CSC 714: Real Time Computer Systems
RoverNet: Mobile Routers and Jammers in Wireless Sensor Networks
Progress Report

Kunal Kandekar	Nandini Kappiah	Indraneel Kelkar
(kakandek)	(nkappia)	(ibkelkar)

1. Introduction

Denial of Service (DoS), also known as jamming, is a major source of concern in any wireless network. Introduction of mobility in wireless nodes further complicates the scenario. This proposal aims to study the dynamics of such a system and develop techniques to take advantage of mobility in establishment and disruption of wireless communication.

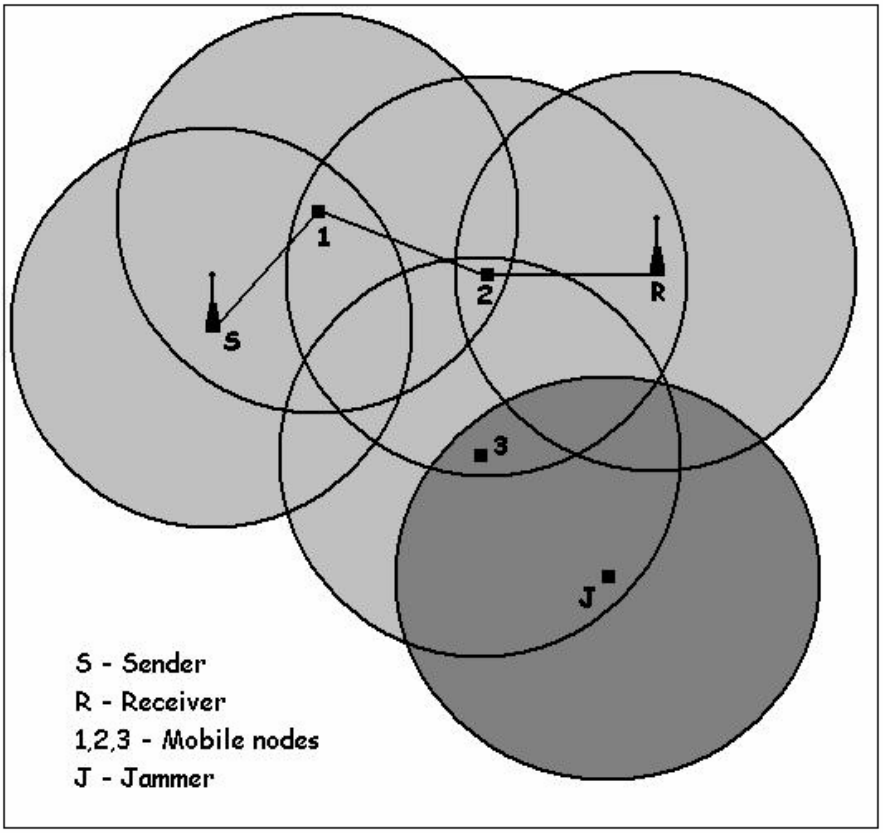


Figure 1: Overview of the RoverNet

One possible defense against DoS attacks is introducing mobile nodes to route around his attacks. We study and develop techniques to dynamically relocate these routers when faced with a DoS attack. The natural counter-offense against such a scheme is the use of mobile Jammers. Hence we study the dynamics of interaction between mobile Routers and Jammers, and attempt to develop effective techniques for communication and disruption using mobile wireless nodes.

2. Issues

We faced several issues in the experimental setup of this analysis, mostly regarding the mobility of the nodes in the wireless network.

2.1 Routing in Mobile Wireless Networks

- Establishing a route over mobile routers: Conventional mobile ad-hoc network (MANET) protocols address the mobility aspect of their nodes by maintaining their routing tables in a soft-state. Since the nodes in rover-net can cover rather large distances relative to their IR-range, a soft-state approach would be inefficient. Alternatively, a large overhead in terms of data and computation (keeping track of the positions of all the mobile nodes) would be needed in establishing a route.
- Maintaining the route: If a route is eventually established and the nodes keep moving, they would leave the range of their previous- or next-hop neighbors and disrupt the route. There will be a large overhead of control messages required to keep rebuilding a route if it is broken.
- Collision-avoidance: All nodes are communicating over a shared medium (Infra-Red) hence chances of a collision are high if there is no established scheme to regulate transmission of messages.

2.2 Detection of Jamming

Conventionally jamming can be detected at several levels:

- Physical: by analyzing the signal
  - Repeated inability to access wireless channel
  - Repeated collisions
  - Low signal-to-noise ratio
  - Excessive received signal level
- Data Link:
  - Bad framing
- Network and higher:
  - Checksum failures
  - Illegal values for addresses or other fields
  - Protocol violations (e.g., missing ACKs)

We cannot detect jamming at the Physical and Data Link layer due to lack of direct access to the IR communication hardware and information about the actual signal. The only indication we do have of a problem at the lower layers is the detection of a collision as provided by the LNP API. Hence there is no clear method to distinguish between genuine medium-access collisions and jamming.

2.3 Re-organizing the rover-network on jamming to resume communication

On detection of jamming, the mobile nodes are to re-organize themselves so as to enable routing around the jammed area. Since there is neither absolute nor relative location feedback, it is difficult to re-organize the mobile nodes with minimum co-ordination and movement.

To address all these issues, we developed the RoverNet Protocol (RNP), which is discussed in detail below.

3. Rover-Net Protocol (RNP) Specification

The broad goals of this transport-level protocol are:

- Establishing a route over mobile nodes
- Maintaining the established route
- Collision avoidance scheme (as we do not have control over the data-link layer)
- Ability to function with a minimum number of mobile routers

- Detection of jamming.
- Avoidance and re-organization when faced with jamming.

3.1 Routing in Mobile Wireless Networks

We use a scheme similar to the Dynamic Source Routing [1]. However we include modifications to address the issue of limited communication range compounded with the high mobility of the nodes.

3.1.1 Establishing a route over mobile routers:

- Rovers move around broadcasting their presence. They maintain a table of nodes within their range. Entries in this table are removed if there is no heartbeat for some period of time from the corresponding node.
- The source initiates a *route-establishment request* for the specified destination. Rovers that receive it append their Id to the message and forward it to any other nodes within their range. On forwarding the message, the rover stops moving for a certain period of time or until it receives a route-established message.
- Whenever a node receives this message and has the destination node in its table, it broadcasts a route-established message, which is received by both, the destination node as well as the previous-hop node.
- The *route-established response* will also have the list of Node Ids, so that this message can be forwarded backwards to all nodes along the route. All nodes receiving this message will stop moving and transition into "routing" mode.

3.1.2 Maintaining the route:

- There is high probability that the established route will be disrupted if the nodes change location. To minimize this threat, the protocol requires the nodes to stop moving unless it detects jamming.

3.1.3 Collision-avoidance scheme – As we do not have access to the IR communication hardware or the data-link layer that is used by the rovers, we need to implement a collision-avoidance scheme at the transport layer. The LNP API notifies the application whether its transmission failed due to a collision. In case of a collision, we use a back-off timer scheme to schedule the retransmission with a minimum probability of collision.

The back-off time is a function of the node Id, which is unique in the rover-net. This should increase the probability of a collision-free retransmission regardless of the participants in the collision.

3.2 Detection of Jamming

Due to lack direct access to IR Communication hardware and information of the actual signal, it is difficult for the rover to identify whether it is being jammed.

From the LegOS LNP API by itself, it can only know whether a transmission resulted in:

- success
- a collision
- network error (miscellaneous)

Moreover, a rover is unable to detect reception of unsuccessful transmissions. This in itself is insufficient data for the rover to determine with certainty if it is being jammed at the data-link level. Hence we use a number of probabilistic and heuristic approaches to detect jamming.

As mentioned earlier a RoverNet node can be in 3 modes (Stationary Source/Sink, Mobile Router, and Standby). Jamming is detected in the same way by any type of node: It assumes that it is being jammed if:

- It receives several bad packets (Jam messages)
- It detects too many successive collisions while sending.

Any node will also assume that its next-hop neighbor is being jammed if:

- It does not receive acknowledgements for its transmitted packets.
- It does not receive a heartbeat for a specified amount of time.
- It receives a heartbeat with the state field set to JAMMED.

However each mode has different techniques of responding to jamming.

1. Stationary Source/Sink:
   - On detecting local jamming: If it is a standalone tower, it stops all transmission until the Jammer node moves away or stops jamming. In case there is an alternate Source/Sink that it can route to over an out-of-band connection (e.g. over the LAN to another PC with an IR-tower), it transfers control to that node.
   - On detecting next-hop jamming: It initiates a new Route Establishment request, so that data can be routed around the jammed area.

2. Mobile Router:
   - On detecting local jamming: It attempts to send heartbeat message with state set to JAMMED to notify nodes within its range. If the jamming is too persistent to successfully transmit this message, it assumes that the next-hop nodes will detect this through the other above-mentioned approaches. It then moves away in an attempt to discover a jamming-free area that still allows a route to be established.
   - On detecting next-hop jamming: It forwards the status of the next-hop neighbor to all nodes within its range.

3. Stand-by Router:
   - On detecting local jamming: Since it is not on the established route, it has 2 choices: it can move away to an un-jammed area, or it can choose to stick around and deceive the jammer (decoy).
   - On detecting next-hop jamming: It attempts to build a new route around the jammed node.

3.1 State Transition Diagrams

The protocol basically differentiates between two types of nodes:

- Stationary (IR Tower)
- Mobile (RCX Rover)

Each type of node has a different state-transition diagram.

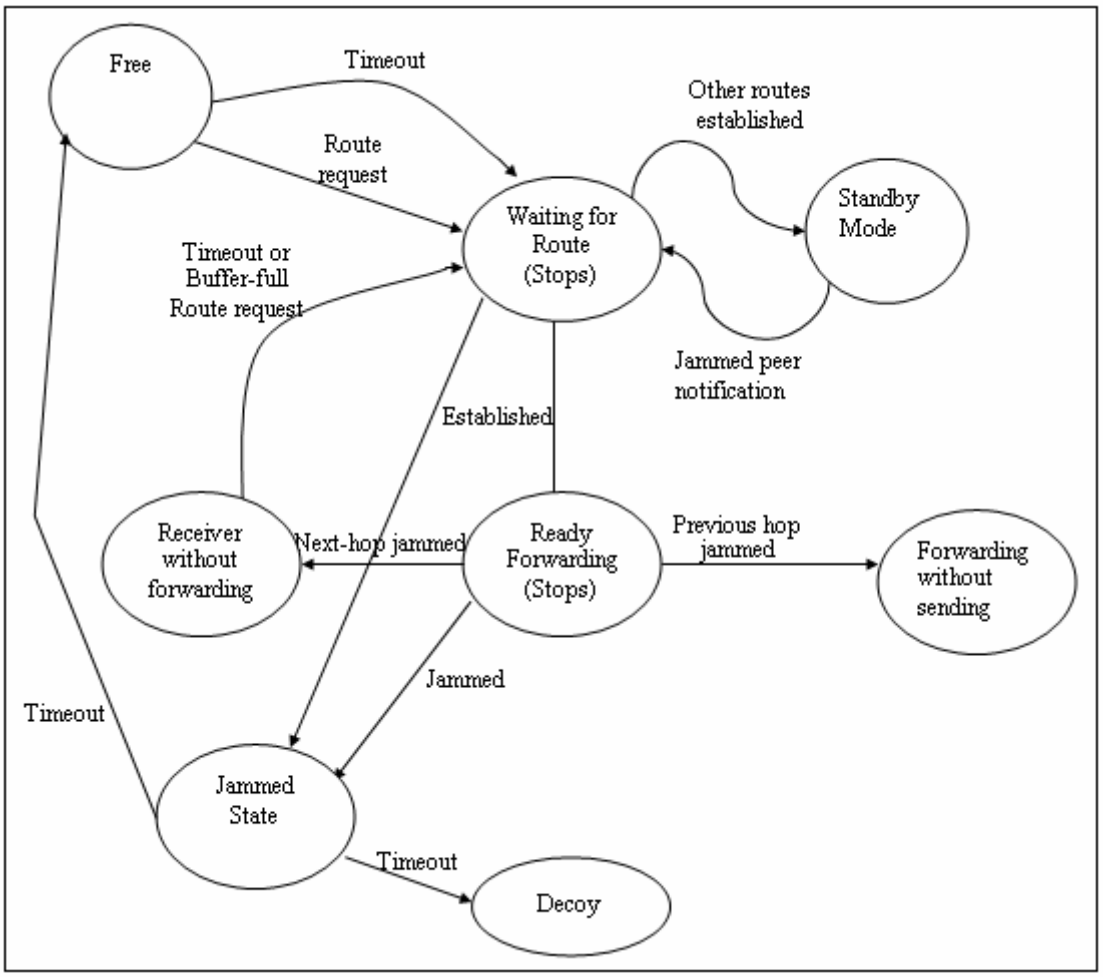


Figure 2: State Transition Diagram of the Rover

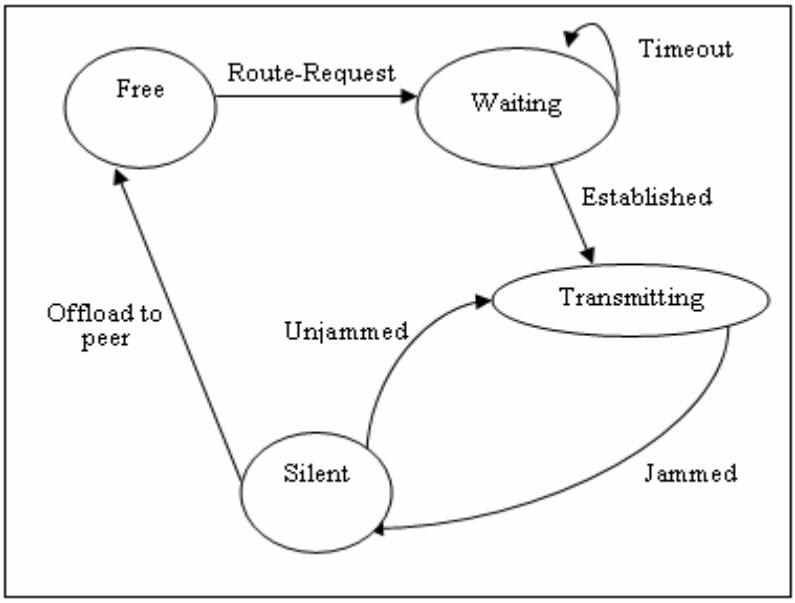


Figure 3: State Transition Diagram of the Tower

3.2 RNP Protocol Headers

Message Types	Description	Fields
RNP_ROUTE	Request/Response for establishing of route	Source, Destination, List of Node-Ids, Type.
RNP_ROUTE_SETUP	Setup Request	-
RNP_ROUTE_REQ_APPOINT	Request appointment	-
RNP_ROUTE_CNCL_APPOINT	Cancel appointment	-
RNP_ROUTE_APPOINT	Grant appointment	-
RNP_ROUTE_REQ_ACK	Acknowledge route establishment	-
RNP_ROUTE_TEARDOWN	Request teardown	-
RNP_UCAST	Unicast data message (intended for a specified destination within the broadcast range)	Source, Destination, Sequence, Length
RNP_MCAST	Multicast data message (intended for all nodes within the broadcast range)	Source, Length
RNP_ACK	Acknowledgement for Unicast message	Source, Destination, Sequence
RNP_NACK	A Negative-Acknowledgement on detecting a gap in sequence numbers.	Source, Destination, Sequence
RNP_HBEAT	Heartbeat message to notify all nodes within range of status of the sending node.	Source, Node-type, Status, Sequence, Statistics
RNP_PING	An “Are you alive?” query	Source, Destination, Hops
RNP_PONG	An “I am alive!” response	Source, Destination, Hops
RNP_QUENCH	Control message to indicate that the buffer is full on sending node. Previous hop node should desist sending more packets until RNP_RESUME.	Source, Destination
RNP_RESUME	Control message to indicate that node is ready to forward again.	Source, Destination

4. Jammer

It randomly moves around listening for messages. If it receives a message, it knows that it is in the vicinity of a victim. Based on the frequency of received messages, it adjusts the rate of its own jamming transmissions (saving power). It follows a very simple algorithm:

1. It starts in "seek" mode by randomly moving around listening for messages.
2. When a message is received, the jammer knows it is in the vicinity of an active node. It stops moving and starts jamming.
3. Periodically it stops jamming and listens for a while to ascertain the node's presence.
4. If there is no message from the victim node, it goes back into seek mode.

5. Implementation

We implemented a sample mobile embedded wireless node network as described above. The mobile nodes are emulated by Lego RCXs units, and the LNP IR acts as the broadcast medium. This project aims to study the dynamics of such a system and develop techniques to take advantage of mobility in establishment and disruption of wireless communication.

5.1. Design

The implementation of this project was divided into the following modules:

5.1.1. RNP Library

We wrote a library for RCX communication using LNP. The Rnet APIs which we implemented have been modeled after the socket library, providing an interface similar to that of sockets to an end user.

RNP Programming Interface

The RNP API is declared in the "rnet.h" file

int rnp_init (void) - Sets up the environment for RCX communications. It starts up the RNP daemon which handles are further communication. Its also call lnp_init which sets up

int rnp_timed_listen (long timeout) - Listens for messages on the broadcast medium for a specified time depending on the argument specified. A listened timeout has been implemented due to the unreliable communication of the RCXs. If no messages are received in the timeout specified, the rnp_timed_listen returns -1. If a message is received, it return 0.

int rnp_connect (usigned char dest) - rnp_connect initiates the dynamic source routing from the source to the specified destinations. It returns the connection identifier if the connection is succesfully set up or it returns -1 if the dynamic source routing fails.

int rnp_disconnect(int conn) - Tears down the connection established. Returns 0 if teardown occurs successfully. Returns -1 if teardown unsuccessful for unspecified reasons.

int rnp_send(int conn, char *data, int len) - Transmits the contents of data of length len on the specified connection, a unicast send in which the data in sent to the specified destination.

int rnp_recv(int conn, char *data, int len) - rnp_recv is a blocking call which waits for on the specified connection. Populates the values of data and len for use by the calling function

int rnp_timed_recv(int conn, char *data, int len, long timeout) - rnp_timed_recv is a blocking call which waits for the specified timeout period on the specified connection. Populates the values of data and len for use by the calling function

int rnp_mcast_send(char *data, int len) - rnp_mcast _sends provides a multicast send in which the message is broadcast to all listening nodes.

int rnp_mcast_recv(char *data, int len, long timeout)
rnp_mcast_recv provides a non-connection oriented multicast receive

int rnp_status(void)
rnp_status returns the status of the node. The valid states of the node are
RNP_FREE – indicates that the node is free and moving
RNP_CONNECTING – indicates that the node is the process of a route establishment and the rover has stopped moving
RNP_CONNECTED – indicates that the node has successfully established connection and the rover is stationary
RNP_INCOMING – indicates that the rover is expecting messages, the node is stationary is this case as well
RNP_JAMMED – indicates that the rover is jammed and is capable of moving around
RNP_STANDBY – indicates that the node is the standby node and is moving around

5.1.2. RNP Daemon

We designed and implemented an RNP daemon which is responsible for communication between the RCX's and the tower and between the RCX's themselves. The RNP daemon runs in the background and handles all communication requests from the tower or the RCX. The tower and the RCX send all messages to the RNP daemon which is responsible for transmitting them over the wireless broadcast medium. The RNP daemon emulates the CSMA-CD (Carrier Sense Multiple Access- Collision Detection) and handles collisions in the medium with a back off (proportional to its own ID) and repeated transmissions.

This daemon allows for asynchronous communication as the tower hands off the communication requests to the daemon and daemon, running on a separate thread, is responsible for the communication. The actual communication is abstracted for the tower and the rover. When a rover gets a packet which is not addressed to it, the daemon handles the packet and the rover is unaware of the underlying communication. This allows the rover to continue undisrupted until it receives a packet that is intended for it.

5.1.3. RoverNet Simulator

The lack of debugging facilities on the RCX prompted us to create a simulator which simulates the networking aspect of the RCX. We created a Java based simulator on multicast environment which simulates the communication between RCX's. IP-multicast emulates the IR broadcast medium on which the RCX works. We also created a Java multicast network monitor which tracks the messages exchanged between the mobile nodes and provides a useful tool for debugging the complex code involved in this project. The monitor also provides a graphical interface which depicts the location of the rovers and their respective broadcast ranges. The display provides an intuitive understanding of the network of mobile nodes and the communication that is taking place.

Our current implementation as of now includes the following features:

- Rnetd Daemon implementation for asynchronous messaging.
- Simulation of mobile wireless node network (RCX-portable code)
    - Emulation of IR-broadcast using IP multicast
    - Graphical Representation of simulation
- Dynamic Source Routing.
- Multi-hop forwarding.

Future work includes:

- Implementing Reliability service (using ACKs, NACKs)
- Implementing Flow Control (using QUENCH, RESUME)
- Jamming strategies
- Anti-jamming detection and response strategies

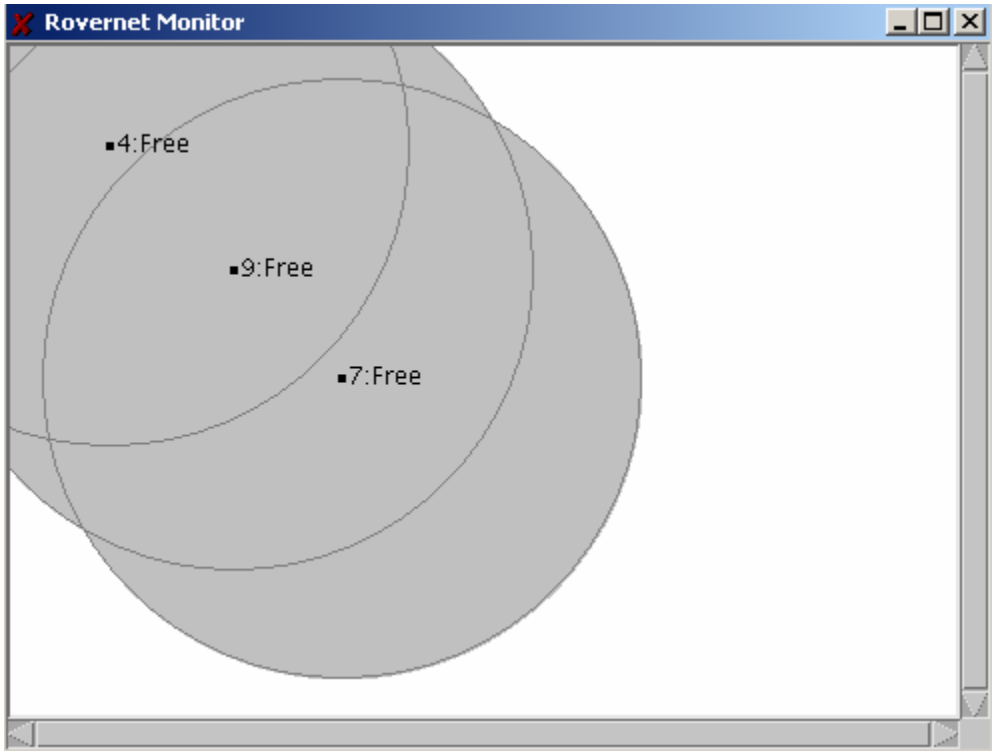
Open Problems and Potential Solutions

- Use of Appointments: A slight deviation we had to make to pure dynamic source routing is the use of appointments. When a number of nodes receive the route establishment request, they return an acknowledgment of the route establishment request to the source node, upon which the source appoints one of the nodes as the next hop. The other nodes are informed of this appointment and they are free to rove further in search of the destination. This improvisation has been made to prevent redundant or partial routes from being established.
- Reliable communication: We have not implemented any form of reliable communication. The packets can be encrypted for protection. The current identification of a jammer packet is done by examining the header. If it does not follow the protocol format it is considered a jammer packet. More extensive methodologies, than the one implemented, can be used to identify and avoid jamming.
- Congestion control: We have not implemented any form of congestion control in our implementation although our protocol provides for it. RNP_QUENCH message which indicate that the buffer on the sending node is full and the previous node should desist from sending more packets. Alternate routes can be established around the congested node to continue transmissions.
- Strategies for jamming and anti jamming: We have only considered few of the methods of the many strategies of jamming and anti jamming. The jammers and the nodes can be respectively provided with different jamming and anti-jamming which results in different behavior of the mobile nodes than the one we simulated.

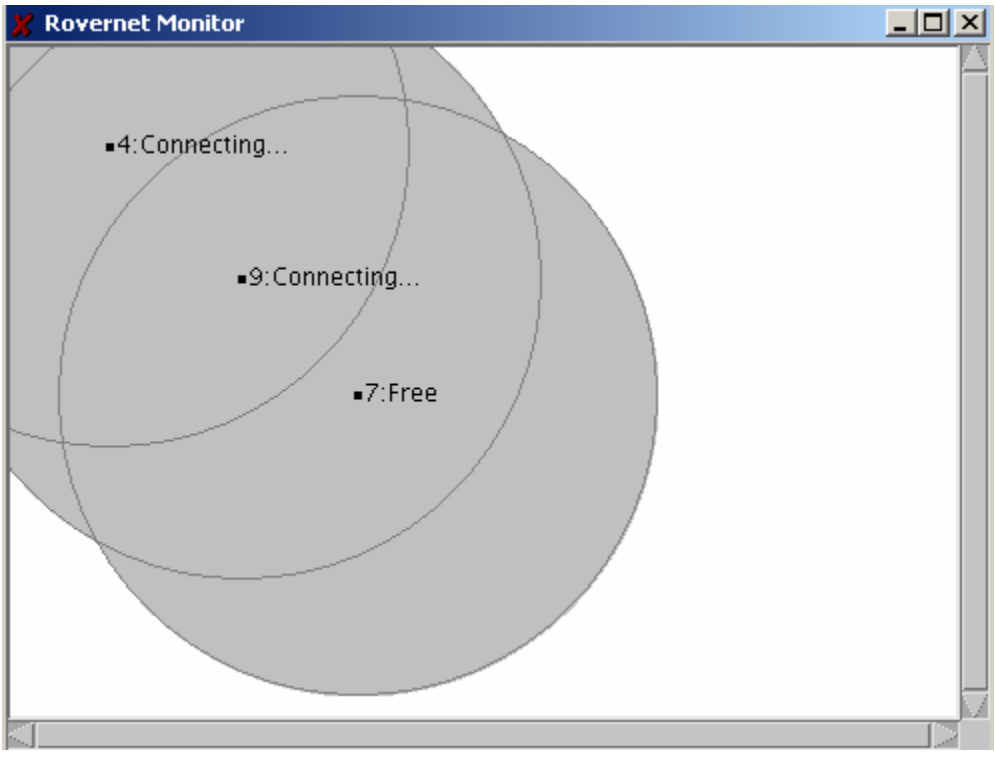
5.1.4. Individual contributions:

Kunal: The RNP daemon (rnetd), Java RoverNet simulation monitor (JRoverNet)
Nandini: The Rover code, Tower code and the Jammer code
Indraneel: Miscellaneous data structures implementation (Circular buffers, Linked list)

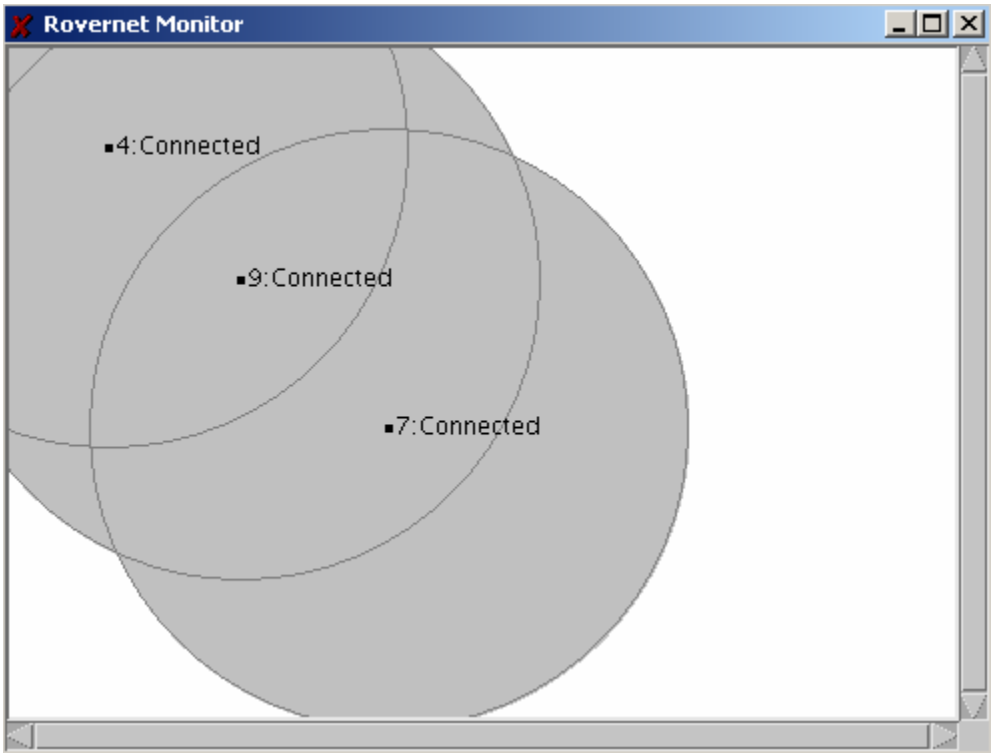
6.1.6. Simulation Results



(a) Initially: All nodes free



(b) Connecting



(c) Connection established
Fig 4: Simulation

6. Open Issues

It is as yet uncertain whether location feedback can lead to optimal performance of this protocol. For instance, using continuous location feedback, nodes may be able to infer the ranges of its peer nodes and plan its movements more intelligently.

7. Future Work

Integrate simulation with LegoSIM to provide the simulator with networking capabilities which it currently lacks.

On top of this framework, it is possible to develop several scenarios and corresponding strategies for both Routers and Jammers.

- **Scenario:** Single Sender and single Receiver
  Strategy: Jammer tries to locate the source and effectively jam all communications.

- **Scenario:** Multiple Senders and Receiver
  Strategy: Senders (and Receivers) can communicate amongst themselves over an out-of-band connection (the LAN) hence the Jammer is forced to target the Routers. Hence the Routers must evade the Jammers without being able to communicate to coordinate. Biologically-inspired algorithms such as swarm behaviour can be applied to develop these strategies.

- **Scenario:** Switching between short and long range transmissions for Routers.
  Strategy: The mobile Routers exploit different communication ranges to their advantage. It is more difficult for the Routers to establish a link using short-range transmissions due to the directional considerations. However it is also difficult for the Jammers to detect these links

and jam them effectively using long-range transmissions without losing an undue amount of power.

- **Scenario:** Robotic Virus
  Strategy: Jammer acts a self-replicating virus. It locates a Router and tries to reprogram ("infect") it to behave as a Jammer. This exploits a feature.

Conclusion

This work has several ramifications in real-world scenarios involving mobile sensor networks in military applications. We have provided a framework for practical implementation as well as simulation to determine strategies that would be most effective with respect to both communication as well as jamming.

References

[1] Dynamic Source Routing Protocol (http://www.ietf.org/internet-drafts/draft-ietf-manet-dsr-09.txt)

[2] A.D. Wood and J.A. Stankovic, "Denial of Service in Sensor Networks", IEEE Computer Society Press, 2002

[3] A. D. Wood, J. A. Stankovic, "JAM: A Jammed-Area Mapping Service for Sensor Networks", Sang Hyuk Son, RTSS'03

[4] Eric Bonabeau, Guy Theraulaz, Swarm Smarts – Scientific American, 1998

[5] J. Butler, "Mobile robots as gateways into wireless sensor networks", (http://www.linuxdevices.com/articles/AT2705574735.html)

[6] Andrew Howard, Maja J Mataric, Cover Me! A Self-Deployment Algorithm for Mobile Sensor Networks

Project Web Page URL: http://www4.ncsu.edu/~nkappia/rtcsproject.html