Selective Reprogramming in Mobile Wireless Sensor Networks

Bence Pásztor1, Luca Mottola2, Cecilia Mascolo1, Gian Pietro Picco3
1University of Cambridge, Computer Laboratory, University of Cambridge, Cambridge, UK,
2Bruno Kessler Foundation, Trento, Italy
3Dipartimento di Ingegneria e Scienza dell’Informazione, University of Trento, Trento, Italy

- Scenario: **wildlife monitoring** with WSNs
  - nodes attached to animals to monitor their behavior
  - disconnected operation, intermittent connectivity among resource-constrained devices
  - different functionality running on different nodes
    - based on the behavior of the animal
    - different users share the same network

- The system operation may require **reprogramming** the nodes
- **Problem**: reprogramming must target a *subset* of devices, e.g.:
  - all nodes attached to *female* animals that sleep often
  - all nodes attached to animals that remain *close to their setts*
- **Current solutions**: flooding the network
  - a considerable waste of resources if only a subset of nodes is to target!

- **Solution**: a *declarative* programming model to identify the target nodes
  - (static or dynamic) *attributes* to characterize the nodes (locations visited, average temperatures)
  - programmer-defined *constraints* to identify the reprogramming target

- Example: “reprogram all nodes attached to animals that are often close to their setts”
  - **TARGET SET Setts**:
    - **REPROGRAM** ALL WHERE **FREQUENCY** (Location == Sett) > 70%

- Example: “reprogram all nodes attached to animals that visit cold places”
  - **TARGET SET Cold**: **REPROGRAM** ALL WHERE **AVG(Temperature)** < 10°C

- Combining previous targets
  - **TARGET SET ColdSetts**: Setts INTERSECT Cold

- **Reprogramming layer**: loads new programs
- **Runtime layer**: responsible for maintaining attributes and evaluating constraints
- **Communication Layer**: delivers data

- **Step-by-step process**:
  - define the target set using our programming constructs
  - give it as input to a dedicated pre-processor
  - spread constraints among nodes
  - evaluate constraint, and decide if it needs the update
  - pass this information to the routing layer
  - let the routing layer deliver the new code to the required nodes only

- The programming model is independent of the routing layer
- We also devised a routing protocol taking advantage of the *social links* between nodes
- Any other routing can support the programming model

- **Proof of concept**: implementation in the Contiki OS on the Tmote Sky sensor boards using a gossip-based dissemination protocol