Improving AODV route discovery through GPS coordinates

VASIL HNATYSHIN[*] and HRISTO ASENOV[+]

[*]Department of Computer Science
Rowan University
Glassboro, NJ 08028
E-mail: hnatyshin@rowan.edu

[+]Department of Computer and Information Science
University of Delaware
Newark, DE 19716
E-mail: hasenov@udel.edu

Outline

- Problem Statement
  - **A**d-hoc **O**n Demand **D**istance **V**ector (AODV)Routing
  - **L**ocation **A**ided **R**outing (LAR)
- Geographical AODV (GeoAODV)
- OPNET Modeler
- Implementation of GeoAODV
- Simulation study
- Summary and Conclusions

Related Work: AODV

- Ad-hoc On Demand Distance Vector Routing
  - Routing protocol for mobile ad-hoc networks
  - Reactive protocol – initiates route discovery only when needed
  - Uses sequence numbers to ensure the freshness of routes
  - Generates loop-free routes
  - Self-starting
  - Scales to large numbers of mobile nodes

AODV Path Discovery

- AODV Path Discovery
  - Initiated only when the route to destination is needed
  - Uses request/reply cycle to discover route to destination
  - Uses request (RREQ) flooding (broadcast) to discover the route to destination
  - Reply (RREP) is unicast back to the source through reverse path
  - Source transmits data once the route is found

AODV Path Discovery

- Source broadcasts RREQ if route to destination is unknown
- RREQ message
  - *Source Address*
  - *Source Sequence Number* – freshness of the reverse route to source
  - *Broadcast ID* – unique ID for RREQ from this source node
  - *Destination Address*
  - *Destination Sequence Number* – freshness of the route to destination required by the source
  - *Hop count* – number of hops visited, i.e., path cost

RREQ arrivals

- Determine if RREQ is a duplicate (i.e., *source address* and *broadcast id* have been seen before)
  - If yes then discard RREQ
- Check if have path to destination or is destination itself
  - If no, then
    - Record reverse path information: source IP address, destination IP addresses, broadcast ID, source sequence number, and expiration time for reverse path, IP address of the neighboring node from which RREQ arrived
    - Re-broadcast RREQ

RREQ arrivals

- If node has path to destination or is destination itself
  - Check if path to destination is fresh (i.e., compare destination sequence numbers in RREQ and in AODV routing table)
    - If yes, then unicast RREP back to source by sending it to the node from which RREQ arrived
    - If no, then set-up reverse path and rebroadcast RREQ

Route Reply (RREP)

- RREP message
  - *Source address*
  - *Destination address*
  - *Destination sequence number* – freshness of path to destination
  - *Hop count* – value copied from RREQ, i.e., route cost
  - *Lifetime* – how long the route is considered valid

RREP Arrivals

- First RREP arrives at intermediate node
  - Set-up forward path
    - Record the IP address from which RREP arrived
    - Update source and destination timeout values
    - Updated destination sequence number
  - Send RREP to neighboring node on the path to the source (i.e., via reverse path)

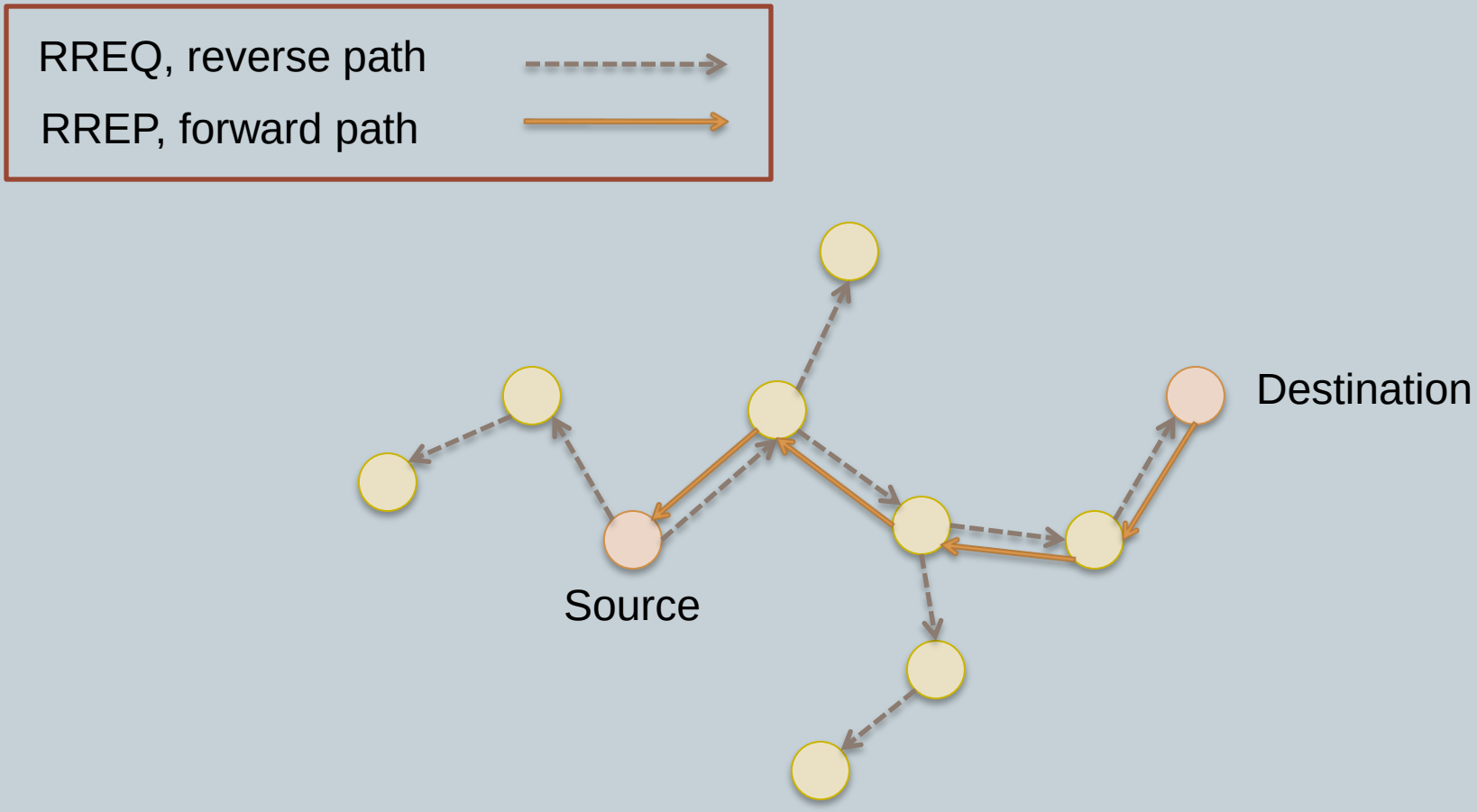
RREP Arrivals

- Duplicate RREP arrives at intermediate node
  - **If** destination sequence number of RREP is greater than that or the recorded route **or**
  - **If** destination sequence number of RREP is the same but the hop count (i.e., cost) is smaller
  - **Then** update forward route routing table entry and send RREP
  - **Otherwise**, discard RREP

RREP Arrival at Source

- RREP arrives at source
  - Update its routing table
  - Start sending application data
  - Update routing table if RREP with a better route arrives at later time
- If a node does not receive RREP then its reverse path entry expires and is discarded after certain time

Path Discovery Example



Other AODV features

- Local connectivity
  - Neighboring nodes periodically exchange HELLO messages
  - Information about neighboring nodes is saved in the routing table
- Expanding ring search
  - Source sets TTL field in RREQ's IP header to TTL_START
  - If path discovery fails to find route to destination, then increment TTL field by TTL_INCREMENT
  - Repeat until route is found or TTL = NET_DIAMETER fails to find a route after certain number of attempts.

Location Aided Routing

- AODV route discovery problem
  - Too many control messages traveling through the network
- Solution idea:
  - Use location information to limit the number of nodes participating in route discovery
  - Perform “limited” flooding within the area that is likely to contain the route to destination

Location Aided Routing

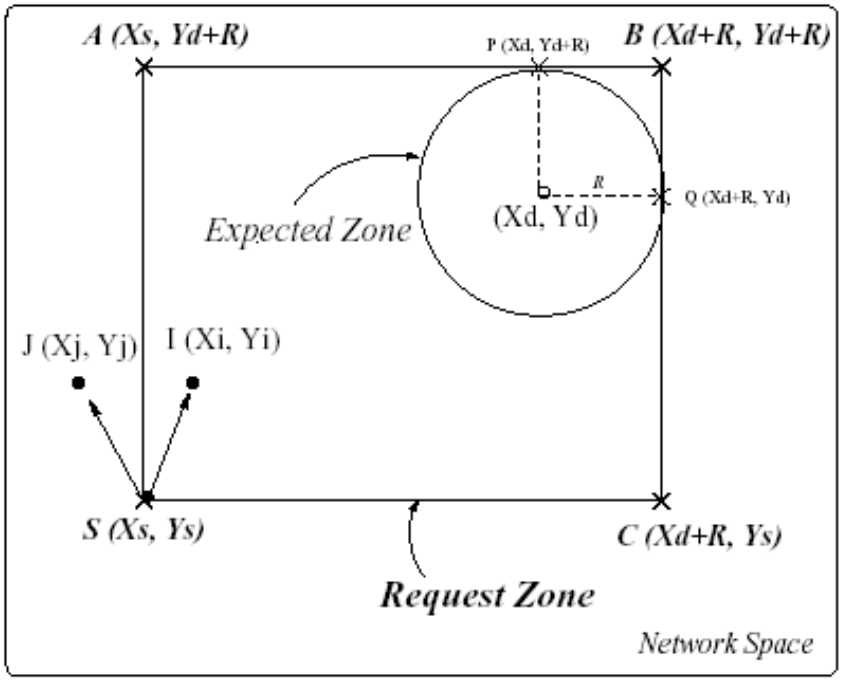
- LAR Assumptions
  - Each node knows its location
  - Source node knows destination's location **L** at time $\mathbf{t_o}$
- **Expected Zone** – an area where destination is expected to be located.
- If destination moves with speed **v** then
  - at time $\mathbf{t_1}$ destination's expected zone is a circle centered in location **L** with radius $\mathbf{v(t_1-t_o)}$

Location Aided Routing

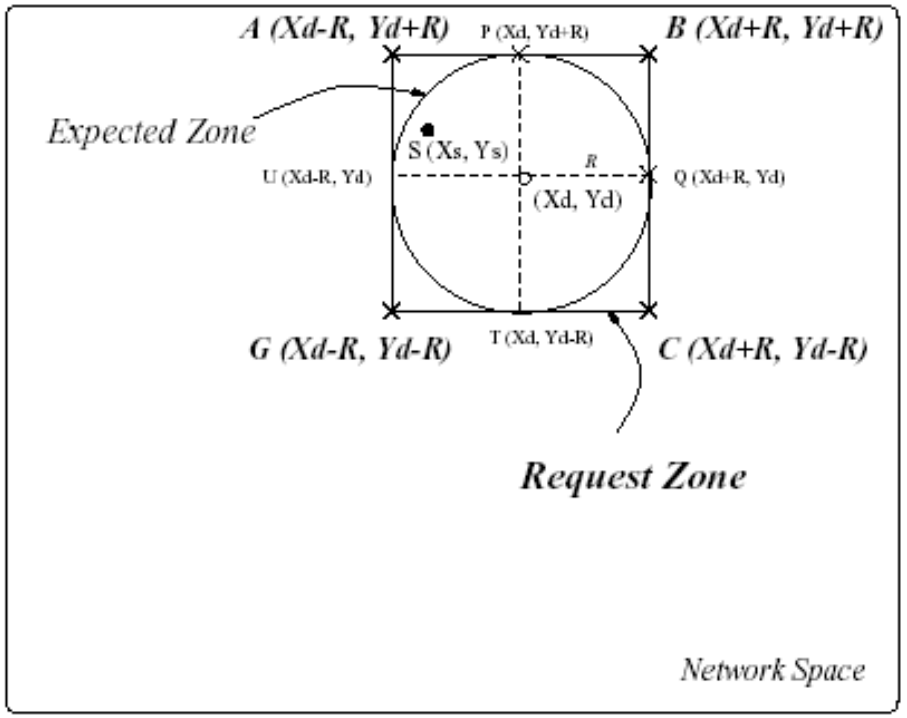
- **Request Zone** is an area which includes expected zone and likely the path from source to destination
- Only nodes within the request zone forward RREQ messages
  - Limits the number of AODV control messages
  - No guarantee that path to destination consists only on nodes within request zone
  - If path is not found then request zone is expanded
- LAR variations differ in requested zone definitions

Location Aided Routing

- LAR version #1



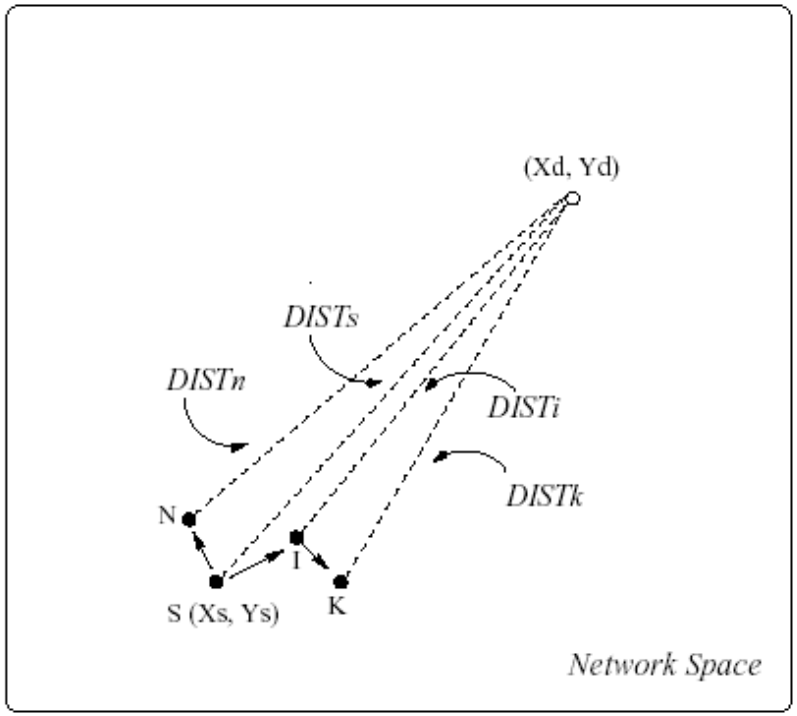
(a) Source node outside the Expected Zone



(b) Source node within the Expected Zone

Location Aided Routing

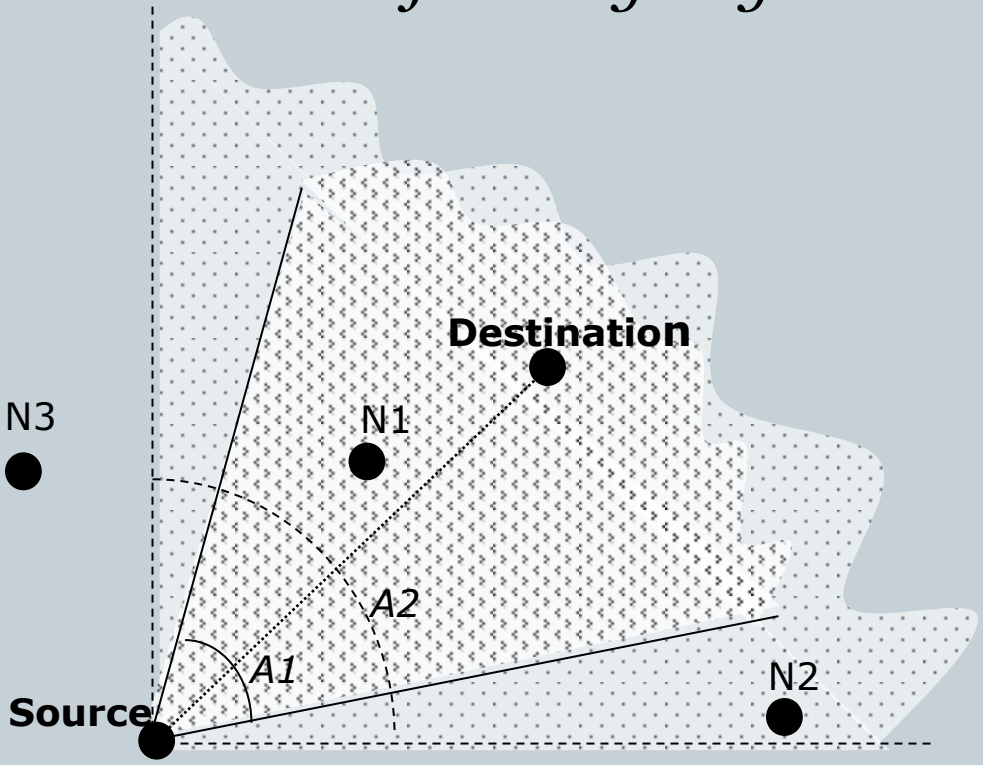
- LAR version #2



- **S** does not need to know **D**'s velocity
- **S** knows the location (X_d, Y_d) of node **D**
- **S** calculates distance **DISTs** between **S** and **D** and includes its value in RREQ
- RREQ arrives at **I**:
  - **I** computes **DISTi**, distance between **I** and **D**
  - For some parameter **δ**,
- If (**DISTs + δ ≥ DISTi**)
  - Then **I** replaces **DISTs** with **DISTi** in RREQ and re-broadcasts RREQ
  - Otherwise, **I** discards RREQ

Geographical Routing

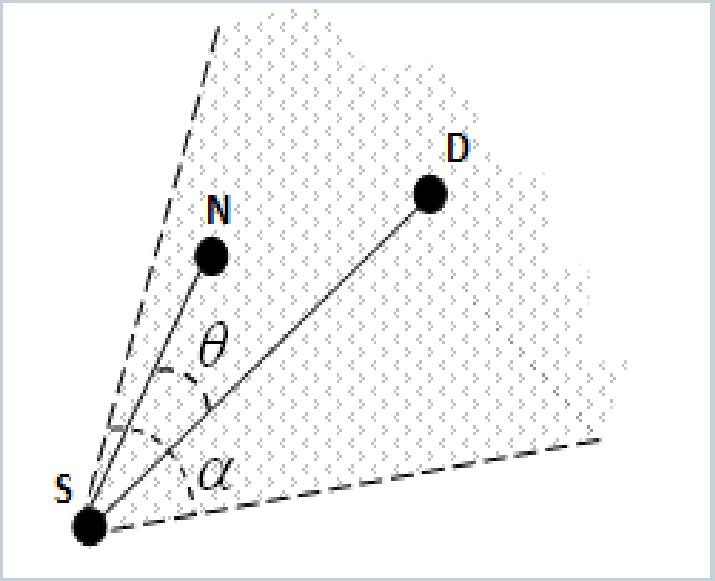
- Request zone is defined as a cone shaped area
- The size of request zone is controlled via *flooding angle*
- Flooding angle is increased if previous round of route discovery fails to find path to destination
- Eventually, GeoAODV may morph into AODV



Geographical Routing

- RREQ arrives at Intermediate node **N**
  - **N** computes angle $\boldsymbol{\theta}$
- If $\boldsymbol{2\theta < \alpha}$ then rebroadcast RREQ
- Else discard RREQ

$$\Theta = \cos^{-1}\left(\frac{\overrightarrow{SD} \bullet \overrightarrow{SN}}{|SD| \times |SN|}\right)$$



SD – source and destination vector

SN – source and neighbor vector

θ – angle between vectors ***SD*** and ***SN***

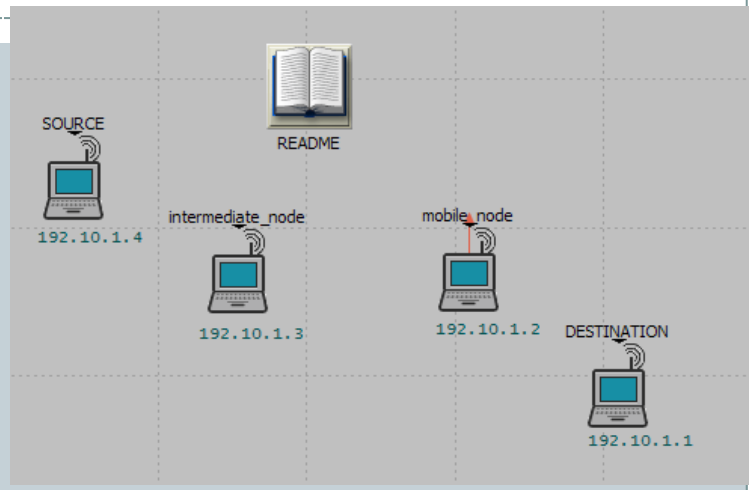
α – flooding angle

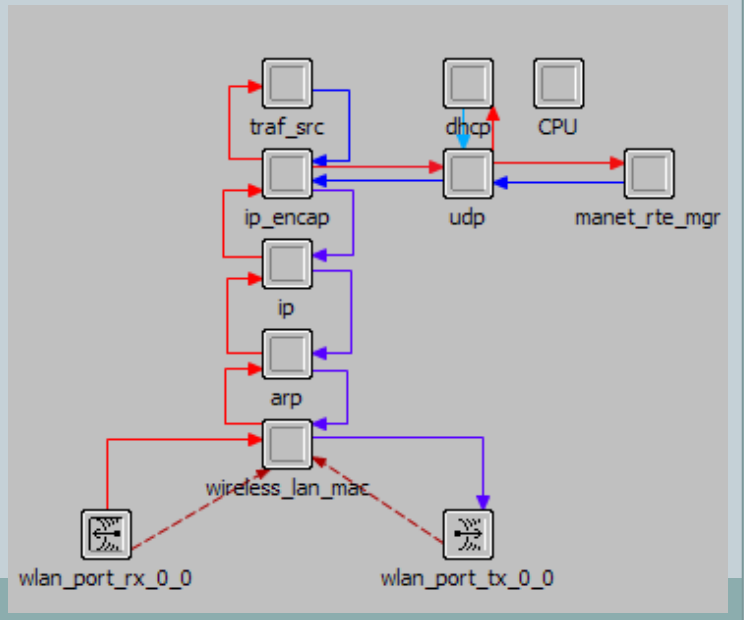
OPNET Modeler

- OPNET Modeler is network simulation software
- System models are represented via
  - C programming language
  - State transition diagrams
- Standard models are simple to use
- Developing custom model could be challenging
  - Huge amount of code spread into numerous files
  - Difficult to identify where code should be updated

OPNET Architecture

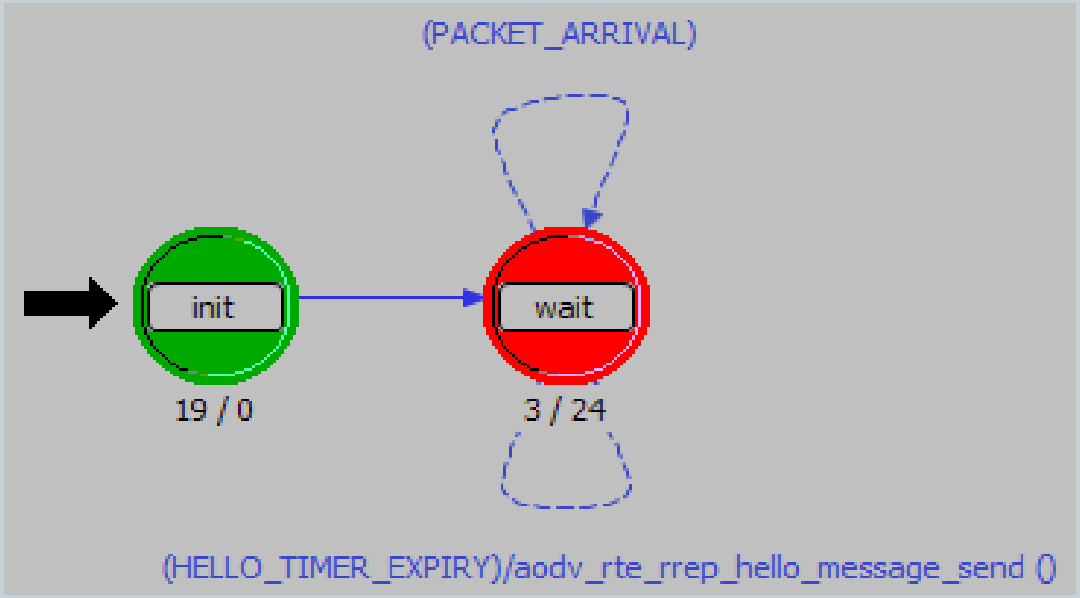
- Network Domain
  - Describes a complete simulated system
- Node Domain
  - Represents a single system device
- Process Domain
  - models a single protocol or technology
  - Process models
    - State Transition Diagram
    - C code
    - External code files





AODV Implementation

- ***init*** state – initializes process model
- ***wait*** state – idles until packet arrival or timer expiry
- The type of arriving packet controls processing
  - Application
  - RREQ
  - RREP
  - HELLO
  - RERR



Key AODV data structures

- All data structures are implemented as C structs
- ***AodvT_Packet_Option***
  - Packet Type
  - Pointer to packet's header (i.e. ***AodvT_Rreq*** or ***AodvT_Rrep***)
- ***AodvT_Route_Table*** – table of active routes to destinations
  - Implemented as hash table
  - Indexed by destination's IP address
  - Populated via RREQ and RREP messages
- ***AodvT_Route_Entry*** – a routing table entry
  - Contains destination address, next-hop address, sequence number, etc.

Key AODV data structures

- ***AodvT_Request_Table*** – maintains data about RREQ messages that travel through the node
  - RREQ created by the node (***AodvT_Orig_Request_Entry***)
    - Keeps track of active route discovery processes
    - Determines if route discovery succeeded
    - Implements expanding ring search
  - RREQ sent to the node
    - Avoids duplicate processing of arriving RREQs
- ***AodvT_Conn_Info*** – maintains information about neighboring nodes
  - Populated via HELLO messages

Modeling GeoAODV

- Add new protocol configuration parameters
  - Changes are implemented in **manet_mgr** process model
  - Specify GeoAODV model attributes
  - Parse attributes in ***attributes_parse_buffers_create*** function
- Modified AODV control packet headers
  - ***AodvT_Rreq***
  - ***AodvT_Rrep***
- Modified AODV control packet managing routines
  - ***aodv_rte_rreq_pkt_arrival_handle***
  - ***aodv_rte_rrep_pkt_arrival_handle***
- Modified AODV route request and connectivity tables
- Added ***geo-table*** to store GPS coordinates of the nodes

Modeling GeoAODV

- Geo-table is updated via RREQ and RREP packets
- Update route discovery process
  - Algorithm for determining if a node rebroadcasts RREQ is changed
    - Perform regular AODV verification procedures
    - Compute search region at each node
    - Check if the node belongs to search region
  - Implement algorithm for computing initial value of flooding angle
  - Implement algorithm for increasing flooding angle if previous round of route discovery failed
- GeoAODV implementation is placed into external files

GeoAODV Implementation

Start of GeoAODV route discovery process:

Retrieve current node's location coordinates

***If** the destination's coordinates are available in geo-table **then** {*

Retrieve destination coordinates from geo-table

Compute the initial value of the flooding angle

}

***Otherwise** (destination coordinates are unknown) {*

Set the flooding angle to 360 (e.g. regular AODV)

Set destination coordinates to (-1, -1) (e.g. unavailable)

}

Generate RREQ message using obtained flooding angle and source and destination coordinates

Send generated RREQ message

Update Request Table (i.e. record RREQ message information)

GeoAODV Implementation

Computing the initial value of flooding angle:

If *destination coordinates are unknown* ***then return*** *360*

Set the value of the flooding angle to 90

Repeat while *flooding angle < 360 {*

> ***If*** *there is at least one neighboring node within the search area defined by the current value of flooding angle*
>
> ***then return*** *the current value of flooding angle*
>
> ***Else*** *increment the value of flooding angle by 90*

}

Return *current value of flooding angle*

GeoAODV Implementation

Processing of RREQ at intermediate node:

Update geo-table with originator node coordinates and destination node coordinates ***if*** *fresher coordinates have been received*

If *the node is destination or the node knows route to destination* ***then*** *{*

Generate RREP

Return

}

If *AODV determines that RREQ message is to be rebroadcast and* ***if*** *the node is within the search area* ***then*** *{*

Rebroadcast RREQ

Update Request Table

Return

}

Drop RREQ message

GeoAODV Implementation

Overview of GeoAODV route discovery process:

```
Done = false
While (!Done) {
        Compute new flooding angle
        Update Request Table
        Send RREQ
        Start Route Request Timer
        Wait until RREP arrival or Route Request Timer expiry
        If RREP arrives then Done = true
        If Route Request Timer expires     then {
                If (flooding angle == 360)  then Done = true
                Else Done = false
        }
  }
Send accumulated application packets
```

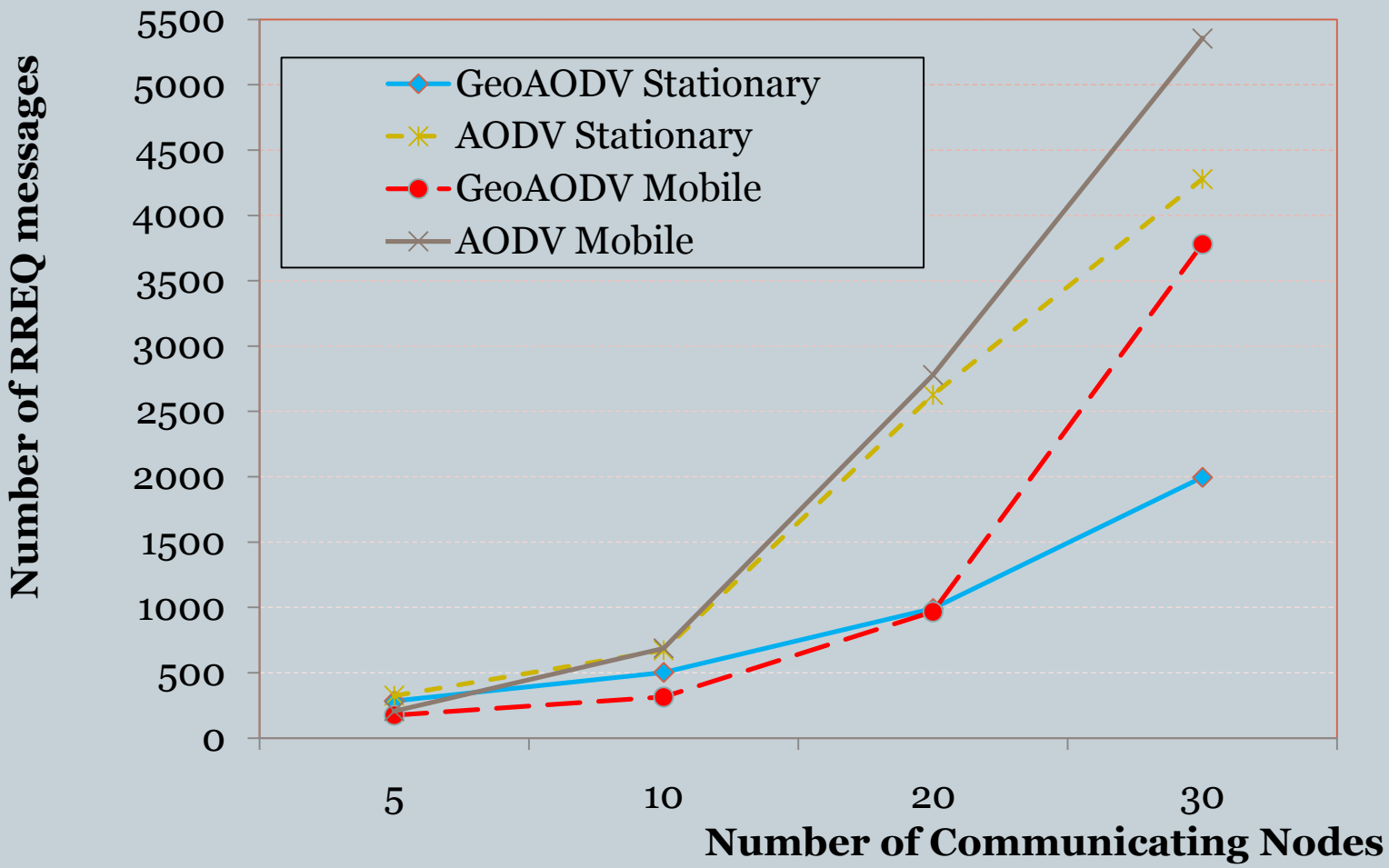
Simulation study

- Simulated area is 1000 meters x 1000 meters
- Randomly placed 50 MANET nodes
- Nodes move using random waypoint model
- Average node speed was uniformly distributed between 1 and 10 meters/second
- The number of communicating nodes range between 5 and 30

Simulation study

Configuration Parameter	Value
Channel Data Rate	11 Mbps
Transmit Power	0.005 Watts
Packet Reception Power Threshold	-95 dBm
Start of data transmission*	*normal* (100, 5) seconds
End of data transmission	End of simulation
Packet inter-arrival time*	*exponential* (1) second
Packet size*	*exponential* (1024) bytes
Duration of Simulation	300 seconds

Simulation study



Summary and Conclusion

- GeoAODV outperforms AODV
  - The whole network is not searched
  - Fewer RREQ and RREP packets travel through the network
- Improvement is lower in mobile scenarios
  - Node coordinates are not as accurate
  - More rounds of route discovery may be needed

Future Work

- Compare GeoAODV to LAR and other related protocols
- Modify the processing at the intermediate node so that it computes the search area formed in respect to previous node – destination node line, instead of sourse–destination line.
- Add provisions to ensure that the route discovery does not extend too far beyond the location of the destination node.

Future Work

- Dynamically increase flooding angle at the intermediate nodes if there are no immediate neighbors within the search area defined by the flooding angle of arriving RREQ message.
- Combine multiple ideas presented above to maximize the overall improvement.
  - One such hybrid approach could be to use only two values of flooding angle, 180 and 360, and always compute the search area based on the location of the previous node.