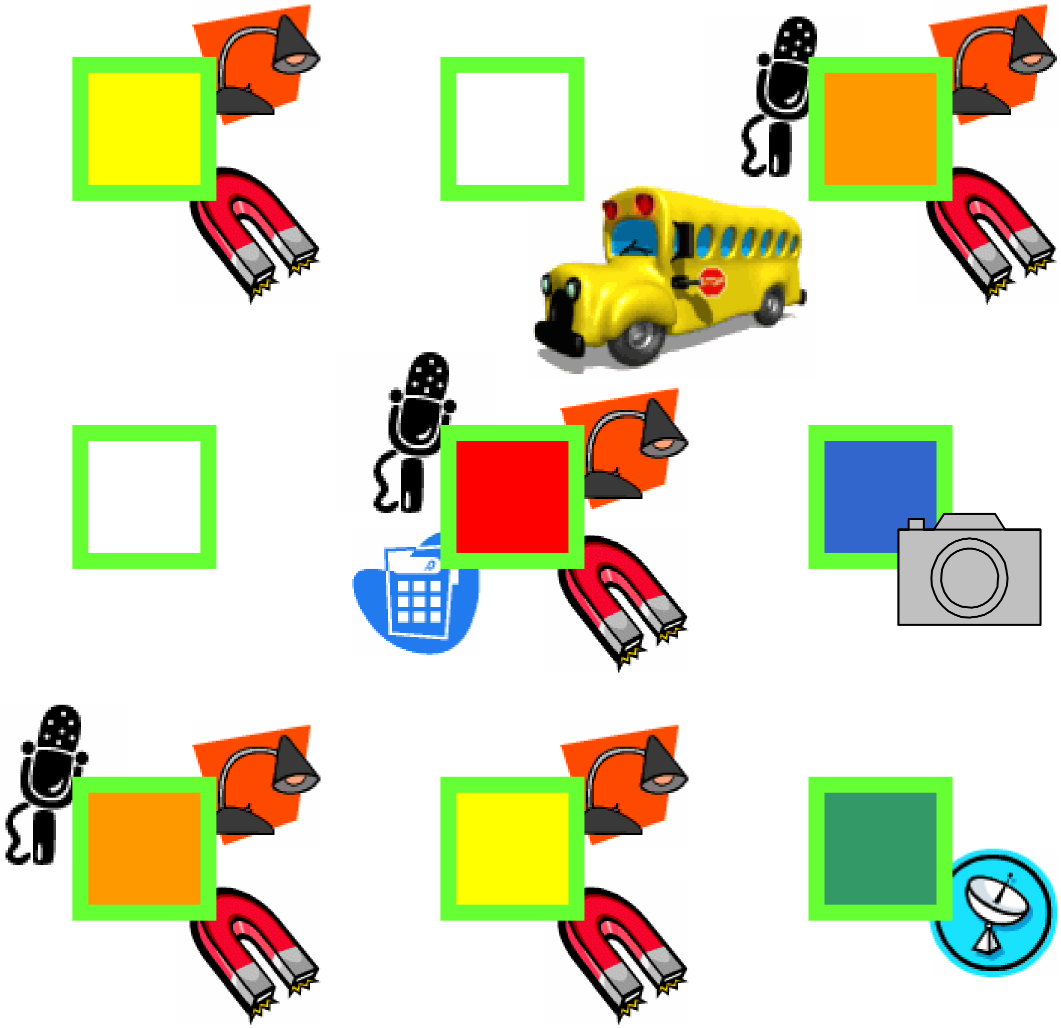
Sensor Coordination using Role-based Programming

Steven Cheung

NSF NeTS NOSS Informational Meeting
October 18, 2005

Motivating example: Object detection, tracking, and classification†



Notation:

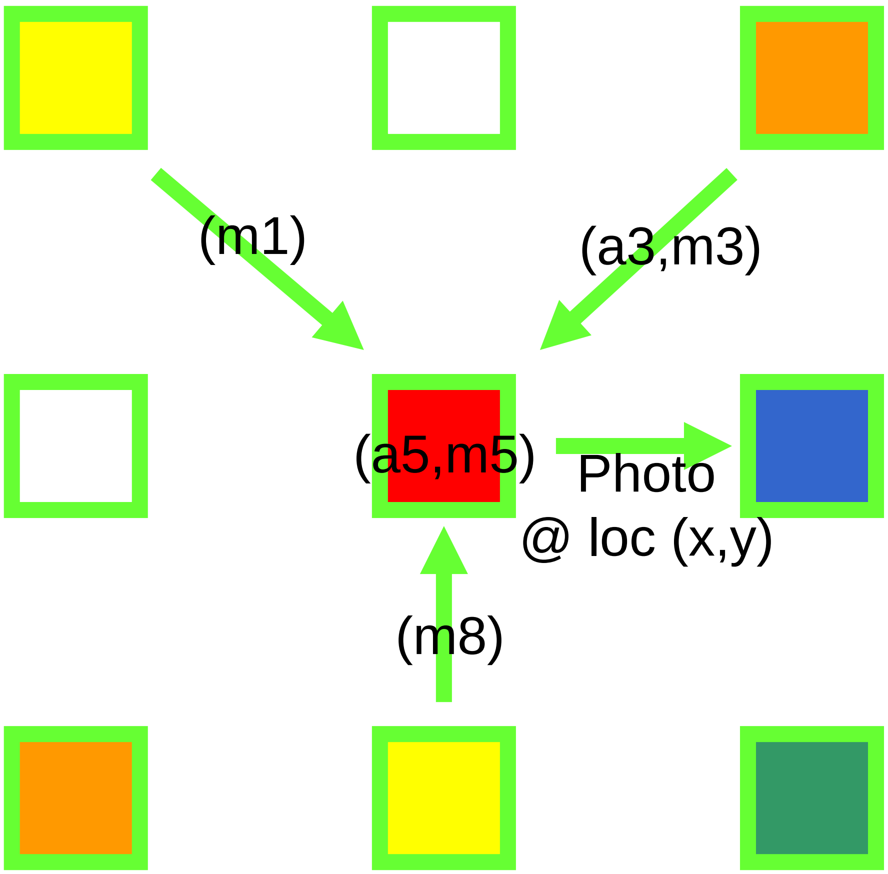
- Sensor node
- Infrared sensor
- Magnetometer
- Acoustic sensor
- Aggregator
- Camera
- GPS module

† Example based on Dutta et al's IPSN'05 paper

Characteristics of the scenario

- **Collaborative processing**
  - **Multiple nodes interact with each other to perform in-network processing**

In-network aggregation



Notation:

ai = acoustic sensor output from node i

mi = magnetometer output from node i

Characteristics of the scenario

- Collaborative processing
  - Multiple nodes interact with each other to perform in-network processing
- **Heterogeneity**
  - **Nodes may have different capabilities**

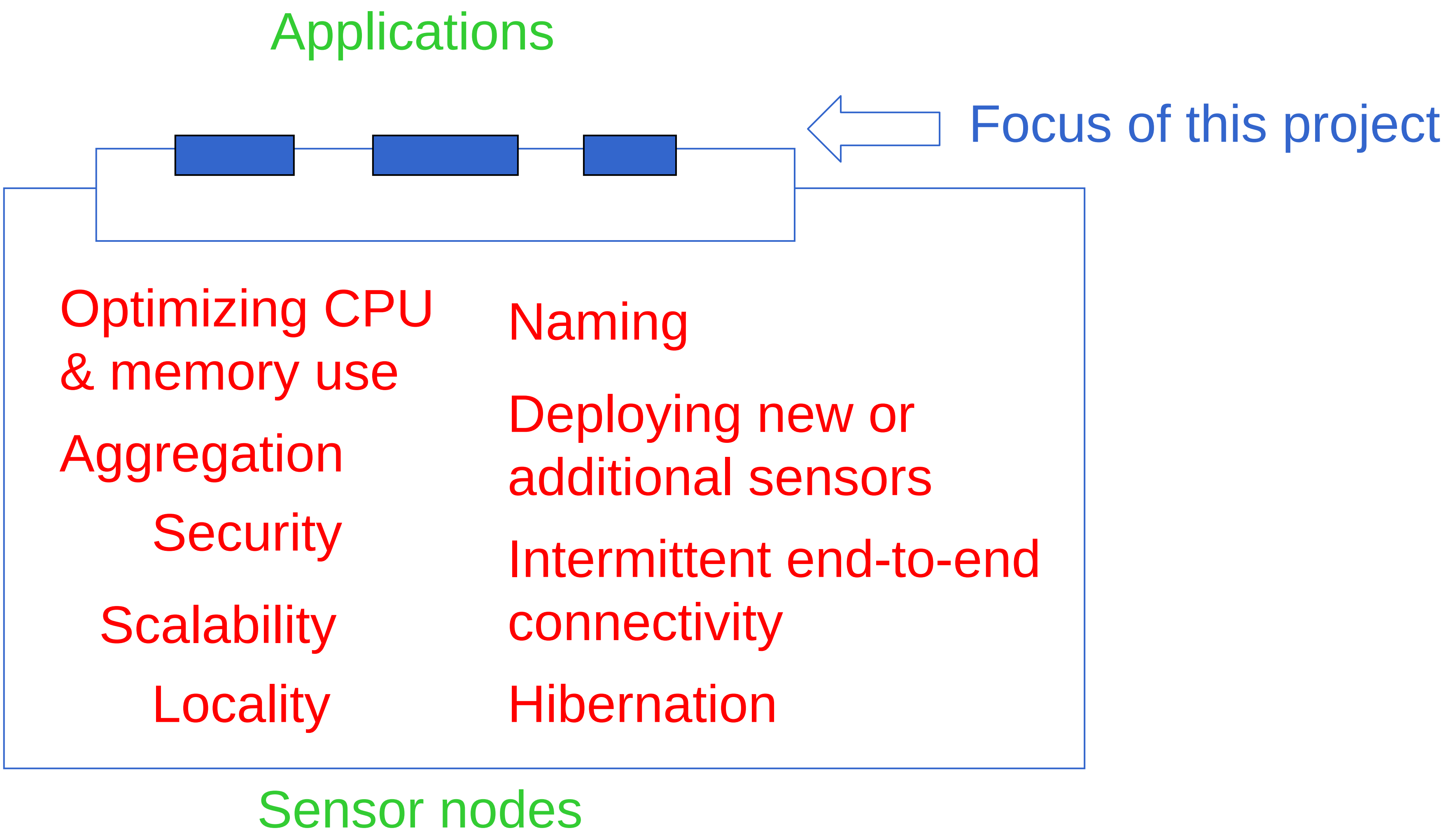
Why use heterogeneous sensor networks?

- Scalability
  - Hierarchical sensor networks
  - Resourceful nodes as cluster heads
- Cost and size constraints
  - Some components may be expensive, and it may not be necessary for all nodes be equipped with all components
  - Number of different components may be more than a node can handle

Characteristics of the scenario

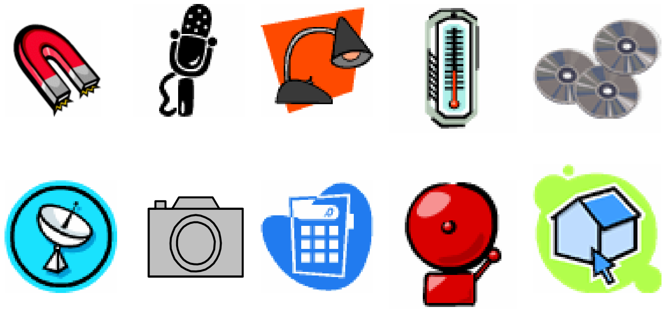
- Collaborative processing
  - Multiple nodes interact with each other to perform in-network processing
- Heterogeneity
  - Nodes may have different capabilities
- **Dynamics of sensor networks**
  - **Nodes, sensors, and actuators may be unavailable, e.g., hibernation to conserve energy**
  - **Network connectivity may change over time**

Goal: High-level programming abstraction



Our approach

- Role-based
  - Nodes play different (sets of) *roles* based on their *attributes*
  - Roles correspond to functions performed by nodes (e.g., providing magnetometer readings)
  - Attributes include hardware configuration (e.g., sensors, processing power, and storage capacity), geographic location, energy reserve, and mobility
- Example roles
  - Temperature sensor
  - Alarm
  - Data store
  - Basestation

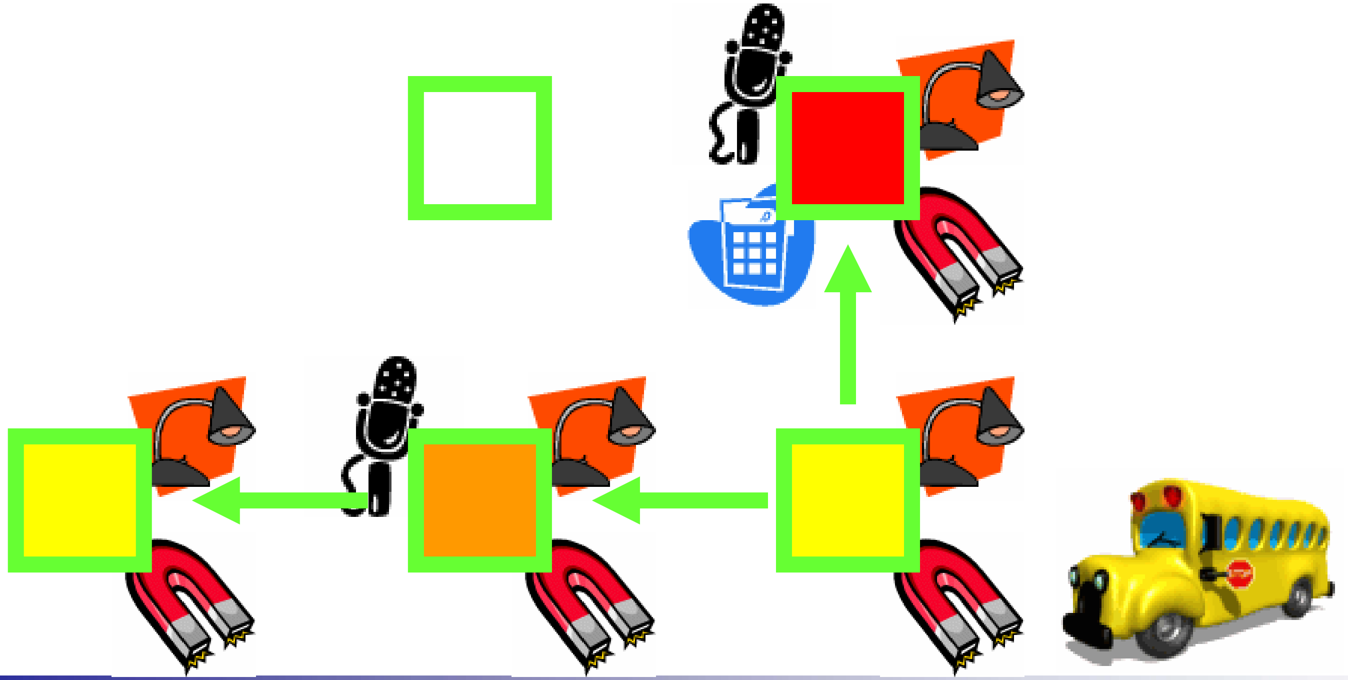


Role advertisement

- Role update
  - Source node id (for distinguishing different role instances)
  - Sequence number
  - For each role:
    - Role name
    - Service coverage
    - Time validity
- Service coverage
  - Specify the set of nodes to serve
  - E.g., nodes within a specified area
- Time validity
  - Specify the time window during which the source will provide services pertaining to the role

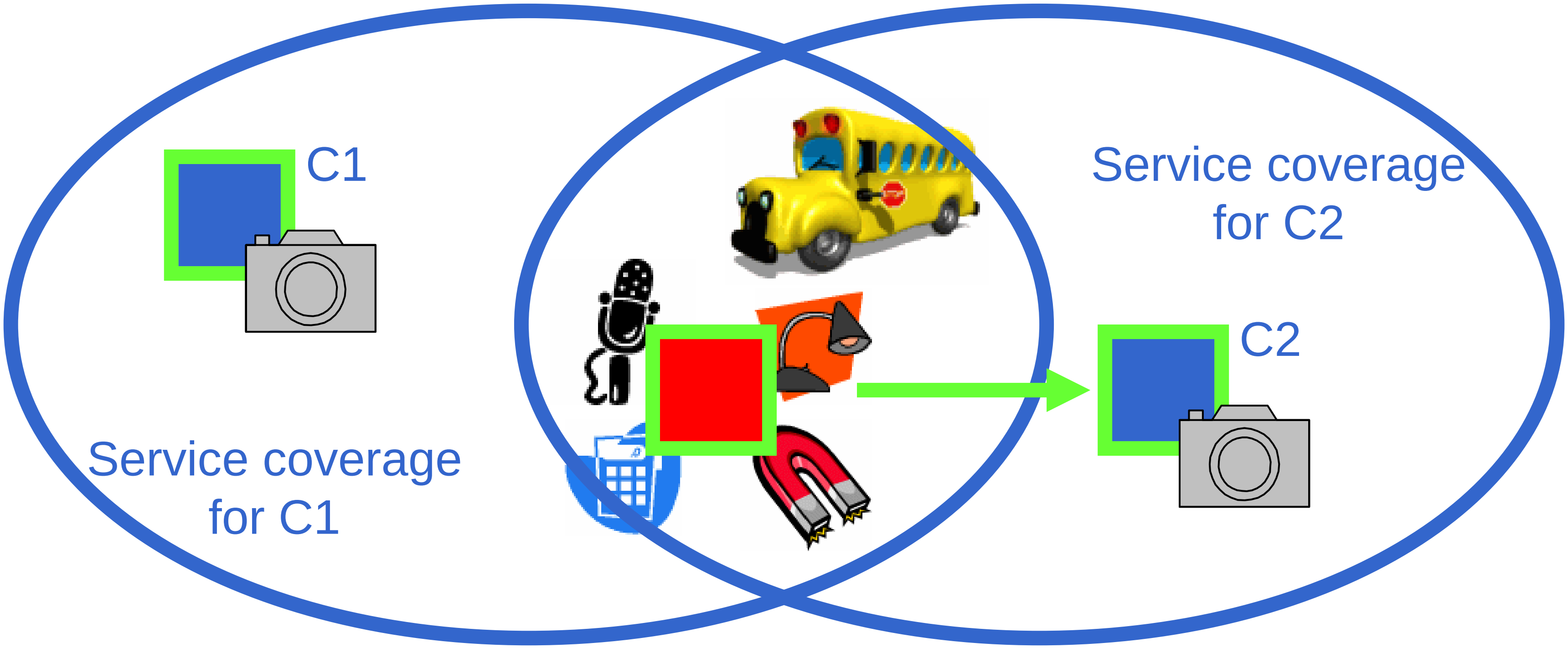
Role-based communication (1)

- Multicast
  - E.g., When nodes with the infrared sensor role detects a "high" reading, they send a message to nodes that play the acoustic sensor and/or the magnetometer roles and are within two hops away to activate them



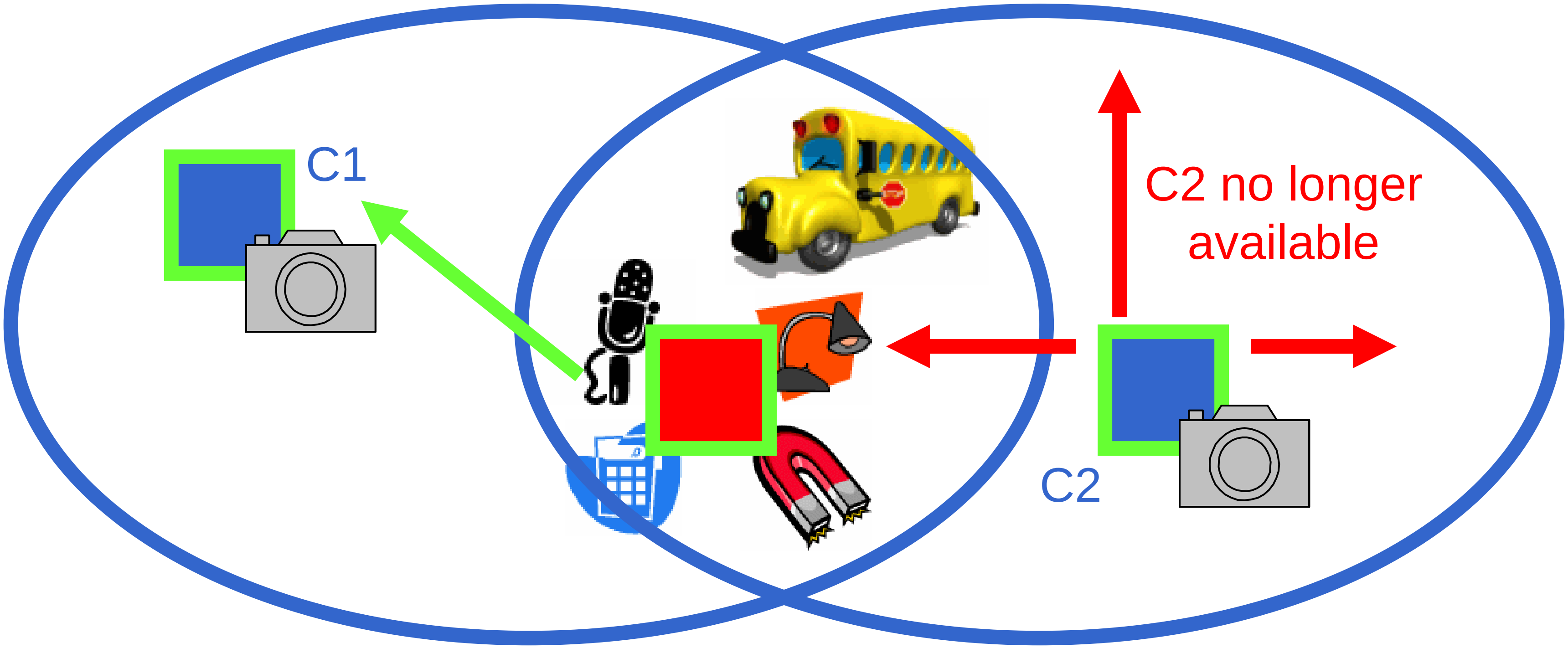
Role-based communication (2)

- Anycast
  - E.g., When a node with the aggregator role detects a vehicle (based on sensor reports received), it sends a request to a node that plays the camera role



Role-based communication (3)

- “Come and go” nodes
  - E.g., When an instance of the camera role decides to go into hibernation, it may send a role advertisement to notify the change to other nodes.



Role management interface

- addRole(*roleID*, *area*, *validity*, *targetRole*)
  - Add role specification for the specified role, service area, service duration, and target role(s) to serve
- removeRole(*roleID*)
  - Remove role specification corresponding to the roleID
- publishRoleAdv(*area*, *validity*)
  - Send a role advertisement update to other nodes specified in the area constraint (e.g., within a specified number of hops from the node). The validity constraint specifies the time interval during which this update is valid.

Summary and status

- Role-based programming abstraction that facilitates sensor coordination with the emphasis on addressing sensor network dynamics and node heterogeneity
- In the process of developing a role-based sensor coordination middleware, called *scorp*, on the nesC/TinyOS platform
- Future work:
  - Evaluation of effectiveness and efficiency
  - Performance optimization
  - Scalability
  - Security

Sensornet Programming Challenges

- Scalability
  - How well does the program perform for a large (say 10k-node) sensor net?
  - Support for heterogeneous sensor net
- Dynamics of sensor networks
  - Nodes that "come and go"
  - How to develop robust programs?
- Tradeoffs among resource usage, reliability, system lifetime, security, costs, ...
  - TinyDB (adjusting sampling frequency based on system lifetime) and abstract region (accuracy vs resource usage)
- Quality of service