FINAL REPORT

INRODUCTION

Ad Hoc networks applications are developed for providing intelligent and safe transport systems. Emergency warning for public safety is one application that is highly time-critical and requires an intelligent broadcast mechanism to distribute warning messages. Two major problems to design a broadcast protocol for VANETs are: (1) The broadcast storm problem (2) The Disconnected Node Problem.

Broadcast storm problem is problem which occurs when there is a dense traffic condition on roads this usually occur at the peak hours of traffic. i.e, when multiple nodes attempt to transmit at the same time, this causes packet loss and extra delay at the medium access control (MAC) layer. The Disconnect network problem occurs when the number of nodes in the area to help disseminate the broadcast message is not sufficient. Actually these problems are well known in Mobile ad hoc networks each problem has been treated separately so far, and also most of the algorithms were developed to solve either broadcast storm problem or disconnected network problem .The main goal of this project is to design a distributed broadcast protocol, that can solve both broadcast storm and maintain network connectivity in disconnected networks. So in this project I have installed and tried to use different Simulators like Omnet++, NS2, Sumo and also veins.

From my perspective DV CAST is very useful protocol designed for broadcast storm problem and disconnected node problem. Because any protocol before just used to solve either broadcast problem or disconnected node problem but DV CAST solves both of them. This protocol had been tested only on highways and it’ll be more useful if it is tested in Urban and sub urban areas.

BACKGROUND

There are a number of routing protocols designed to improve routing in sparsely connected MANETs. Data mules are proposed to function as mobile messengers that collect data from sensors and deliver to a virtual backbone in sensor networks. Role-based multicast. Epidemic routing relies on mobile nodes to exchange the data they possess whenever they encounter new neighbors. multi-copy Spray and Wait. Single-copy are shown to be efficient alternatives for message delivery. However,most of these reviews concentrate on utilizing arbitrary way point display node mobility has fawer true limitations and may not be suitable for VANET considers, as typical road topology is a great deal more compelled and has an exceptionally all around characterized structure. Distance Based Relay Selection (DBRS) is a straightforward system to spread data in a vehicular network. After getting a package, the vehicle keeps it for a period interim which is relative to the backwards of the separation until the transmitter vehicle. Consequently, to spread data is desirable over utilize the more far off vehicles from the transmitting vehicle. The vehicles that listen to the broadcast package which is as of now modified to broadcast, scratch off their transmissions to stay away from the broadcast storm. This approach is solid as in it might evade the broadcast storm, in any case, it might get two issues: (i) the delay might be high, for there is no assurance of the presence of vehicles near the correspondence run (those which transmit with smaller delay), and (ii) the scope might be lessened, since the vehicles will wipe out their transmissions on an aimless way by listen to the transmission of a same package. Adaptive approach for Information Dissemination (AID) is a distributed and versatile approach for VANETs information dissemination. In such approach, the vehicle chooses on the off chance that it ought to or not transmit a package, contingent upon the number of times that it gets similar information package in a given period of time. In dense systems, for example, a few vehicles may choose to dispose a package , since it was at that point sent by a few vehicles, and this lessens the communicate storm issue. Nevertheless, the AID convention does not manage the network partition issue and introduces low execution to perform information spread on sparse network systems. Distributed Vehicular Broadcast (DV-CAST) is a broadcast protocol which only depends on the local topology information to undertake the data dissemination. It has some characteristics such as: (i) robustness against different vehicle traffic conditions and (ii) dependence on the information about the neighbor’s location distant one hop of each vehicle through the use of periodic discovery messages. The DV-CAST consists of three main components: neighborhood detection, broadcast suppression and store-carry-forward. The neighborhood detection mechanism estimates the local topology monitoring messages of periodic discovery as from a one hop distant neighbor. The motivation to use the local topology information in the DV-CAST is to minimize the additional network overhead and maintain the minimum complexity of the protocol. For the broadcast storm suppression the DV-CAST uses the weighed p-persistence scheme and in order to deal with the network partitions, the DV-CAST uses the store-carry-forward scheme.

ALGORITHM FLOWCHART

This fig. illustrates the detailed implementation of DV-CAST. In order to handle the broadcast message properly, we propose that each vehicle follow two basic routing rules:

• If DFlg is set to 1, the vehicle should ignore any duplicate broadcast or follow the diagram if the message is received for the first time.

• If DFlg is set to 0, the vehicle is a relay node and should follow the routing diagram shown. This is because in a very sparse network environment, a certain relay vehicle may have to help store-carry-forward the same message more than once. Depending on the level of local connectivity the vehicle experiences, we propose three different courses of action the vehicle should follow in order to properly handle the broadcast packet.

Broadcast Suppression

IDLE

WAIT II

ODC=?

MCD=?

WAIT I

Rebroadcast

DFlg=?

Packet arrival

1 0

1 0

Pkt timer expires

Hello Pkt arrives from New ODN

Hello Pkt arrives From ODN/ODC=1 or MDC=1

1 0

Rebroadcast

Pkt arrival <HOP=N+2>or Pkt Timer Expires

MDC: Message Direction Connectivity

ODC: Opposite Direction Connectivity

DFlg: Destination Flag

SUMO

SUMO is an open source traffic simulation package including net import and demand modeling components. We describe the current state of the package as well as future developments and extensions. SUMO helps to investigate several research topics e.g. route choice and traffic light algorithm or simulating vehicular communication. Therefore the framework is used in different projects to simulate automatic driving or traffic management strategies.

1. Vehicular Communication

The probably most popular application for the SUMO suite is modeling traffic within research on V2X – vehicleto-vehicle and vehicle-to-infrastructure – communication. Here, usually SUMO is coupled to an external communication simulation, such as ns2 or ns3 using TraCI. For obtaining a functioning environment for the simulation of vehicular communications, an applications instance which models the V2X application to simulate is needed. Additionally, a synchronization and message exchange mechanism has to be involved. Here, the TraNS extensions to ns2 were responsible for synchronizing the simulators and the application had also to be modeled within ns2. TraNS was the major reason for making TraCI open source. With the end of the projects the original TraNS authors were working on, TraNS itself got no longer maintained and works with a very old SUMO version only, as the TraCI API was changed.

1. Route Choice and dynamic Navigation

The assignment of proper routes to a complete demand or a subset of vehicles is investigated both, on a theoretical base and as new applications. On the theoretical level, the interest lies in a proper modeling of how traffic participants choose a route – a path through the given road network – to their desired destination. As the duration to pass an edge of the road graph highly depends on the numbers of participants using this edge, the computation of routes through the network under load is a crucial step in preparing large-scale traffic simulations. Due to its fast execution speed, SUMO allows to investigate algorithms for this “user assignment” or “traffic assignment” on a microscopic base. Usually, such algorithms are investigated using macroscopic traffic flow models, or even using coarser road capacity models which do not resemble dissolving road congestions.

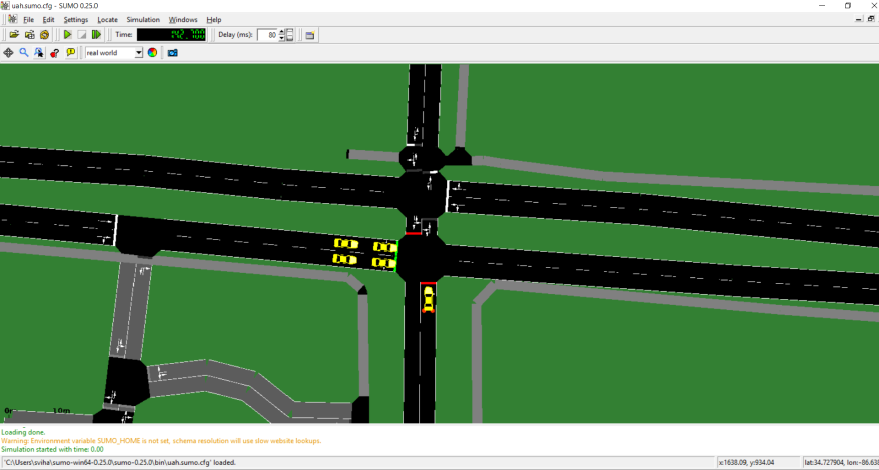
1. Traffic Light Algorithms

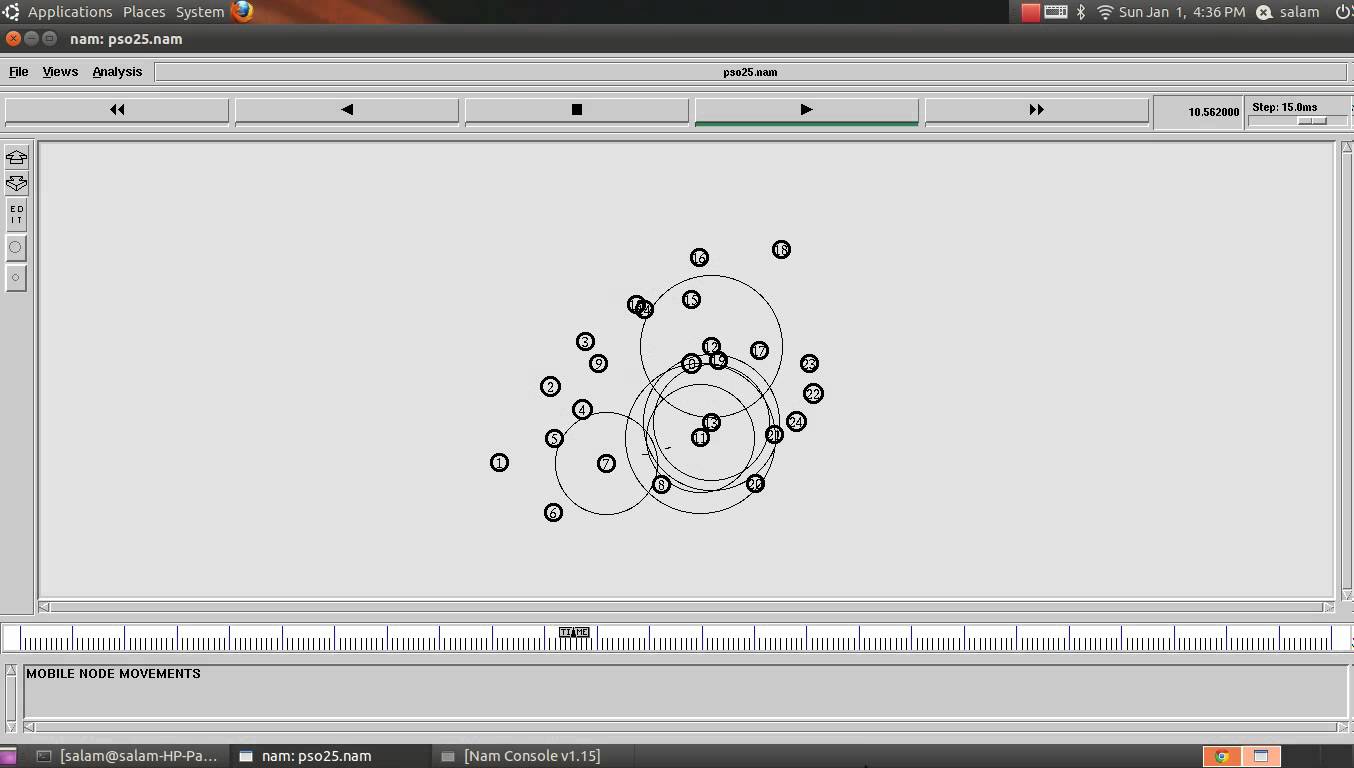
The evaluation of developed traffic light programs or algorithms for making traffic lights adaptable to the current traffic is one of the main applications for microscopic traffic flow simulations. As SUMO’s network model is relatively coarse compared to commercial applications as Vissim, SUMO is usually not used by traffic engineers for evaluating real-life intersections. Still, SUMO’s fast execution time and its open TraCI API for interaction with external applications make it a good candidate for evaluating new traffic control algorithms, both for controlling a single intersection and for net-wide investigations. By distinguishing different vehicle types, SUMO also allows the simulation of public transport or emergency vehicle prioritization at intersections.

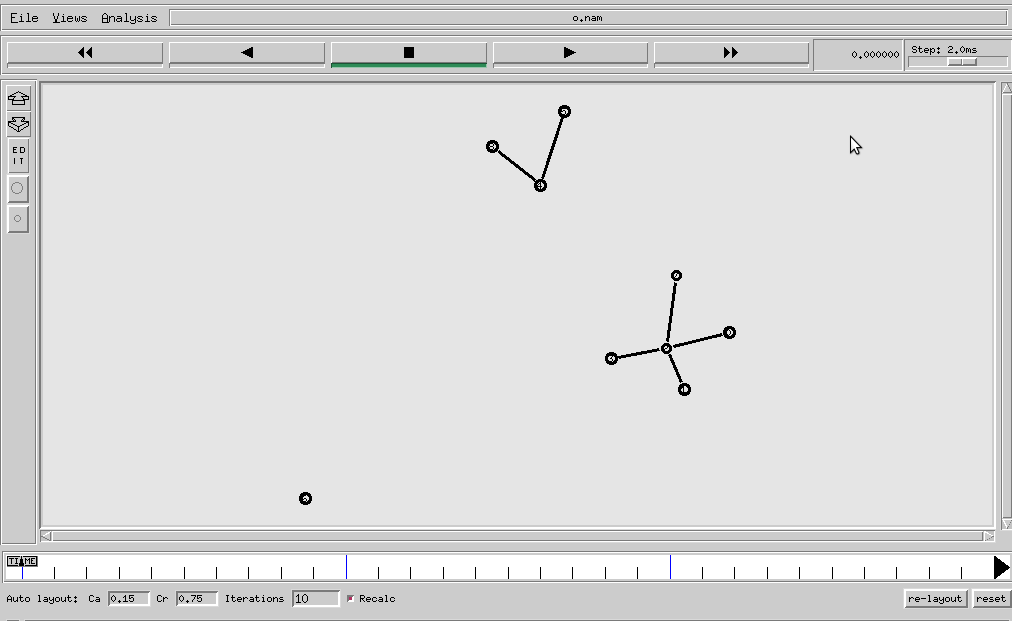
1. Network Editor

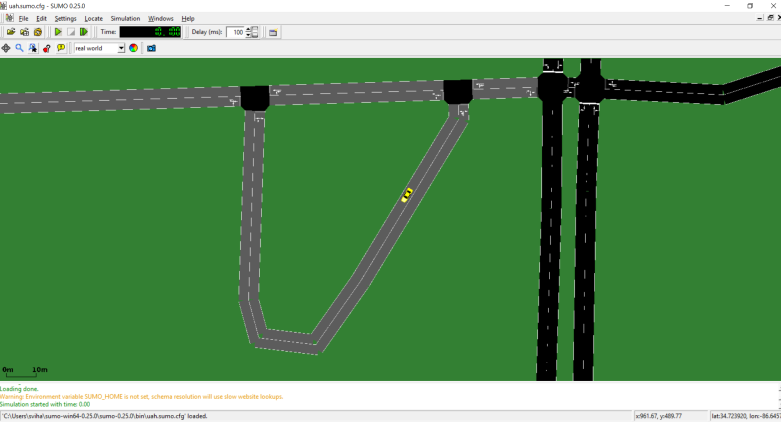
A graphical network editor is implemented. It allows to set up a complete road network for SUMO, including all needed information, such as correct lane number, speed limits, connections across intersections, and traffic lights. For now, this tool is not part of the open source package, but is held for internal purposes only.

SIMULATION

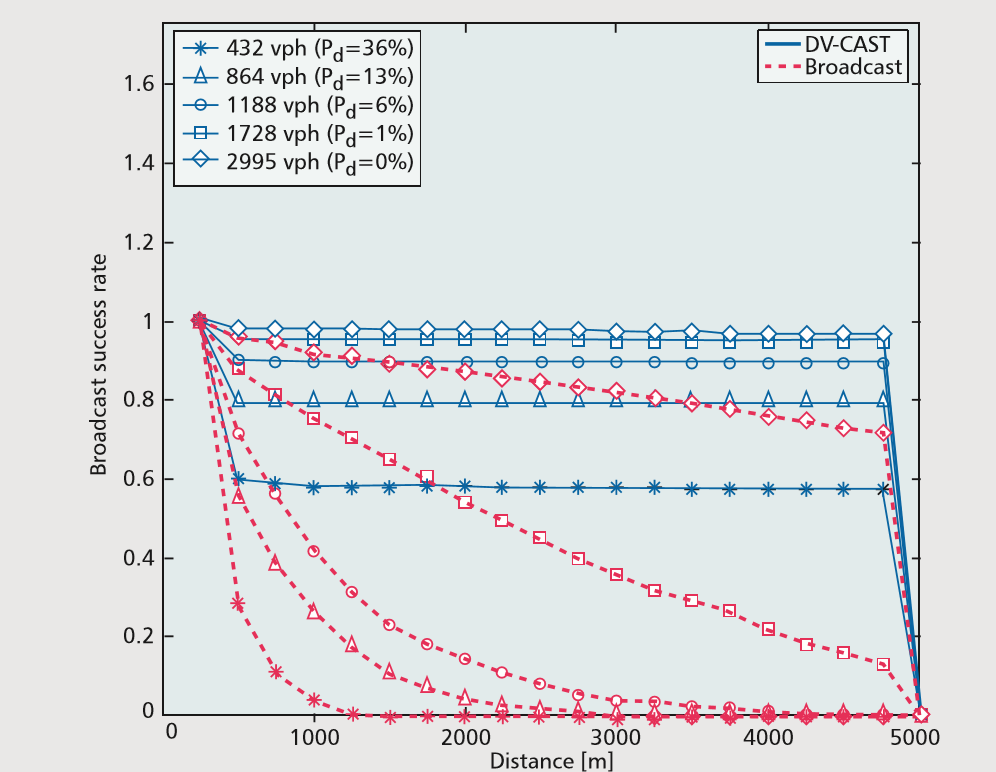


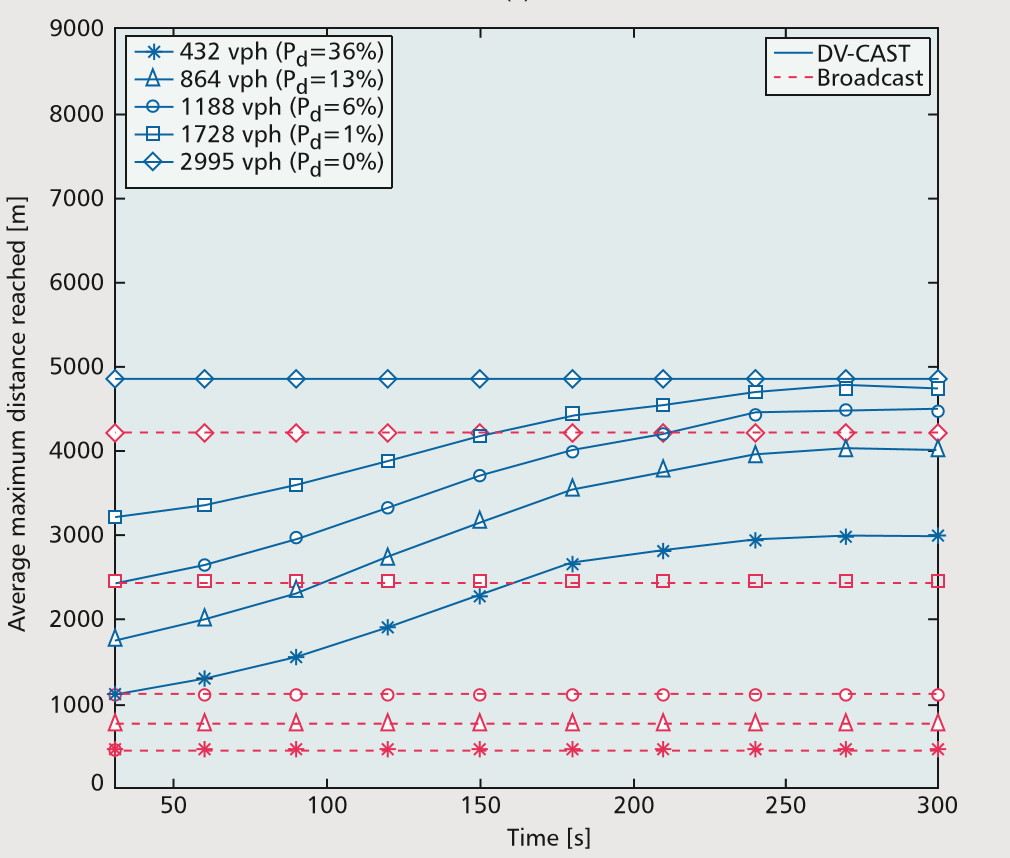






EXPECTED OUTPUTS



CONCLUSION

In this project I tried toscale performance in terms of reliability, efficiency, and scalability of the DV-CAST protocol on highways under multiple traffic conditions. Since the protocol is designed to address how to deal with extreme conditions such as heavy traffic during rush hours, very light traffic during certain hours of the day (e.g., midnight to 4 a.m. in the morning), and low market penetration rate of cars using DSRC technology, the simulation results show that the DVCAST performs well in every aspect considered and is robust against various extreme traffic conditions. For future work, it would be interesting to see how much of the underlying design principles of DV-CAST would also be applicable to urban areas. At a high level, it seems clear that large cities might have a very different and much richer topology (e.g., Manhattan Street type of topologies) than highways. In this sense, one could consider DV-CAST as an instance (or a subset) of designing a distributed vehicular broadcasting protocol that would work anywhere (urban, suburban, and rural areas). Our current research efforts are focused on extending vehicular broadcasting to urban areas.

REFERENCES

[1] R. C. Shah et al., “Data MULEs: Modeling and Analysis of a Three-tier Architecture for Sparse Sensor Networks,” Elsevier Ad Hoc Net. J., vol. 1/2–3, Sept. 2003, pp. 215–333.

[2] A. Vahdat and D. Becker, “Epidemic Routing for Partially Connected Ad Hoc Networks,” tech. rep. no. cS200006, Duke University, Apr. 2000.

[3] L. Briesemeister and G. Hommel, “Role-Based Multicast in Highly Mobile but Sparsely Connected Ad Hoc Networks,” Proc. ACM MobiHoc, Boston, MA, Aug. 2000, pp. 45–50

[4] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, “Single-Copy Routing in Intermittently Connected Mobile Networks,” 1st IEEE SECON ‘04, Oct. 2004, pp. 235–44.

[5] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, “Spray and Wait: An Efficient Routing Scheme for Intermittently Connected Mobile Networks,” Proc. 2005 ACM SIGCOMM Wksp. Delay-Tolerant Net., Philadelphia, PA, Aug. 2005, pp. 252–59.

[6] DV-CAST: A Distributed Vehicular broadcast protocol for vehicular ad hoc networks by ozan k. tonguz and nawaporn wisitpongphan, carnegie mellon university fan bai, general motors corporation.

[7] N. Wisitpongphan et al., “Routing in Sparse Vehicular Ad Hoc Wireless Networks,” IEEE JSAC, vol. 25, no. 8, Oct. 2007, pp. 1538–56.

[8] O.K. Tonguz, W. Viriyasitavat, and F. Bai, “Modeling Urban Traffic: A Cellular Automata Approach,” IEEE Commun. Mag., vol. 47, no. 5, May 2009.

[9] G. Korkmaz, E. Ekici, and F. Ozguner, “An Efficient Fully Ad-Hoc multihop Broadcast Protocol for Inter-Vehicular Communication Systems,” Proc. IEEE ICC, Istanbul, Turkey, June 2006, pp. 423–28.

[10] G. Korkmaz et al., “Urban multihop Broadcast Protocol for Inter-Vehicle Communication Systems,” Proc. ACM VANET, Philadelphia, PA, Oct. 2004.

[11] J. Zhao and G. Cao, “VADD: Vehicle-Assisted Data Delivery in Vehicular Ad Hoc Networks,” Proc. IEEE INFOCOM, Apr. 2006, pp. 1–12.

[12] Z. D. Chen, H. T. Kung, and D. Vlah, “Ad Hoc Relay Wireless Networks over Moving Vehicles on Highways,” Proc. ACM MobiHoc, Long Beach, CA, Oct. 2001, pp. 247–50.

[13] SUMO – Simulation of Urban MObility by michael behrisch, laura bieker, jakob erdmann, daniel krajzewicz institute of transportation systems german aerospace center.

APPENDIX

TCL program:

set val(chan)           Channel/WirelessChannel    ;# channel type

set val(prop)           Propagation/TwoRayGround   ;# radio-propagation model

set val(netif)          Phy/WirelessPhy            ;# network interface type

set val(mac)            Mac/802\_11                 ;# MAC type

set val(ifq)            Queue/DropTail/PriQueue    ;# interface queue type

set val(ll)             LL                         ;# link layer type

set val(ant)            Antenna/OmniAntenna        ;# antenna model

set val(ifqlen)         50                         ;# max packet in ifq

set val(nn)             41                        ;# number of mobilenodes

set val(rp)             AODV                      ;# routing protocol

set val(x)                4706                     ;# X dimension of topography

set val(y)              3001                  ;# Y dimension of topography

set val(stop)           150                        ;# time of simulation end

set ns          [new Simulator]

set tracefd       [open uah.tr w]

set windowVsTime2 [open win.tr w]

set namtrace      [open uah.nam w]

$ns trace-all $tracefd

$ns namtrace-all-wireless $namtrace $val(x) $val(y)

# set up topography object

set topo       [new Topography]

$topo load\_flatgrid $val(x) $val(y)

create-god $val(nn)

#

#  Create nn mobilenodes [$val(nn)] and attach them to the channel.

#

# configure the nodes

        $ns node-config -adhocRouting $val(rp) \

             -llType $val(ll) \

             -macType $val(mac) \

             -ifqType $val(ifq) \

             -ifqLen $val(ifqlen) \

             -antType $val(ant) \

             -propType $val(prop) \

             -phyType $val(netif) \

             -channelType $val(chan) \

             -topoInstance $topo \

             -agentTrace ON \

             -routerTrace ON \

             -macTrace OFF \

             -movementTrace ON

    for {

set i 0

} {

$i < $val(nn)

} {

 incr i

} {

        set node\_($i) [$ns node]

        $node\_($i) set X\_ [ expr 10+round(rand()\*480) ]

        $node\_($i) set Y\_ [ expr 10+round(rand()\*380) ]

        $node\_($i) set Z\_ 0.0

}

    for {

set i 0

} {

$i < $val(nn)

} {

 incr i

} {

        $ns at [ expr 15+round(rand()\*60) ] "$node\_($i) setdest [ expr 10+round(rand()\*480) ] [ expr 10+round(rand()\*380) ] [ expr 2+round(rand()\*15) ]"

}

# Generation of movements

# $ns at 10.0 "$node\_(0) setdest 250.0 250.0 3.0"

# $ns at 15.0 "$node\_(1) setdest 45.0 285.0 5.0"

# $ns at 70.0 "$node\_(2) setdest 480.0 300.0 5.0"

# $ns at 20.0 "$node\_(3) setdest 200.0 200.0 5.0"

# $ns at 25.0 "$node\_(4) setdest 50.0 50.0 10.0"

# $ns at 60.0 "$node\_(5) setdest 150.0 70.0 2.0"

# $ns at 90.0 "$node\_(6) setdest 380.0 150.0 8.0"

# $ns at 42.0 "$node\_(7) setdest 200.0 100.0 15.0"

# $ns at 55.0 "$node\_(8) setdest 50.0 275.0 5.0"

# $ns at 19.0 "$node\_(9) setdest 250.0 250.0 7.0"

# $ns at 90.0 "$node\_(10) setdest 150.0 150.0 20.0"

# Set a TCP connection between node\_(2) and node\_(8)

set tcp [new Agent/TCP/Newreno]

$tcp set class\_ 2

set sink [new Agent/TCPSink]

$ns attach-agent $node\_(2) $tcp

$ns attach-agent $node\_(8) $sink

$ns connect $tcp $sink

set ftp [new Application/FTP]

$ftp attach-agent $tcp

$ns at 10.0 "$ftp start"

set tcp [new Agent/TCP/Newreno]

$tcp set class\_ 2

set sink [new Agent/TCPSink]

$ns attach-agent $node\_(5) $tcp

$ns attach-agent $node\_(0) $sink

$ns connect $tcp $sink

set ftp [new Application/FTP]

$ftp attach-agent $tcp

$ns at 10.0 "$ftp start"

# Printing the window size

proc plotWindow {

tcpSource file

} {

global ns

set time 0.01

set now [$ns now]

set cwnd [$tcpSource set cwnd\_]

puts $file "$now $cwnd"

$ns at [expr $now+$time] "plotWindow $tcpSource $file"

}

$ns at 10.1 "plotWindow $tcp $windowVsTime2"

# Define node initial position in nam

for {

set i 0

} {

$i < $val(nn)

} {

 incr i

} {

 # 30 defines the node size for nam

$ns initial\_node\_pos $node\_($i) 30

}

# Telling nodes when the simulation ends

for {

set i 0

} {

$i < $val(nn)

} {

 incr i

} {

    $ns at $val(stop) "$node\_($i) reset";

}

# ending nam and the simulation

$ns at $val(stop) "$ns nam-end-wireless $val(stop)"

$ns at $val(stop) "stop"

$ns at 150 "puts \"end simulation\" ; $ns halt"

proc stop {

} {

   global ns tracefd namtrace

    $ns flush-trace

    close $tracefd

    close $namtrace

}

$ns run