Z25 Adaptive and Mobile Systems
Dr. Cecilia Mascolo

Programming Sensor Networks using Abstract Regions

Matt Welsh and Geoff Mainland
Harvard University

Programming Sensor Networks

- Development of sensor network application is difficult
  - Highly distributed systems
  - Constrained devices, energy, unreliable channels
- Designers make many complex low level decisions

Aims

- Provide general purpose primitives for sensor networks
  - Addressing, data sharing and reduction in local regions
- Raise the abstraction level exposed to the designer so to allow decoupling of application level concepts from communication and energy considerations

Application development

- Local coordination is an important aspect of many applications
  - Coordination at 1hop neighbours (spatial/local coordination)
  - Local coordination primitives are built by developers (at the moment)

Abstract Regions

- Spatial operators to capture local communication
- Node addressing through tuple space programming model
- Exposure of tradeoffs between communication and resource usage

- General enough to support different sensor applications

Notion of locality

- An abstract region defines a neighbourhood relationship among nodes
  - Set of nodes within N hops
  - Set of nodes within distance D
- Each node defines multiple abs regions
- Each node can be member of multiple regions at once

Programming operator

- Neighbour discovery
- Enumeration
- Data sharing
- Reduction

- Unified interface regardless of definition of membership
  – Readily change the underlying region implementation without affecting the application logic
  – E.G., Application using N radio-hop region can easily use geo neighbour region

Discovery Operator

- Discovery: discovering neigbours depends on the type of region
  – Broadcast messages, collecting node location, estimate radio links
- Continuous process
- Each node informed of changes (joining, leaving, moving nodes)
- Terminated: quality metric returned (accuracy of region info: % of nodes which responded)

Enumeration Operator

- Enumeration operator returns the set of nodes participating in a region, allowing them to be addressed
- Supplementary info can be provided (location)

Data Sharing Operator

- Allows variable sharing in the region
  - Get/put
    - Get(v,n): get value of v from node n
    - Put(v,l) stores value l in v
- The operator implementation is not specified
  - Unicast can be used to retrieve a value from a node
  - Put could store value locally
  - Alternative implementation could broadcast values (put) and get info locally from a cache

Reduction Operator

- Takes a shared variable and reduces the value of the variable across a number of nodes storing the value in a shared variable
- Reductions: sum, max, min,…
- Implementation could be done by
  - Collecting values locally
  - Forming spanning tree reducing at each level
- The shared variable interface hides the algorithmic details

Regions implementation

- N-radio hops: nodes within N hops
- With geo filter: nodes within N hops and distance d
- K-nearest neighbour: K nearest nodes in N hops
- K-best neighbour: K nodes with best link quality in N hops
- Approx planar mesh: mesh with small number of crossing edges
- Spanning tree: rooted at a node (used to aggregate)

Radio and Geo neighbourhood

- Node broadcasts geo adverts
- Data sharing
  - Push (put) to other nodes
  - Pull (get) messages from nodes
  - Reduction: collects locally and stores in other var

Approx Planar Mesh

- Each node discovers k-nearest radio neighbours
- Divide in m equal size sectors of 2pi/m
- Select nearest node in each sector
- 1hop broadcast of these outedges
- Each node receiving it checks if these cross their edges: in which case send a message invalidating the outedge
- Various rounds of broadcasts

Spanning Tree

- Nodes broadcast messages with their id and number of hops from root
- When one of these is received a node increments the hop count and reforwards with its id
- A node selects a parent with minimum hopcount
- However it also estimates link quality: if this falls below a threshold it selects another parent
- Other spanning tree implementations are possible

Spanning Tree Operations

- Useful at aggregating data at a single point
- A put from the root propagates the value to the whole tree
- A put from a non root node propagates the value to the root
- Reductions propagate data up the tree causing each node to aggregate local value with the data of its children

Quality feedback and tuning interface

- Quality of information implies energy consumption
  - More messages-> more energy used
- Abstract regions expose this trade off
  - Tune the number/frequency of messages used or rate of broadcast of location advert
  - Feedback to applications: quality measure (fraction of nodes which responded, timeouts of operation)
  - Tuning can be applied by applications to specify low level parameters

Implementation

- TinyOS: sensor operating system
- Added support for blocking/synchronous operations
  - This decreases the complexity of the code

API

```
/* Discover region */
result_t Region.formRegion(<region specific args>,
  int timeout);

/* Wait for region discovery */
result_t Region.sync(int timeout);

/* Set local shared variable */
result_t SharedVar.put(sv_key_t key, sv_value_t val);

/* Get shared variable from give node */
result_t SharedVar.get(sv_key_t key, addr_t node,
  sv_value_t *val, int timeout);

/* Wait for shared variable gets */
result_t SharedVar.sync(int timeout);

/* Reduce 'value' to 'result' with given op */
/* 'yield' returns pct of nodes responding */
result_t Reduce.reduceToOne(op_t operator,
  sv_key_t value, sv_key_t result,
  float *yield, int timeout);

/* Reduce and set result in all nodes */
result_t Reduce.reduceToAll(op_t operator,
  sv_key_t value, sv_key_t result,
  float *yield, int timeout);

/* Wait for reductions to complete */
result_t Reduce.sync(int timeout);
```

Applications

- Object tracking
- Contour finding
- Direct diffusion
- GPSR

Object Tracking

- Each node takes magnetic field readings & compares with a threshold
- If above communicate with neighbours and elect a leader (the max reading)
- The leader computes average and sends to basestation

Object Tracking with Abstract Regions

```
location = get_location();
/* Get 8 nearest neighbors */
region = k_nearest_region.create(8);

while (true) {
  reading = get_sensor_reading();

  /* Store local data as shared variables */
  region.putvar(reading_key, reading);
  region.putvar(reg_x_key, reading * location.x);
  region.putvar(reg_y_key, reading * location.y);

  if (reading > threshold) {
    /* ID of the node with the max value */
    max_id = region.reduce(OP_MAXID, reading_key);

    /* If I am the leader node ... */
    if (max_id == my_id) {
      /* Perform reductions and compute centroid */
      sum = region.reduce(OP_SUM, reading_key);
      sum_x = region.reduce(OP_SUM, reg_x_key);
      sum_y = region.reduce(OP_SUM, reg_y_key);
      centroid.x = sum_x / sum;
      centroid.y = sum_y / sum;
      send_to_basestation(centroid);
    }
  }
  sleep(periodic_delay);
}
```

Evaluation: adaptive reduction

- Reduction quality depends on replies from neighbours
- Hence it depends on retransmissions of reduction requests/replies
- Adaptive reduction allows to maintain a certain quality by retransmitting more when link quality seems bad

Adaptive Reduction

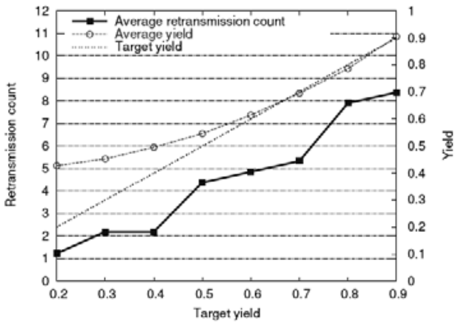

Figure 8: **Adaptive reduction algorithm performance.** *This figure shows the effectiveness of dynamically tuning the maximum retransmission count to meet a reduction yield target. The figure shows the average yield across 100 reduction operations, as well as the average retransmission count determined by the controller.*

Object Tracking Evaluation

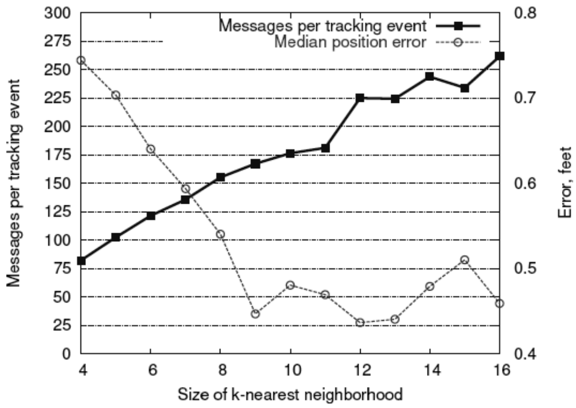

Figure 12: **Accuracy and overhead of object tracking as a function of neighborhood size.** *Results are the average of three runs for each neighborhood size.*

Comments

- First to provide an abstraction
  - Many other works improved this in terms of what can be provided to the developer and how
- Evaluation of this is difficult as you have to measure usefulness