# Crowd Management using Ad-hoc Networks

## Project Demonstration

Abhinav Golas Mohit Rajani

Pulkit Gambhir Nilay Vaish

Department of Computer Science and Engineering, IIT Delhi

Project Website

November 26, 2005

# What We Wanted to Deliver

- Crowd and environment monitoring in large gatherings.
- System to achieve the same must be scaleable, portable and require minimal configuration.
- Building such a system on a robust ad-hoc network backbone.

# Real Time Applications

- Indoor/Outdoor Environmental Monitoring
- Security and Tracking
- Health and Wellness Monitoring
- Factory and Process Automation
- Seismic and Structural Monitoring

# Why We Chose Ad-hoc Networks

- Easy deployability.
- Scalability much beyond any other wired or wireless network solution.
- Zero infrastructural requirement.
- Self configurable and Self healing networks.
- Global networking parameters determined by purely local decision making.
- Minimal external computation required.

Home Page
Title Page
Page 4 of 24
Go Back
Full Screen
Close
Quit

# System Specifications

- (Network Mote + Sensor Board) deployed at every entrance and exit.
- Each network mote logs data about environment.
- Data routed in a efficient store and forward manner to a base station.
- Base station may deploy data over an alternative network interface such as LAN or GPRS.

# Hardware Used

## Berkeley MPR410 wireless motes containing

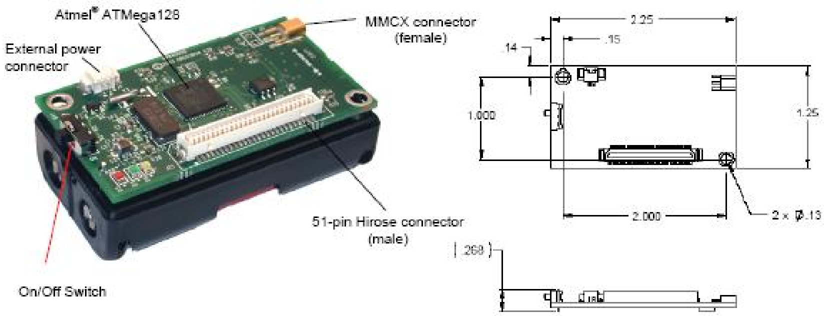



Figure 1: MICA Mote : MPR410

- ATMega128L $\mu$-controller
- CC1000 radio transceiver
- 512Kb Flash logger
- Pluggable MTS400CA sensor boards to detect personnel movement and environment monitoring

# MIB510 Programming Board Having

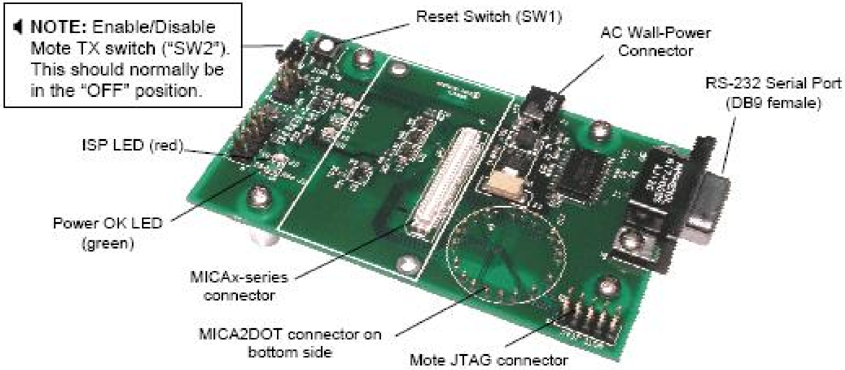



Figure 2: Programming Board : MIB510

- On-board in-system processor (ISP)-an Atmega16L
- ISP runs at a fixed baud rate of 115.2 kbaud
- Serial interface to PC
- Easy on-board testing of motes

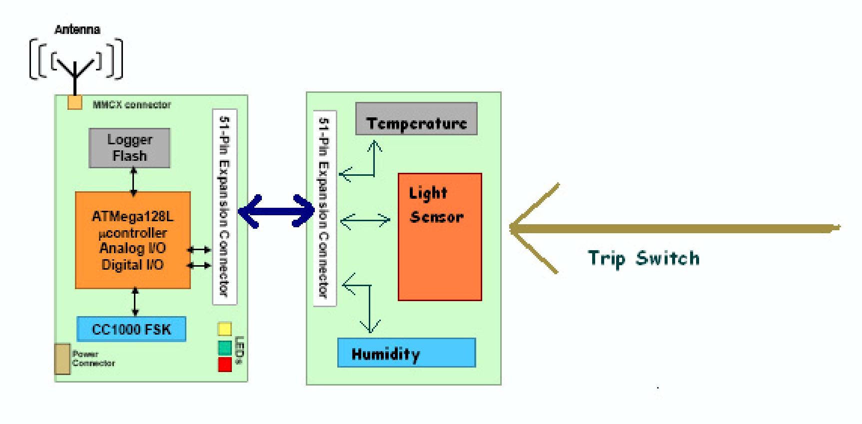



Figure 3: Block Diagram : Motes and Sensors

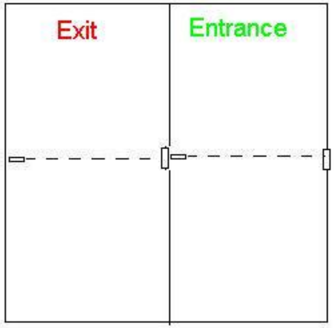
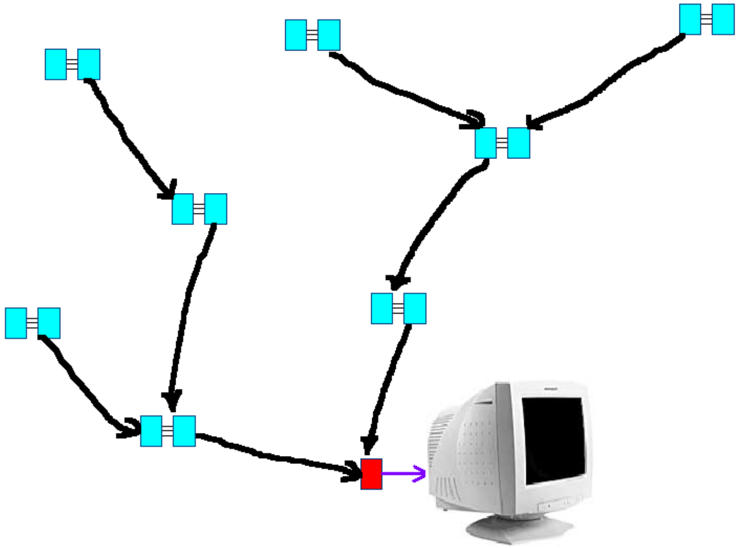



Figure 4: Door Setup

# Algorithms Implemented

Three algorithms for routing and scheduling were developed and coded independently. Of these, we faced problems in one, the other two have been implemented.

1. Message Store and Forward Based Scheduling (*Not Implemented*)
2. Post Order Traversal Based Scheduling (*Implemented*)
3. Time Synchronisation Based Scheduling (*Implemented*)



# Network Overview : Inverted Tree

# Store and Forward

- Stage 1 : Figure out routes via packet flooding.
- Stage 2 : Extend one-way routes to two-way links by parent informing messages.
- Stage 3 : Query subtree sizes of each node within the network.
- Stage 4 : Based on subtree sizes dynamically decide length of network query windows.
- Stage 5 : Recursively traverse network tree, making multiple queries at any time within the tree.
- Stage 6 : No message transferred directly to base, all messages of every sub-tree collated before transmission.

# Post Order Based Scheduling

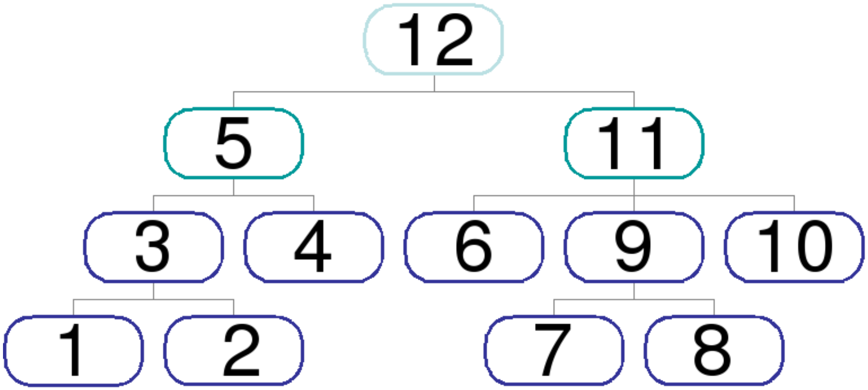



Figure 5: Post Order Tree

- Visit parent after visiting children
- Each node assigned a number according to its position in Post Order traversal
- Nodes transmit data in order of their post order nos.

# Why It Works

- Simultaneous transmissions take place all over the tree. What about collisions ?
- Node m+1 is atmost a level higher than node m.
- Message of node m is always above that of node m+1 at any time t.

Messages cannot collide !

# Advantages

- Time Period - proportional to number of nodes
- Collision Avoidance guaranteed by theoretical model
- Message Loss is tolerated by system.

# Time Synchronisation Based Scheduling

- Initially a route finding phase is implemented on a freshly booted network.
- During the route finding it is ensured that all motes synchronise their clocks with the base station.
- Once all the motes have synchronised, the scheduling scheme can be implemented.
- Entire time span is divided into *Global Data Collection Windows*.
- Each of these global window is further sub-divided into *Local Data Collection Windows*.
- Time synchronisation keeps motes aware of the *Global Data Collection Windows*, within which they choose their unique *Local Data Collection Window* using their unique network ID.

# Implementation Details

- The networking protocol was initially implemented and throughly tested on the network simulator *TOSSIM*.
- Due to inherent bugs in *TOSSIM*, substantial time was spent in porting the code onto hardware.
- Functionality of code has since been extended to query various environmental sensors and route their data through the network as well.
- Current Data Packet Size : $5 + 9$ bytes. Information relayed includes sensor values.

# Protocol advantages

- Minimizes number of message transmissions.
- Very little communication overhead.
- Nearly no computation required on motes.

# Problems Faced

## Deciding on the algorithm

- Energy efficient
- Minimal packet loss and collision avoidance
- Time Synchronization
- Routing and Scheduling

## Implementation Issues

- Understanding NesC and its constructs
- Functioning of Mica Motes and the Programming Board

## Algorithmic Hitches

- Whether message sending be acknowledge based
- Who stole my messages
- Debugging programmes

## Testing Blues

- Lab environment
- Open environment testing
- Mobile platform

## Personal Side

- Procrastination
- Internal tiffs

# What We Achieved

- Both the algorithms have been fully tested on networks of varying topologies having upto 6 motes with sensor boards attached to them.
- Algorithms have been found to be functioning successfully with almost no message loss.
- Power simulation for the first algorithm has been done.
- A C++ based interactive GUI using FLTK has been built.
- GUI has been successfully interfaced with the network via Serial port communication.
- Trip switch component has been tested and needs to be plugged into rest of the system.

# Interactive GUI

- Data read via serial port onto base server
- This data can be accessed anywhere via the net using our interactive GUI
- Configuration setup initialisation done first
- Network port setup
- GUI displays updated information as they become available
- Also displays network topology
- Separate information can also be seen from each mote

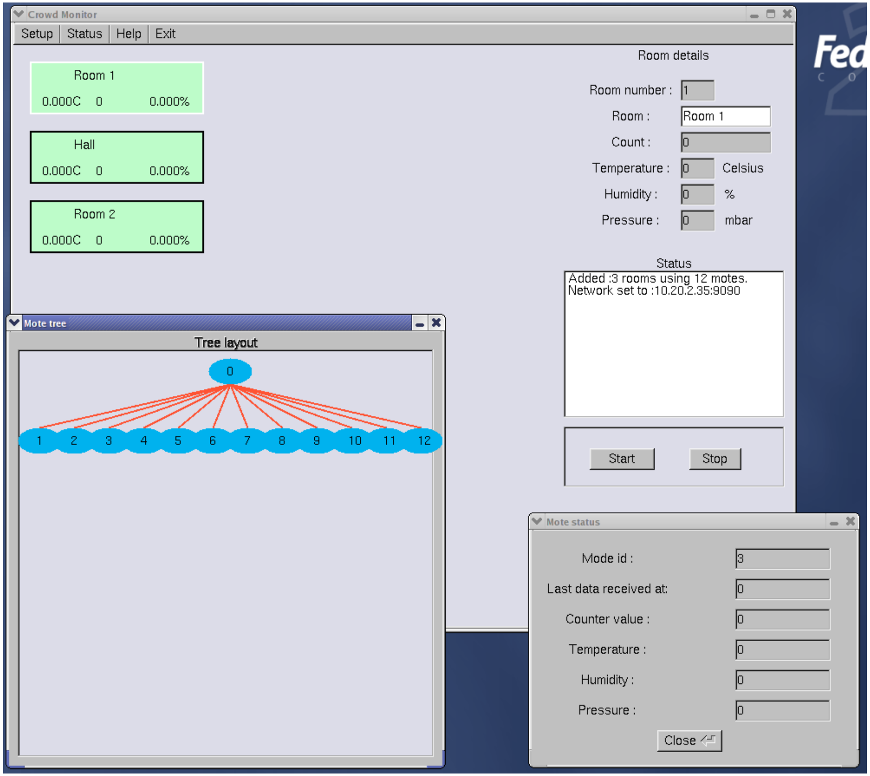



Figure 6: The Interactive GUI

Home Page
Title Page
Page 20 of 24
Go Back
Full Screen
Close
Quit

# Work Distribution

- **Basic Network Building Stages** : Nilay, Pulkit
- **Store & Forward Protocol** : Nilay, Pulkit
- **Time Synchronisation Protocol** : Pulkit
- **Post Order Traversal Protocol** : Abhinav, Mohit, Nilay
- **Door Switch** : Abhinav, Mohit
- **GUI & TCP/IP interfaces** : Abhinav

# Demonstration Plan

- Placing motes over a sizeable geographical area
- Bringing the network to life from a mobile platform
- Time synchronization
- Dynamic tree building
- Information routing after fixed intervals of time
- Interactive GUI display

# Scope for Future Work

- Refreshing network after fixed period of time
- Over the network programming
- Auto-detection of new motes and incorporating them into the network
- Detecting faults and reporting