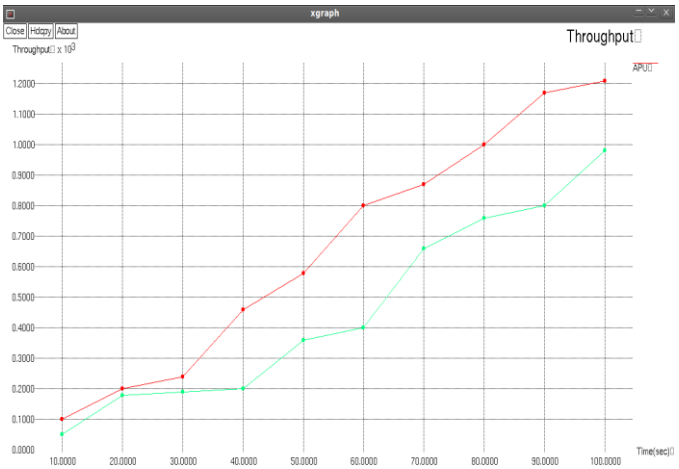

Fig 5.2 Simulation showing throughput

The above simulation shows the throughput of the adaptive position updating and periodic broadcasting where throughput is high for APU since number of successful transmissions are increased.

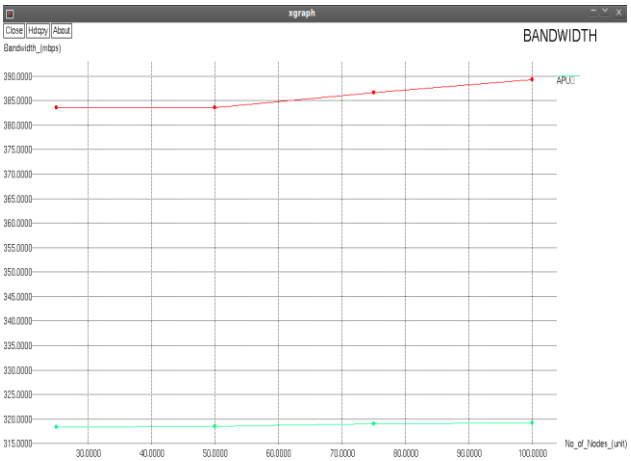

Fig 5.3 Simulation showing bandwidth

The above simulation shows the bandwidth usage of both periodic broadcasting and the adaptive position update. It is seen that the bandwidth is effectively utilised by APU.

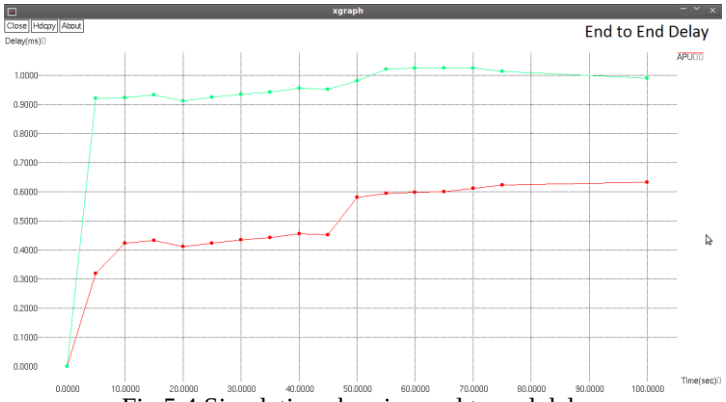

Fig 5.4 Simulation showing end to end delay

The above simulation shows end to end delay comparison of periodic broadcasting and adaptive position updating of the nodes. It is seen that the delay is more for periodic broadcasting.

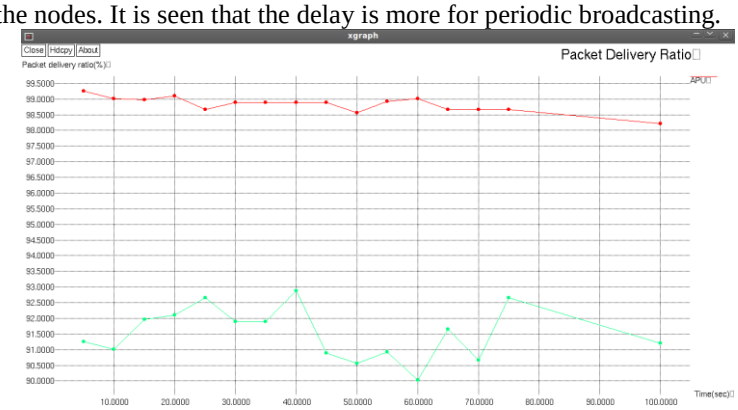

Fig 5.5 Simulation showing packet delivery ratio

The above simulation shows packet delivery ratio of the periodic broadcasting and adaptive position updating with respect to time, it is seen that packet delivery ratio is efficient for APU

V. Conclusion

In this project the need to adapt the beacon update policy employed in geographic routing protocols to the node mobility dynamics and the traffic load is identified. We proposed the Adaptive Position Update strategy to address these problems. The APU scheme employs two mutually exclusive rules. The MP rule uses mobility prediction to estimate the accuracy of the location estimate and adapts the beacon update interval accordingly, instead of using periodic beaconing. The ODL rule allows nodes along the data forwarding path to maintain an accurate view of the local topology by exchanging beacons in response to data packets that are overheard from new neighbours. The beacon overhead and local topology accuracy of APU and validated the analytical model with the simulation results.

References

[1]. Seungjoon Lee Bobby Bhattacharjee Suman Banerjee (2004), 'Efficient Geographic Routing in Multihop Wireless Networks' Technical Report CS-TR 4625, University of Maryland

[2]. Marc Heissenbuttel , Torsten Braun, Markus Walchli, Thomas Bernoulli (2005), 'Evaluating the limitations of and alternatives in beaconing' Tech. Rep. CS-2003-11

[3]. Young-Bae Ko and Nitin H. Vaidya (2006), 'Location-Aided Routing (LAR) in mobile ad hoc networks' Department of Computer Science, Texas A&M University, TX 77843-3112, USA

[4]. David B. Johnson David A. Maltz Josh Broch (2006)' DSR: The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad Hoc Networks' Computer Science Department Carnegie Mellon University Pittsburgh, PA 15213-3891

[5]. Manel Guerrero Zapata (2007), 'Secure Ad hoc On-Demand Distance Vector Routing' Mobile Networks Laboratory Nokia Research Center

[6]. Marc Heissenbuttel, Torsten Braun, Thomas Bernoulli, Markus Walchli (2008) 'BLR: beacon-less routing algorithm for mobile ad hoc networks' Institute of Computer Science and Applied Mathematics, University of Bern, Bern, Switzerland